Vysoká škola logistiky o.p.s.

Zefektivnění činnosti tvorby simulačního modelu v programu Tecnomatix Plant Simulation

(Bakalářská práce)

Přerov 2022 Filip Dolák

Vysoká škola logistiky o.p.s.

Zadání bakalářské práce

student	**Filip Dolák**
studijní program	LOGISTIKA
specializace	Informatika pro logistiku

Vedoucí Katedry bakalářského studia Vám ve smyslu čl. 22 Studijního a zkušebního řádu Vysoké školy logistiky o.p.s. pro studium v bakalářském studijním programu určuje tuto bakalářskou práci:

Název tématu: **Zefektivnění činnosti tvorby simulačního modelu v programu Technomatix Plant Simulation**

Cíl práce:

Vytvořit aplikaci sloužící jako překladač pro Simulační program, který výrazně ulehčí vytvoření potřebné simulace. Aplikaci rozebrat a vysvětlit, jak funguje, včetně jejich jednotlivých části a zdůvodnit, proč bylo konkrétní řešení použito. Funkčnost aplikace demonstrovat na několika příkladech, které umožní objasnit fungování aplikace krok po kroku.

Zásady pro vypracování:

Využijte teoretických východisek oboru logistika. Čerpejte z literatury doporučené vedoucím práce a při zpracování práce postupujte v souladu s pokyny VŠLG a doporučeními vedoucího práce. Části práce využívající neveřejné informace uveďte v samostatné příloze.

Bakalářskou práci zpracujte v těchto bodech:

Úvod
1. Teoretická východiska počítačové simulace
2. Analýza současného stavu v oblasti počítačové simulace v logistice
3. Návrh zefektivnění tvorby simulačních modelů
4. Zhodnocení návrhů

Závěr

Rozsah práce: 35 – 50 normostran textu

Seznam odborné literatury:

BANGSOW, Steffen. Tecnomatix Plant Simulation: Modeling and Programming by Means of Examples - Second Edition. Springer Nature Switzerland AG, 2020. ISBN 978-3-030-41544-0

BANGSOW, Steffen. Tecnomatix Plant Simulation: Modeling and Programming by Means of Examples. Springer Nature Switzerland AG, 2015. ISBN 978-3-319-19503-2

KEOGH, James. Java bez předchozích znalostí: průvodce pro samouky. Computer Press Brno 2012. ISBN 978-80-251-0839-0

SCHILDT, Herbert. Java7: Výukový kurz. Computer Press Brno 2012. ISBN 978-80-251-3748-2

GROSS Ivan, BARANČÍK Ivan, ČUJAN Zděněk. Velká kniha logistiky. VŠCHT Praha (1. vydání, 2016). ISBN 978-80-7080-952-5

WELLING, Luke. Mistrovství - PHP a MySQL: Kompletni průvodce vývojáře. Computer Press Brno 2017. ISBN 978-80-251-4892-1

Vedoucí bakalářské práce: prof. Ing. Gabriel Fedorko, PhD.

Datum zadání bakalářské práce: 31. 10. 2021
Datum odevzdání bakalářské práce: 6. 5. 2022

Přerov 31. 10. 2021

Ing. et Ing. Iveta Dočkalíková, Ph.D.
vedoucí katedry

prof. Ing. Václav Cempírek, Ph.D.
rektor

Čestné prohlášení

Prohlašuji, že předložená bakalářská práce je původní a že jsem ji vypracoval samostatně. Prohlašuji, že citace použitých pramenů je úplná a že jsem v práci neporušil autorská práva ve smyslu zákona č. 121/2000 Sb., o autorském právu, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon), ve znění pozdějších předpisů.

Prohlašuji, že jsem byl také seznámen s tím, že se na mou bakalářskou práci plně vztahuje zákon č. 121/2000 Sb., o právu autorském, právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon), ve znění pozdějších předpisů, zejména § 60 – školní dílo. Beru na vědomí, že Vysoká škola logistiky o.p.s. nezasahuje do mých autorských práv užitím mé bakalářské práce pro pedagogické, vědecké a prezentační účely školy. Užiji-li svou bakalářskou práci nebo poskytnu-li licenci k jejímu využití, jsem si vědom povinnosti informovat o této skutečnosti Vysokou školu logistiky o.p.s.

Prohlašuji, že jsem byl poučen o tom, že bakalářská práce je veřejná ve smyslu zákona č. 111/1998 Sb., o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších předpisů, zejména § 47b. Taktéž dávám souhlas Vysoké škole logistiky o.p.s. ke zpřístupnění mnou zpracované bakalářské práce v její tištěné i elektronické verzi. Tímto prohlášením souhlasím s případným použitím této práce Vysokou školou logistiky o.p.s. pro pedagogické, vědecké a prezentační účely.

Prohlašuji, že odevzdaná tištěná verze bakalářské práce, elektronická verze na odevzdaném optickém médiu a verze nahraná do informačního systému jsou totožné.

V Přerově, dne DD. MM. 2022 …….

podpis

Poděkování

Chtěl bych zde poděkovat především své rodině za podporu, všem pedagogům, co mi pomohli získat potřebné schopnosti, a také především panu prof. Ing. Liborovi Kavkovi, Ph.D za pomoc s vypracováním softwarové části a svému vedoucímu bakalářské práce panu prof. Ing. Gabrielovi Fedorkovi, Ph.D. za profesionální přístup, ochotu, rady a návrhy při vedení této práce.

Anotace

Cílem bakalářské práce je vytvořit, popsat a vytvořit aplikaci sloužící jako překladač pro Simulační program, který by dokázal výrazně zlehčit vytvoření potřebné simulace. Také následně rozebrat a plně vysvětli, jak daná aplikace funguje, včetně jejich jednotlivých části a objasnění důvodu proč bylo konkrétní řešení použito. Na závěr je aplikace zde demonstrována na několika příkladech, které předvedou funkčnost aplikace a napomohou objasnění fungování aplikace krok po kroku.

Klíčová slova

Logistika, logistická simulace, počítačová simulace, překladač, programovací jazyk, simulační program, schéma

Annotation

The aim of the bachelor thesis is to create, describe and create an application serving as a compiler for the Simulation Program, which was able to significantly simplify the creation of the necessary simulation. Also, then analyze and fully explain how the application works, including their individual parts and clarify why a particular solution was used. Finally, the application is demonstrated on several examples, which will demonstrate the functionality of the application and help clarify the application working step by step.

Keywords

Logistics, logistic simulation, computer simulation, compiler, programming language, simulation program, scheme

Obsah

Úvod

Simulace je nedílnou součástí logistiky, protože plánování a nákup, a příprava na jakoukoliv část logistického systému často velmi náročná na nejrůznější zdroje. Proto je nutné provést před realizací a využití určitého logistického procesu simulaci, aby bylo se bylo možno rozhodnout, jak to co nejefektivněji s dostupnými zdroji provést. Pro vytvoření simulací bylo vytvořena celá řada simulačních programů, s různými funkcemi a po různu uživatelsky přitažlivým ovládáním. Tedy vyžadujícím různou úroveň zkušeností, nebo alespoň pravidel, jak daný simulační software správně využít. Je právě úkolem překladačů zjednodušit, nebo aspoň převést data z jedné námi přijatelné a pochopitelné podoby, do druhé potřebné, nebo obráceně.

Cílem bakalářské práce je vytvoření a rozbor aplikace, která slouží jako překladač jednoduchých, snadno pochopitelných instrukcí, podle kterých lze stanovit různé specifikace logistické simulace do kódu co simulační program dokáže zpracovat a vytvořit podle něj funkční logistickou simulaci se stanovenými parametry. Bakalářská práce to bude prezentovat na několika praktických příkladech, které budou hlouběji rozebrány a bude na nich krok po kroku definováno, jak daná aplikace potřebný kód pro simulaci vytvořila.

V úvodní části se budeme zabývat především teorie potřebnou k úplnému pochopení cílů bakalářské práce. Objasním a vysvětlím důležité pojmy a nahlídneme i do historie vzniku samotné logistiky. Další část bude ta hlavní a bude konkrétně rozebírat jednotlivé příklady na které výsledná aplikace dokáže svou funkčnost. U každého příkladu vysvětlím napřed co by měla simulace umět. Pak objasním, jak jsem využil aplikaci k jejich její vytváření. Dále jak přesně aplikace převádí dané instrukce na kód ve správné formě a nakonec, jak výsledný kód vypadá. V poslední části zhodnotím praktičnost mojí aplikace a případě navrhnu její možné vylepšení, úpravy a případné možnosti dalšího rozšíření funkčnosti.

1 Teoretická východiska počítačové simulace

1.1 Logistika

Pojmem logistika se rozumí celý proces řízení toho, jak jsou zdroje získávány, skladovány a přepravovány do jejich konečného místa určení (Obr. 1.1). Řízení logistiky zahrnuje identifikaci potenciálních distributorů a dodavatelů a stanovení jejich efektivity a dostupnosti. Manažeři logistiky jsou označováni jako logistici.

Pojem „logistika“ byl původně vojenský termín používaný v odkazu na to, jak vojenský personál získával, skladoval a přesouval vybavení a zásoby. Tento termín je nyní široce používán v obchodním sektoru, zejména společnostmi ve výrobních sektorech, k označení toho, jak se zachází se zdroji a jak se pohybují v dodavatelském řetězci.

Jednoduše řečeno, cílem řízení logistiky je mít správné množství zdrojů na vstupu ve správný čas, dostat je na příslušné místo ve správném stavu a dodat je správnému internímu, nebo externímu zákazníkovi.

Efektivní dodavatelský řetězec a efektivní logistické postupy jsou zásadní pro snížení nákladů, údržby a zvýšení efektivity. Špatná logistika vede k předčasným nebo opožděným dodávkám, neuspokojování potřeb klientely a v konečném důsledku způsobuje ztráty a problémy podniku.

V dnešní době technologický rozvoj a složitost logistických procesů stvořily programy pro řízení logistiky a specializované firmy zaměřené na logistiku, které urychlují pohyb zdrojů v dodavatelském řetězci. Jedním z důvodů, proč velcí online prodejci, jako je Amazon, prakticky ovládly maloobchodní prostředí, je celková inovace a efektivita jejich logistiky v každém článku dodavatelského řetězce.

Výrobní společnosti se mohou rozhodnout outsourcovat (předat) řízení své logistiky specialistům, nebo řídit logistiku interně, pokud je to nákladově efektivnější a jsou dostatečně schopni.

Kvalifikovaný logistik naplánuje logistický proces, koordinuje kroky při zásobování a pohyb zdrojů v dodavatelském řetězci. Specializované školení v oblasti řízení dodavatelského řetězce a logistiky jsou často základními nebo volitelnými kurzy nebo dokonce samostatnými studijními programy v oblasti obchodního vzdělávání. Obchodní titul, který klade důraz na tyto dovednosti, nebo v některých případech technický titul v

oblasti systémové analýzy, nebo správy databází je obvykle nezbytný pro zahájení často dobře placené kariéry v logistice. [5]

Obr. 1. 1 Sféry působení logistiky
Zdroj: [7].

1.2 Členění logistiky

Logistika se všeobecně dělí na makrologistiku a miktrologistiku. Makrologistika se sterá o logistické procesy v okolí firmy, které jsou potřeba pro výrobu, od získávání o zásob pomocí těžby, nebo dodavatele, až po dodání výsledného produktu zákazníkovy. Zabývá se globálními aspekty, ať už v rámci blízkého okolí, tak třeba i celého světa. Mikrologistika se naopak zabývá logickými procesy přímo v podniku namísto jeho okolí. Zajímá se tedy například o výrobu, skladování, a podobně.

Logistika se může dále dělit také do dalších různých skupin podle zaměření. Logistika podniku se stará a vztahy mezi dodavatelem a zákazníkem. Nazývá se také v některých případech podnikovou logistikou. Logistika zásobování na druhou se místo toho zajímá o nákup všech materiálů potřebných pro výrobu, jinak řečeno o zásoby. Vnitropodniková logistika se stará přímo o řízení pohybu materiálů v podniku a logistika distribuce se stará o dodávání výsledného produktu klientovy. [5]

1.3 Logistické Technologie

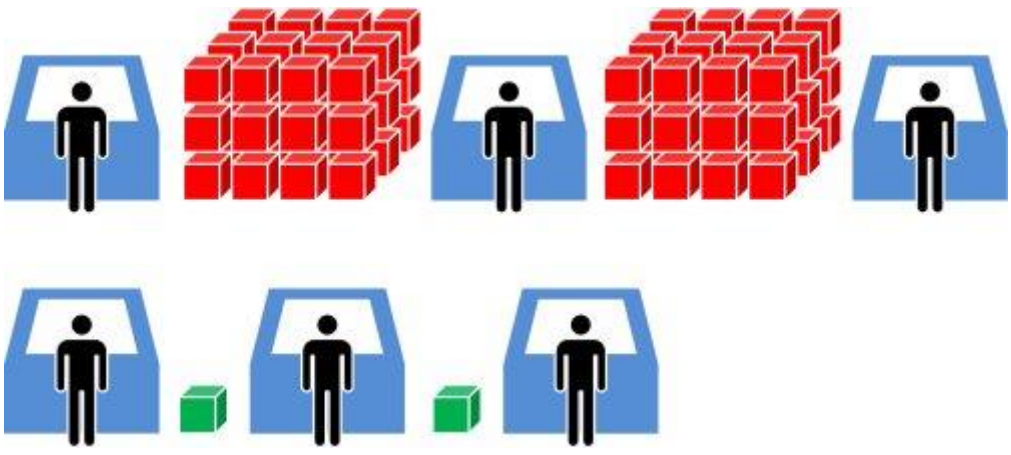
1.3.1 Just-In-Time

Logistická technologie Just-In-Time (JIT) je manažerská filosofie, která slaďuje objednávky surovin od dodavatelů přímo s výrobními plány. Společnosti využívají tuto strategii zásob ke zvýšení efektivity a snížení plýtvání tím, že dostávají zboží pouze tehdy, když je potřebují pro výrobní proces, což snižuje jejich náklady u zásob. Tato metoda vyžaduje, aby výrobní podnik dokázal přesně předpovídat poptávku.

Systém zásob (Obr. 1.2) Just-In-Time (JIT) se tedy snaží minimalizovat zásoby a zvyšovat efektivitu. Výrobní systémy JIT snižují náklady na zásoby, protože výrobci dostávají materiály a díly podle potřeby pro výrobu a nemusí je tedy skladovat a platit náklady spojené s jejich skladováním. Výrobcům také nezůstanou nechtěné zásoby, pokud je objednávka zrušena, nebo není splněna.

Jedním příkladem inventarizačního systému JIT je výrobce automobilů, který operuje s nízkými zásobami, ale silně spoléhá na svůj dodavatelský řetězec, aby dodal díly, které potřebuje k výrobě automobilů, podle potřeby. V důsledku toho výrobce objednává díly potřebné k sestavení vozidel až po obdržení objednávky.

Aby výroba JIT uspěla, společnosti musí mít stabilní výrobu, vysoce kvalitní zpracování, bezporuchové strojní zařízení, spolehlivou obsluhu a spolehlivé dodavatele. [5]



Obr. 1. 2 Just-In-Case VS Just-In-Time
Zdroj: [8].

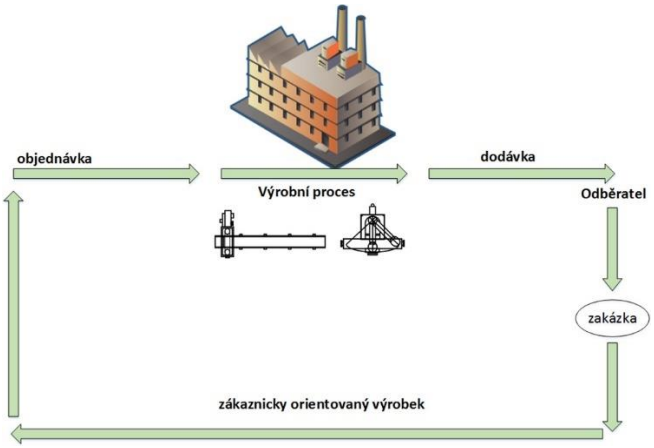
1.3.2 Kanban

Kanban je systém řízení zásob používaný ve výrobě just-in-time (JIT). Vyvinul jej Taiichi Ohno, průmyslový inženýr v Toyotě, a jeho název je odvozen od barevných karet, které sledují výrobu a objednávají nové dodávky dílů nebo materiálů, jakmile dojdou. Kanban je japonský výraz pro znak, takže systém kanban jednoduše znamená používat vizuální podněty k vyvolání akce potřebné k udržení plynulosti procesu.

Systém kanban lze chápat jako systém signálu a odezvy. Když na provozní stanici dochází položka, zobrazí se vizuální nápověda upřesňující, kolik si ze zásoby objednat. Osoba používající díly provede objednávku na množství uvedené v kanbanu a dodavatel poskytne přesné požadované množství (Obr. 1.2).

Systém kanban lze snadno použít v rámci továrny, ale lze jej také použít pro nákup zásob od externích dodavatelů. Systém kanban vytváří mimořádnou viditelnost jak pro dodavatele, tak pro kupující. Jedním z jeho hlavních cílů je omezit hromadění přebytečných zásob v kterémkoli místě výrobní linky. Stanoví se limity pro počet položek čekajících na zásobovacích místech a poté se sníží, když se zjistí a odstraní neefektivita. Kdykoli je překročen limit zásob, ukazuje to na neefektivnost, kterou je třeba řešit.

Jak se kontejnery s díly nebo materiály vyprazdňují, objevují se karty, barevně odlišené v pořadí priority, což umožňuje výrobu a dodání více, než dojde k zadržování nebo nedostatku. Často se používá systém dvou karet. Přepravní karty T-kanban opravňují k přesunu kontejnerů na další pracovní stanici na výrobní lince, zatímco výrobní karty P-kanban opravňují pracovní stanici k výrobě pevného množství produktů a objednávání dílů, nebo materiálů po tom co byli prodány, nebo použity. [5]



Obr. 1. 3 Kanban - výroba tahem
Zdroj: [9].

1.3.3 Quick Response

Quick Response kód, nebo jen QR kód je typ čárového kódu, který lze snadno přečíst digitálním zařízením a který ukládá informace jako řadu pixelů v mřížce čtvercového tvaru. QR kódy se často používají ke sledování informací o produktech v dodavatelském řetězci a často se používají v marketingových a reklamních kampaních.

QR kódy jsou považovány za pokrok od starších jednorozměrných čárových kódů a byly schváleny jako mezinárodní standard v roce 2000 Mezinárodní organizací pro standardizaci (ISO).

QR kódy byly vyvinuty v 90. letech minulého století jako způsob, jak poskytnout více informací než standardní čárový kód. Vynalezla je společnost Denso Wave, dceřiná společnost Toyoty, jako způsob, jak sledovat automobily během výrobního procesu. Na rozdíl od čárové kódy, které používají paprsek světla, který se odrazí z rovnoběžných čar, QR kódy mohou být digitálně snímány zařízeními, jako jsou například mobilní telefony.

QR kódy se skládají z černých čtverců uspořádaných do mřížky (matice) na bílém pozadí a jsou čteny specializovaným programem, který je schopen extrahovat data ze vzorů, které jsou v matici přítomny. Tyto kódy jsou schopny obsahovat více informací než tradiční čárové kódy a primárně zpracovávají čtyři režimy dat: alfanumerický, numerický, binární a Kanji (japonské znaky/písmo).

Navzdory zvýšené datové kapacitě nebyly QR kódy u spotřebitelů tak oblíbené, jak se očekávalo. Spíš, než aby byly vytvořeny spotřebiteli ke sdílení informací, jsou nejčastěji spojovány s inzerenty a marketingovými kampaněmi.

QR kódy se staly více rozšířené při usnadňování digitálních plateb a kryptoměna systémech, jako je zobrazení něčí Bitcoin adresu. QR kódy se také stále častěji používají k přenosu webových adres do mobilních telefonů (Obr. 1. 4).[5]

Obr. 1. 5 Skenování QR mobilem
Zdroj: [10].

1.3.4 Efficient Consumer Response

Efficient Consumer Response, či ECR je kolaborativní obchodní strategie založená na standardech, dovednostech a systematické spolupráci mezi interními obchodními divizemi a společnostmi zapojenými do hodnotového řetězce. Prvořadým cílem je co nejefektivněji uspokojit potřeby zákazníků. Každý, kdo je zapojen do hodnotového řetězce, se musí zaměřit na výhody pro spotřebitele. Jedině tak lze odhalit příležitosti a dosáhnout zákaznických výhod, které by jedna divize nebo jedna společnost jednající samostatně nedokázala realizovat.

Cíly ECR jsou tedy poskytování služeb zákazníkům a zvyšování prodeje při současném snižování náklady. Obchodní model ECR je založen na spolupráci napříč společnostmi a zahrnuje holistický koncept standardů, procesních modelů a metod.

Vizí konceptu ECR je uspokojit přání spotřebitelů rychle, levně a efektivně, Tohle všechno by nejlepším případě mělo být udržitelné a zvládat uspokojovat potřeby, ještě dřív než, si je vůbec klienti uvědomí.

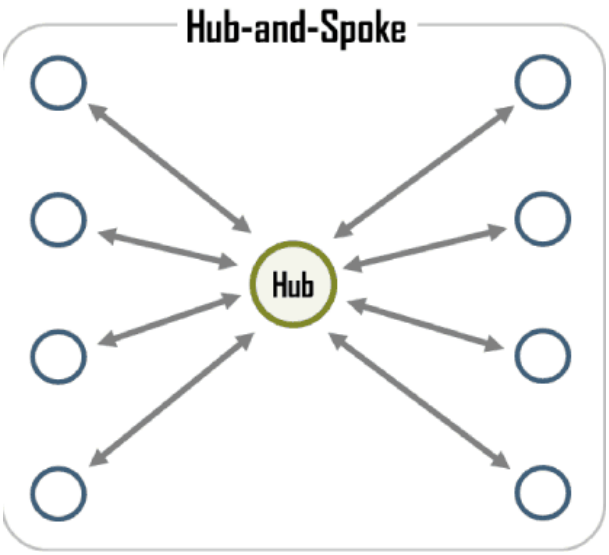
ECR kombinuje strategické, procesní a technické aspekty k odstranění neefektivnosti v hodnotovém řetězci. [5]

1.3.5 Hub and Spoke

Struktury Hub and Spoke, zkráceně H&S využívají investiční společnosti ke sdružování aktiv, snižování nákladů a zvyšování efektivity. Několik investičních nástrojů, z nichž každý zůstává samostatně spravován, kombinuje svá aktiva a přispívá do jednoho centrálního nástroje (Obr. 1.5). Toto může být také nazýváno strukturou master-feeder.

Všechny fondy v systému mají obvykle stejný investiční cíl a správce portfolia nebo hlavní fond, který slouží jako „hub“. Menší investiční nástroje neboli feeder fondy se označují jako „spoke“.

H&S poskytuje výhody správcům investičních fondů tím, že nabízí četné výhody z jejich sdílené struktury. Díky struktuře H&S je kapitál směrován do hlavního fondu, kde se provádějí všechny transakce, což pomáhá snižovat transakční náklady. [5]



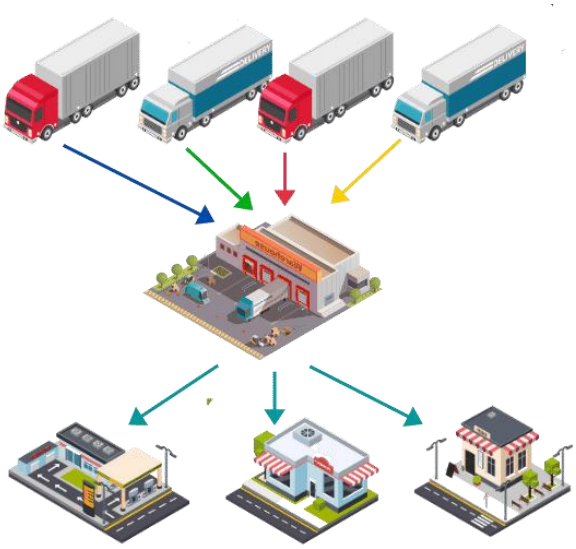
Obr. 1. 6 Propojení H&S
Zdroj: [11].

1.3.6 Cross–Docking

Cross-docking je logistický postup, při kterém jsou produkty od dodavatele, nebo výrobního závodu distribuovány přímo zákazníkovi, nebo maloobchodnímu řetězci s minimální, nebo žádnou dobou manipulace, nebo skladování. Křížové dodávání probíhá v distribučním centru (Obr. 1.6), obvykle sestávající z nákladních vozidel a dvou docích na (příchozí a odchozí strany) s minimálním úložným prostorem.

Název „cross-docking“ vysvětluje proces přijímání produktů přes příchozí dok a jejich následné přenášení přes dok do odchozího přepravního doku.

Takže distribuční centra prakticky nic neskladují a jen třídí, kompletují a předávají zásilky, takže nepotřebují (nebo by správně neměly potřebovat) velký úložný prostor, protože se zboží v distribučním centru prakticky neskladuje. [5]



Obr. 1. 7 Fungování cross-docking
Zdroj: [12].

1.3.7 Just-In-Case

Just-In-Case (JIC) je strategie zásob, kdy společnosti drží velké množství zásob v rezervě. Jedná se v podstatě o pravý opak JIT. Tento typ strategie řízení zásob má za cíl minimalizovat pravděpodobnost, že se produkt vyprodá.

Společnost, která používá tuto strategii, má obvykle potíže s předpovědí spotřebitelské poptávky, zažívá velký nárůst poptávky v nepředvídatelných časech, nemůže si být jistý plynulostí dodávek, nebo věří v maximální jištění i a za cenu větších nákladů.

Společnosti praktikující tuto strategii v podstatě vznikají vyšší náklady na držení zásob výměnou za snížení počtu ztrát tržeb v důsledku vyprodaných zásob hotových produktů, mají zajištěné zásoby pro pokračující produkci na určitý čas i v případě problémů na s straně dodavatele a jsou schopni rychle reagovat na závadu, ať už jde o předělání nepovedeného výrobu, nebo reagování na objednávku na poslední chvíli. [5]

1.4 Průmysl 4.0

Průmysl 4.0 je označení čtvrté průmyslové revoluce (Obr. 1.7), která začal přibližně v roce 2015 a stále trvá. Nejjednodušeji by se dalo říct že principem za Průmyslem 4.0 je digitalizace a automatizace výroby. Svět je tedy uprostřed zásadního vývoje ve způsobu výroby produktů, který je hluboce spjat s budoucností internetu věcí (IoT).

Pokroky v sítích, strojovém učení, analýze dat, robotice, 3D tisku a dalších technologiích výrazně zlepšují průmyslové procesy a snižují naši závislost na lidské práci a rozhodování. Opíráním se o digitální řešení může výroba snížit možnost chyby kvůli lidské účasti, zkrátit dobu uvedení na trh a zvýšit rychlost, s jakou se průmyslové procesy mohou přizpůsobit novým informacím.

Využitím průmyslového internetu věcí (IIoT), kybernetických fyzických systémů a internetu služeb mohou „chytré továrny“ pomoci operátorům činit rozhodnutí na základě dat, nebo dokonce automaticky spouštět potřebné procesy. Sdílení dat mezi zařízeními umožňuje těmto továrnám sledovat produkty s naprostou přesností, když procházejí zařízením, pomocí senzorů k zaznamenávání pokroku a shromažďování cenných informací.

Ve spojení s novou infrastrukturou mohou Chytré Továrny vyrábět produkty hromadně, a přitom zůstat dostatečně přizpůsobiví, aby efektivně vytvářely specializované produkty na základě individuálních požadavků zákazníků, tak aby to bylo z pohledu zisku výhodné. Jsou také schopny dynamicky reagovat na nepředvídané výzvy, které by mohly významně narušit výrobní operace, jako je například změna dodavatelů, technické potíže a jiné neočekávatelné situace a krize.

Jak z názvu vyplívá Průmysl 4.0, znázorňuje čtvrtou průmyslovou revoluci, takže logicky byly i některé před ní. První tři průmyslové revoluce vznikly jako výsledek mechanizace, elektřiny a IT. Nyní zavádění internetu věcí a služeb do výrobního prostředí značí právě čtvrtou průmyslovou revoluci.

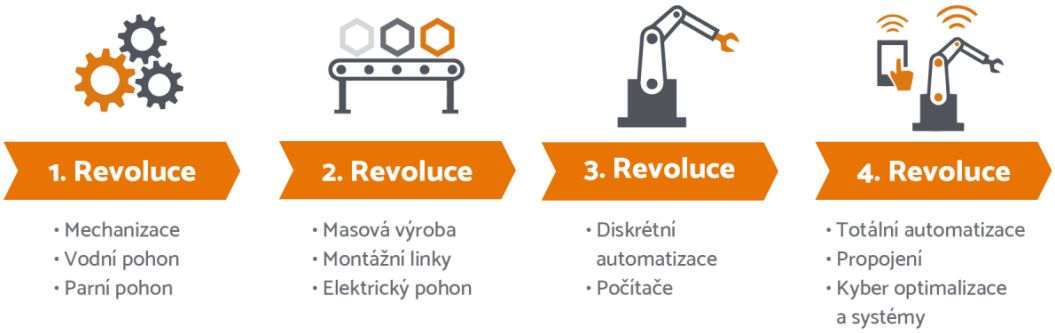
Od té doby se termín a doporučení pracovní skupiny Průmysl 4.0 prosadily po celém světě, poháněly budoucnost výroby a vyvolaly novou vlnu akademické diskuse kolem čtvrté průmyslové revoluce.

V době, kdy pracovní skupina Průmysl 4.0 vypracovala svá doporučení, některé chytré továrny již přijaly určité aspekty Průmyslu 4.0. Přední světoví výrobci od té doby rychle investovali do těchto technologií a zaváděli změny ve své infrastruktuře.

Globální diskuse o Průmyslu 4.0, navzdory jejímu významu ve výrobním světě, často postrádala celistvost a soudržnost. Dá se tedy obecně říci, že Průmysl 4.0 popisuje rostoucí trend k automatizaci a výměně dat v technologiích a procesech ve zpracovatelském průmyslu, jedná se tedy především o tyto technologie:

- internet věcí (IoT),

- průmyslový internet věcí (IIoT),
- kybernetické fyzické systémy (CPS),
- chytrá výroba,
- chytré továrny,
- cloud computing,
- kognitivní výpočty,
- umělá inteligence. [5]



Obr. 1. 8 Rozdělení průmyslových revolucí
Zdroj: [13].

1.4.1 Simulace v Průmyslu 4.0

Simulace je jednou z hlavních částí Průmyslu 4.0 a úzce souvisí s dalšími aspekty, jako je například systémová integrace, umělá inteligence a využití dat k vyhodnocování problémů a hledání jejich řešení, nebo dokonce hledání zlepšení.

Vzhledem k tomu, že Průmysl 4.0 počítá se stále větší automatizací procesů a s nemožností zastavení a testování, používání digitálních dvojčat v systémech se stává velice důležité. Díky tomu mohou vedoucí dělat svá rozhodnutí na základě přesných výsledků, a to bez nutnosti rutinní implementace.

Mezi výhody simulací v Průmyslu 4.0 patří to, že při využití internetu věcí (IoT) se stává realitou mezi systémové zpracování dat a komunikace mezi systémy stávají realitou. Všechny takhle získaná data tak napomáhají přesnějším simulacím. Také je možné díky tomu vyhodnotit a (následně řešit), problémy ve vícero oblastech společností současně. Díky tomu se získá celkovější obraz pro snadnější rozhodování

Do vytvořeného digitálního dvojčete, jsou zapojeny všechny systémy a procesy v průmyslovém závodě. To umožňuje shromáždit vysoce přesné a užitečné informace, a to

tak přesně jako by byly prováděny testy v reálném světě. Což by samozřejmě bylo náročnější a komplikovanější, (nebo dokonce nereálné).

Zatímco v netechnologických průmyslových závodech jsou za strategická rozhodnutí zodpovědní pouze lidé, Díky automatizace a zapojení umělé inteligence v Průmyslu 4.0 může spoustu rozhodnutí dokonce učinit přímo podle stanovených pravidel počítačové systémy, a to bez potřeby což má za následky redukce nutné lidské práce a zrychlení rozhodovacího procesu.

Totéž se samozřejmě děje při provádění simulace, protože díky datům a specifikacím definovaných před provedením dané simu simulace mohou systémy identifikovat poruchy, nebo problémy a následně pomocí zpráv/hlášení, i navrhnout potenciální řešení.

U testy prováděné v rámci průmyslového odvětví, je třeba proměnné kontrolovat jednotlivě, takže tyto testy jsou časově náročné, nebo dokonce v nejsou zcela kompletní pro všechny posuzované možnosti. Díky tomu je velkou výhodou simulací jejich možnosti sledování mnoha proměnných a možných navrhovaných systémů. Díky tomu je možné sledovat a následně vybrat nejvýhodnější možnost.

Simulace v Průmysl 4.0 pomáhají i obyčejným zaměstnancům. Protože simulace dokáží identifikovat opakující a nebezpečné procesy dokáží vedoucí rozhodnout, a zorganizovat pořadí procesů tak, aby byly nejenom rychlejší a efektivnější, ale aby i zlepšily pracovní podmínky pro zaměstnance, kterých se dotýkají.

To, jestli je pro zaměstnance výhodné, že spoustu procesů mohou obsloužit autonomní roboti, kteří je dokáží ve většině případů vykonávat jak efektivněji, tak bezpečněji, sice umožní osvobodit zaměstnance od určitých prací, ale na druhou stranu tím omezí počet potřebných pracovních pozic. Firma potřebuje méně zaměstnanců.

Díky všem výhodám, až už zmíněných, nebo i ne je vše co přináší Průmysl 4.0 skvělou inovací, a to včetně simulací. Spousta aspektů v Průmysl 4.0 je přímo spojená se simulacemi (např. Digitální dvojče) a bez jejich zapojení by nebyl potenciál Průmysl u 4.0 z daleka tak využit. Umožňuje hodnocení a testy pro analýzu fungování továrny bez obtíží. Které by způsobily při jejich implementaci v reálném světě. Navíc díky nic dochází, k snadnějšímu nalezení zlepšení procesů a jejich simulací díky kterému lze činit rozhodnutí pro zamezení možné ztráty v důsledku zavedení nesprávného a nedostatečně ověřeného postupu. [5]

1.4.2 Digitální dvojče

Digitální dvojče je digitální kopie, či model fyzického objektu (Obr. 1.8), procesu nebo služby. Digitální dvojče může být tedy duplikací prakticky jakéhokoli objektu ve fyzickém světě. Může, se jednat o určité části strojů, jako jsou motory, přes celá auta včetně jednotlivých součástí, budovy, nebo dokonce města.

Digitální dvojče, jak bylo řečeno nemusí být kopií fyzického objektu, ale může se jednat a repliku určité činnosti, či procesu kvůli získání dat pro analýzu. Využívá například na simulaci určitých výrobních systémů v Průmyslu 4.0.

Digitální dvojče je tedy v zásadě počítačový program, který využívá data získaná z reálného světa k vytváření dané simulací, které dokáže díky datům přesně simulovat a předvídat jak by daný objekt, nebo proces pracoval v reálném světě. Tyto využívá programy integraci mnoha technologií z Průmyslu 4.0 jako jsou například internet věcí a umělá inteligenci, za účelem získání co nejlepšího výsledku.

V dnešní době technologických a softwarových inovací, kdy je potřeba plánovat a pracovat se stále většími projekty, se staly tyto virtuální kopie základem plánování, a inovací, až už pro inženýrství, továrny, a spoustu dalších. Navíc s rozvojem technologií se odvětví využívají digitální dvojče stále rozšiřují, je docela možné, že se v budoucnu vyplatí i například v medicíně.

Obr. 1. 9 Ukázka digitalizace reálného objektu
Zdroj: [14].

Zaokrouhleno tedy, vytvoření virtuálního dvojčete nám dokáže umožnit díky simulování vylepšování procesů, předcházení rizikům a poruchám. Také díky nim a pokročilým monitorovacím a analytickým možnostem testovat konkrétní služby a procesy, abychom věděly, zda fungují přesně tak jaká byla představa.

Zjednodušeně se dá říci, že digitální dvojče se dá rozdělit do tří skupin. Prví skupinou je Digitální dvojčete produktu. Jedná se o digitální reprezentaci produktu a spojuje návrh a řízení celého životního cyklu produktu ve výrobě. Je zkonstruován jako součást procesu výzkumu a vývoje a pomáhá při vývoji produktů tím, že umožňuje simulovat a testovat produkt v rané fázi procesu.

Další je digitální dvojče produkčního aktiva. Jedná se o digitální dvojče jednoho nebo více produkčních aktiv se používá pro návrh, virtuální spuštění a pokračující provoz. Důraz je kladen na simulaci operací aktiva, nastavení a optimalizaci jeho klíčových parametrů a umožnění konceptů, jako je prediktivní údržba nebo rozšířená realita.

Poslední a pro práci nejdůležitější je digitální dvojče továrny. To pomáhá přesně a efektivně naplánovat, navrhnou a následně stavět továrny a další infrastruktury. Může být použito například k testování fungování skrze simulace a následně udržování budovy v efektivním provozu. [5]

1.4.3 Stavba simulačního modelu

Když si analytici přejí studovat systém, prvním obecným krokem je sestavení modelu. Pro většinu simulačních účelů by se jednalo o statisticky založený model, který se tam, kde je to možné, opírá o empirické důkazy. Takový model by byl matematickou abstrakcí, která aproximuje realitu zkoumané situace. Vyvažování potřeby detailů s potřebou mít model, který bude přístupný rozumným technikám řešení, je neustálým problémem. Bohužel neexistuje žádná záruka, že model lze úspěšně postavit tak, aby přesně odrážel vztahy v reálném světě, které jsou ve hře. Pokud lze sestavit platný model a pokud má systém nějaký prvek, který je náhodný, ale je definován specifickým vztahem pravděpodobnosti,

Zvažte příklad letecké ambulance. Náhodné procesy ovlivňující provoz takového systému zahrnují výskyt havárií, místa takových havárií a to, zda je či není počasí schopné letu. Ve hře mohou být jistě i další náhodné faktory, ale analytici možná zjistili, že tyto jsou

všechny významné. Obvykle by analytici vyvinuli program, který by simuloval provoz systému po nějakou vhodnou dobu, řekněme měsíc. Potom by se vrátili a simulovali mnoho dalších měsíců aktivity, zatímco prostřednictvím vhodného počítačového programu shromažďovali pozorování průměrné doby letu, průměrné doby odezvy, počet obsluhovaných pacientů a další proměnné, které považovali za zajímavé. Mohou velmi dobře simulovat stovky nebo dokonce tisíce měsíců, aby získali rozdělení hodnot důležitých proměnných. Získali by tak rozložení těchto proměnných pro každou konfiguraci služby, řekněme počet vrtulníků a jejich umístění, což by umožnilo porovnat různé konfigurace a možná tu nejlepší identifikovat pomocí jakéhokoli vhodného kritéria.

1.4.4 Internet věcí

Stručně řečeno internet věcí, nebo jinak internet of things zkráceně IoT, je koncept připojení jakéhokoli zařízení (pokud má vypínač) k internetu a dalším připojeným zařízením. IoT je obrovská síť propojených funkcí a lidí, kteří všichni shromažďují a sdílejí data o způsobu, jakým jsou využíváni, a o prostředí kolem nich.

To zahrnuje mimořádné množství objektů všech tvarů a velikostí (Obr. 1.9), od chytrých mikrovlnek, které automaticky vaří vaše jídlo na správnou dobu, přes samořídící auta, jejichž složité senzory detekují předměty v jejich dráze, až po nositelná fitness zařízení, která měří svou tepovou frekvenci a počet kroků, které jste ten den udělali, a poté použijte tyto informace k navržení cvičebních plánů přizpůsobených vám. Existují dokonce připojené fotbalové míče, které mohou sledovat, jak daleko a rychle jsou vrženy, a zaznamenávat tyto statistiky prostřednictvím aplikace pro budoucí tréninkové účely. [5]

Obr. 1. 10 Možnosti aplikování IoT
Zdroj: [15].

1.4.5 Průmyslový internet věcí

Průmyslový internet věcí, nebo také industrial internet of things (IIoT) je označení pro všechny různé sady hardwarových součástí, které spolupracují prostřednictvím konektivity internetu věcí a pomáhají zlepšit výrobní a průmyslové procesy. Když lidé mluví o průmyslovém internetu věcí, mluví o všech senzorech, zařízeních a strojích, které přispívají k fyzickým obchodním procesům v průmyslovém prostředí. Naproti tomu, když lidé mluví o internetu věcí obecně, mluví o všech připojených zařízeních, která odpovídají modelu IoT – například když lidé přemýšlejí o internetu věcí, často přemýšlí o chytrých domácích zařízeních, která jsou propojena. společně poskytovat spotřebitelské pohodlí.

IIoT přináší spoustu výhod do výroby a průmyslu (Obr. 1.10), jednou je například to, jakým způsobem zajišťují kvalitu výsledného produktu. Zařízení IIoT mohou automatizovat určité monitorovací funkce, aby se snížila lidská chyba při zajišťování kvality (QA). Pokud je například díl vyroben nesprávně, kamera nebo jiný senzor připojený ke stroji může detekovat chybu dříve, takže se díl ve výrobním procesu neposune dopředu, a tak lze vyrobit opravený díl.

Další velkou výhodou je bezesporu, že přináší možnost monitorování strojů. Díky neustálému sledování stavu a zdraví strojů lze poruchy zachytit dříve. Pro sledování celkového stavu strojů lze použít teplotní senzory. Vibrační senzory mohou monitorovat stav součástí stroje. Pokud se zdravotní stav jedné součásti začne zhoršovat, lze ji vyměnit, čímž se zabrání poruše, která by mohla způsobit další poškození stroje. [5]

Obr. 1. 11 Aplikační možnosti IoT
Zdroj: [16].

1.4.6 Kybernetické fyzické systémy

Cyber-physical Systems (CPS) jsou kombinací výpočtů, sítí a fyzických procesů, ve kterých mají vestavěné výpočetní algoritmy a sítě schopnost monitorovat a řídit fyzické komponenty.

Pomocí kombinace strojů, senzorických zařízení, vestavěné výpočetní inteligence a různých komunikačních mechanismů monitoruje CPS fyzické prvky pomocí počítačových algoritmů propojených s internetem. To znamená, že jsou schopny autonomně fungovat na základě svého fyzického okolí.

S ohledem na pokrok v analytice, umělé inteligenci (AI) a komunikacích se zvyšuje poptávka po inteligentních strojích, které dokážou komunikovat s okolním prostředím, jako jsou auta bez řidiče, která monitorují okolí a komunikují s ním, a chytrá zařízení, která optimalizují spotřebu energie (Obr. 1.11). CPS stimulují významné změny v kvalitě života a tvoří základ chytré infrastruktury, produktů a služeb. [5]

Obr. 1. 12 Koncept CPS
Zdroj: [17].

1.4.7 Chytrá výroba

„Smart Manufacturing“, v referenčních pracích často zkráceně SM a označovaný jako „inteligentní výroba“, odkazuje na novou globální průmyslovou metodu, která do značné míry spoléhá na vývoj nejnovějších technologií, pokud jde o propojené výrobní prostředky během výrobního procesu.

Jinými slovy, jde o nastavení systémů v továrnách, ve kterých jsou stroje propojeny navzájem, ale také, a především s internetem, aby bylo zajištěno optimální a škálovatelné řízení výrobních procesů.

V pracovním prostředí, které uplatňuje principy Smart Manufacturing, se snažíme zautomatizovat co nejvíce operací tak, aby byly prováděny s maximální efektivitou (Obr. 1.12). Cílem je provést je rychleji, zaručit kvalitu a ziskovost při nižších nákladech.

K tomu je ústředním prvkem typu organizace analýza dat shromážděný přímo ze zařízení je digitálně analyzován, aby bylo možné učinit správná rozhodnutí a provést úpravy za účelem zlepšení a optimalizace výrobního výkonu.

Inteligentní výroba je jednou z aplikací IoT, přesněji řečeno IIOT, což je průmyslový internet věcí. [5]

Obr. 1. 13 Ukázka chytré výroby
Zdroj: [18].

1.4.8 Chytrá továrna

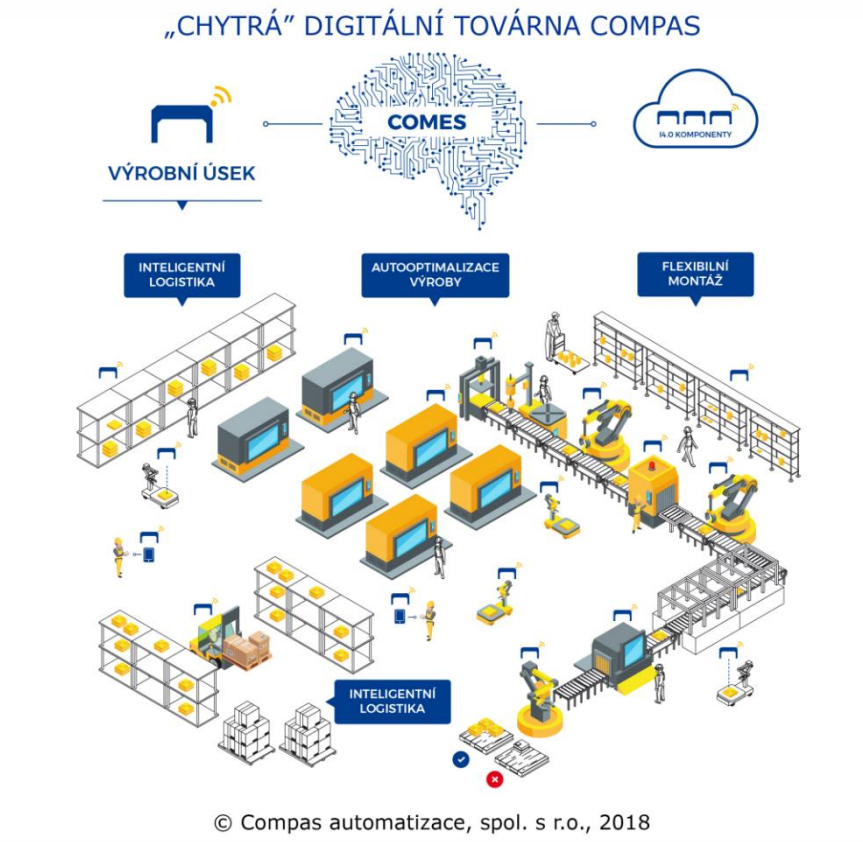
Smart Factory je koncept pro vyjádření konečného cíle digitalizace ve výrobě.

Jak se tento termín nejčastěji používá, Smart Factory je vysoce digitalizovaná dílna, která nepřetržitě shromažďuje a sdílí data prostřednictvím připojených strojů, zařízení a výrobních systémů (Obr. 1.13). Data pak mohou být použita samo optimalizačními zařízeními nebo v celé organizaci k proaktivnímu řešení problémů, zlepšování výrobních procesů a reakci na nové požadavky.

Díky různým technologiím, jako je AI, Big Data Analytics, Cloud Computing a Industrial IoT (Internet of Things), jsou chytré výrobní postupy plně komplexní.

Propojením fyzického a digitálního světa mohou chytré továrny sledovat celý výrobní proces, od výrobních nástrojů a dodavatelského řetězce až po jednotlivé operátory na dílně.

Když jsou chytré továrny plně realizovány, používají plně integrované, kolaborativní výrobní systémy, aby byly operace flexibilní, adaptabilní a optimalizovatelné. [5]



Obr. 1. 14 Koncept chytré továrny
Zdroj: [19].

1.4.9 Cloud computing

Cloud computing je použití systémů mimo pracoviště, které počítačům pomáhají ukládat, spravovat, zpracovávat a/nebo sdělovat informace. Tyto systémy mimo lokalitu jsou hostovány v cloudu (nebo internetu) namísto ve vašem počítači nebo jiném místním úložišti. Mohou zahrnovat cokoli od e-mailových serverů po softwarové programy, úložiště dat nebo dokonce zvýšení výpočetního výkonu vašeho počítače.

„Cloud“ je termín, který jednoduše znamená „internet“. Výpočetní technika zahrnuje infrastruktury a systémy, které umožňují počítači provozovat a sestavovat, nasazovat nebo interagovat s informacemi. V cloud computingu to znamená, že namísto hostování infrastruktury, systémů nebo aplikací na pevném disku nebo na místním serveru je hostujete na virtuálních/online serverech, které se k vašemu počítači připojují prostřednictvím zabezpečených sítí.

Cloud computing je použití hardwaru nebo softwaru mimo pracoviště, ke kterému se pro potřeby výpočetní techniky přistupuje přes sítě. Příklady cloud computingu závisí na typu poskytovaných služeb cloud computingu.

Mezi hlavní typy cloud computingu patří software jako služba, platforma jako služba a infrastruktura jako služba. Bezserverové výpočty, známé také jako funkce jako služba (FaaS), jsou také populární metodou cloud computingu pro podniky.

SaaS nebo Software jako služba. SaaS znamená, že místo instalace softwaru do počítače přistupujete k platformě online. Příkladem je Square, která zpracovává platby online, Google Apps, jako je Disk Google nebo Kalendář a Slack, který umožňuje spolupráci a chatování mezi ostatními uživateli.

IaaS nebo Infrastruktura jako služba. IaaS poskytuje komponenty infrastruktury, jako jsou servery, úložiště, sítě, zabezpečení a navíc cloud. Příkladem je Dropbox, systém pro ukládání a sdílení souborů, Microsoft Azure, který nabízí služby zálohování a obnovy po havárii, hosting, atd a Rackspace, který nabízí datové, bezpečnostní a infrastrukturní služby.

PaaS nebo Platforma jako služba. PaaS poskytuje výpočetní platformy, jako jsou operační systémy, prostředí pro provádění programovacích jazyků, databáze a webové servery. Příkladem je Google App Engine a Heroku, které umožňují vývojářům vyvíjet a poskytovat aplikace, bez serverové výpočty (také nazývané jednoduše „bezserverové“)

jednoduše používají server v cloudu. To nabízí větší pružnost, snadnější údržbu a je často cenově výhodnější než hosting serverů na místě. [5]

1.4.10 Kognitivní výpočetní technika

Kognitivní výpočty se týkají použití uvažování, zpracování jazyka strojového učení a lidských schopností, které pomáhají běžným počítačům lépe řešit problémy a analyzovat data. Tím, že se počítačový systém naučí vzorce a chování a stane se inteligentnějším, může řešit složité rozhodovací procesy.

1.4.11 Vlastnosti kognitivních počítačů

Technologická platforma kognitivních počítačů využívá strojové učení a rozpoznávání vzorů k přizpůsobení a pochopení informací, dokonce i nestrukturovaných informací, jako je přirozená řeč. K dosažení těchto výsledů využívá kognitivní výpočetní technika převážně své následující schopnosti.

První je bezesporu je schopnost adaptivního učení. Kognitivní systémy se musí přizpůsobit přílivu rychle se měnících informací a dat, které pomáhají plnit vyvíjející se sadu cílů. Platformy zpracovávají dynamická data v reálném čase a přizpůsobují se okolnímu prostředí a potřebám dat.

Další je jeho schopnost interakce člověka s počítačem (HCI) je nezbytnou součástí kognitivních strojů. Uživatelé interagují s kognitivními systémy a nastavují parametry, i když se tyto parametry mění. Technologie spolupracuje s dalšími zařízeními, procesory a cloudovými platformami.

Kognitivní výpočetní systémy identifikují problémy kladením otázek nebo stahováním doplňujících dat, pokud je pevný dotaz neúplný nebo vágní. Technologie to umožňuje ukládáním informací o souvisejících situacích a potenciálních scénářích.

Kognitivní výpočetní systémy musí identifikovat, porozumět a vydolovat kontextová data, jako je čas, syntaxe, doména, umístění, požadavky nebo úkoly, profil nebo cíle konkrétního uživatele. Mohou čerpat z více zdrojů informací, včetně sluchových, vizuálních nebo senzorových dat, stejně jako strukturovaných nebo nestrukturovaných dat. [5]

2 Analýza současného stavu v oblasti počítačové simulace v logistice

Počítačová simulace slouží počítači k reprezentaci dynamických odezev chování jednoho systému chováním jiného systému modelovaného podle něho (toho reálného). Simulace využívá matematický popis, nebo model reálného systému ve formě počítačového programu. Tento model se skládá z rovnic, které duplikují funkční vztahy v reálném systému. Když je program spuštěn, výsledná matematická dynamika tvoří analogii chování reálného systému, přičemž výsledky jsou prezentovány ve formě dat. Simulace může mít také podobu počítačové grafiky, která představuje dynamické procesy v animované sekvenci.

Počítačové simulace se používají ke studiu dynamického chování objektů nebo systémů v reakci na podmínky, které nelze snadno a bezpečně aplikovat v reálném životě. Například jaderný výbuch lze popsat matematickým modelem, který zahrnuje takové proměnné jako teplo, rychlost a radioaktivní emise. Další matematické rovnice pak mohou být použity k přizpůsobení modelu změnám určitých proměnných, jako je množství štěpitelného materiálu, který vytvořil výbuch. Simulace jsou zvláště užitečné, protože umožňují pozorovatelům měřit a předpovídat, jak může být fungování celého systému ovlivněno změnou jednotlivých komponent v tomto systému. [20]

Možnosti realizace počítačové simulace v logistice je možné dělit do dvou skupin. Jedná se o užití specializovaných simulační softwarů, nebo přímo použití všeobecných programovacích jazyků.

2.1 Pojmy spojené s programováním

2.1.1 Překladatel/překladač (program)

Překladač nebo jazykový překladový program je softwarová aplikace nebo služba, která překládá text nebo řeč z jednoho jazyka do druhého.

Překladače mohou také interpretovat programovací kód a převádět jej na instrukce, kterým počítač rozumí a které je možné provést. Například kompilátor je příkladem

překladače, který používá programovací jazyk (např. C++) a překládá do strojového jazyka nebo jazyka symbolických instrukcí , kterému počítač rozumí.

Tímto ulehčují tvoření programů a aplikací programátorům, pro které by bylo vytvoření programu přímo ve strojovém jazyku, nebo jazyku symbolických instrukcí neskutečně obtížné. [21]

V zásadě tedy překladače převádí určitá data z jedné formy či jazyka, do druhé tak aby mezi sebou mohli komunikovat účastníci, kteří rozumí jen určitému (nebo je to pro ně jen pohodlnější).

2.1.2 Kompilátor

Kompilátor je počítačový software, který překládá (kompiluje) zdrojový kód napsaný v jazyce na vysoké úrovni (např. C++, Java, PHP) do sady instrukcí ve strojovém jazyce, které jsou pro CPU digitálního počítače srozumitelné. Kompilátory jsou velmi rozsáhlé programy s kontrolou chyb a dalšími schopnostmi. Některé kompilátory překládají jazyk vysoké úrovně do středního assembleru, který je pak přeložen strojového kódu pomocí dalšího programu nebo přímo assembleru. Jiné kompilátory generují strojový jazyk přímo. Termín kompilátor zavedl americký počítačový vědec Grace Hopper, který na počátku 50. let navrhl jeden z prvních kompilátorů. [21]

2.1.3 Tlumočník

Interpret překládá kód do strojového kódu, instrukce po instrukci, to znamená, že CPU provádí každou instrukci předtím, než tlumočník přejde k překladu další instrukci. Interpretovaný kód zobrazí chybu, jakmile narazí na problém, takže je snazší užít kompilátor k ladění než ke konečné komplementaci kódu.

Tlumočník nevytváří nezávislou konečný ucelený zdrojový (celou jakoby sadu) zdrojového kódu, ale zdrojový kód se vytváří při každém spuštění. Interpretovaný kód se provádí pomaleji než zkompilovaný kód.

Mezi interpretované jazyky patří JavaScript, PHP a Python. Interpretované jazyky jsou také často označovány jako skriptovací jazyky. Tyto jsou ideální pro použití uvnitř dynamické webové aplikace. [21]

2.1.4 Assembler (Jazyk symbolických adres)

Assembler je program, který převádí jazyk symbolických instrukcí na strojový kód. Přebírá základní příkazy a operace z kódu sestavení a převádí je do binárního kódu, který dokáže rozpoznat konkrétní typ procesoru.

Assemblery jsou podobné kompilátorům v tom, že vytvářejí spustitelný kód. Assemblery jsou však jednodušší, protože převádějí pouze nízko úrovňový kód strojový kód. Vzhledem k tomu, že každý jazyk assembleru je navržen pro konkrétní procesor, sestavení programu se provádí pomocí jednoduchého mapování jedna ku jedné z kódu assembleru do strojového kódu. Na druhou stranu kompilátory musí převádět generický zdrojový kód vysoké úrovně na strojový kód pro konkrétní procesor.

Většina programů je napsána ve vyšších programovacích jazycích a jsou kompilovány přímo do strojového kódu pomocí kompilátoru. V některých případech však může být kód sestavy použit k přizpůsobení funkcí a zajištění, že budou fungovat specifickým způsobem. Proto IDE často obsahují assemblery, takže mohou vytvářet programy z jazyků vysoké i nízké úrovně. [21]

2.2 Obecné programovací jazyky

2.2.1 Python

Python je univerzální, interpretovaný programovací jazyk na vysoké úrovni. Je multiparadigmatický, takže podporuje více programovací paradigmat. Python umožňuje programátorům používat různé styly programování k vytváření jednoduchých nebo složitých programů, dosahovat rychlejších výsledků a psát kód téměř tak, jako by mluvili lidským jazykem. Některé z populárních systémů a aplikací, které během vývoje využívaly Python, zahrnují YouTube, Google Search, Google App Engine, BitTorrent, a jiné. Python je vysokoúrovňový, takže je vhodnější pro skušenější programátory, ale je velice populární díky čitelnému, stručnějšímu kódu a velkému množství balíčků což dovolují se vyhnout psaní kódu úplně od začátku, tím že využijete před chystanou část.

2.2.2 Java

Jedná se o celosvětově pravděpodobně nejrozšířenější objektově orientovaný programovací jazyk. Využívá spoustu dalších technologií například Spring framework

využívaný pro tvorbu webových aplikací. Java je multiparadigmatický, takže podporuje více programovací paradigmat. Java slouží k tvorbě, aplikací napříč platformy, díky tomu že se jedná o interpretovaný jazyk. Java je relativně lehce pochopitelný programovací jazyk, v porovnání s jinými jako je například Python. Kvůli své relativní snadnosti a rozšířenosti ve svět, je mu často dán přednost jako programovacímu jazyky vyučovanému na školách

2.3 Simulační softwary pro logistiku

2.3.1 Tecnomatix Plant Simulation

Tecnomatic obsahuje vícero propojitelných softwarových nástrojů pro různé oblasti výroby. Plant Simulation je modul Tecnomatix určený pro dynamickou simulaci. Tecnomatix je součásti produktové řady firmy Siemens PLM Software. Simulační modely vytváří pomocí bloků, které jsou rozdělené podle svých funkcí a účelů.

Obr. 2. 1 Ukázka simulace Tecnomatix Plant Simulation
[74]

2.3.2 Simul8

Velice rozšířený simulační software využívaný v mnoha odvětvích. Je považován za relativně snadno ovladatelný a nenáročný. Používá se například v automobilovém

průmyslu, v bankovnictví, ve veřejné správě, ve zdravotnictví a v dalších. Je postavený na bázi simulace diskrétních událostí pro modelování podnikových procesů.

2.4 Přístupy k vytvoření simulačního modelu

Následovné tři přístupy ukazují mou vlastní představu možných přístupů k vytvoření simulačního modelu pro konkrétního klienta. Jejich označení a základní myšlenky jsou lehce odvozené přístupů pro využívání simulace v oblasti logistiky. Jedná se o reprezentaci různých možných postupů, jak je vidím já a jak by podle mě mohli v reálném světě fungovat a jak jsou podle mě jednotlivé přístupy efektivní. Na závěr zmiňuje využívaný přístup přímo pro tuhle bakalářskou práci a simulační modely s ní spojené.

2.4.1 Klasický přístup

Tímto přístupem se vytváří většina simulačních modelů, zvláště když je v dnešní době vytvářet obecné simulační modely. Většinou firmy vytvoří balíčky a různé verze svých produktů a ty už hotové nabízejí firmám. Tyto simulační modely se postupem vyvíjejí, odstraňují se v nich chyba a přidává funkčnost. Zájem o specifické simulační modely šité na zakázku konkrétní firmě je stále menší a menší. Tento přístup napomáhá udržitelnosti týmu, co se o simulační modely stará a pomocí kontaktu může reagovat na vzniklé problémy a máte určitou jistotu, že tomu tak bude i v budoucnu, tedy pokud nedojde k zrušení společnosti, bez toho, aby předala práva a zodpovědnost jiné firmě. To ale není moc pravděpodobné. Když, je totiž produkt tak populární a kvalitní aby, jste si ho pořídily tak je pravděpodobné, že o něj budou mít zájem i jiní a firma má dobré vyhlídky do budoucnosti.

Klasický přístup je silně propojený i s originálními a specifickými simulačními modely vyrobené na zakázku pro určitou firmu. Je docela běžné, že se prostě jen stanový při setkání přesné detaily simulačního modelu, termíny, testy a cena a následně firma, které byl projekt zadán na simulačním modelu pracuje bez vnějšího zásahu. Po tom, co ten simulační model vytvoří, tak aby splňoval všechny podmínky a úspěšně prošel všemi stanovenými testy, je prezentován klientovy. Ten si ji projde a až tedy když objeví nějakou nesrovnalost, nebo chybu tak na ni reaguje. Smlouva vytvořená během zadání z, větší částí vymezuje přesný specifikace, takže klient, ani poskytovatel nesmí porušitel smluvené podmínky. Klient musí zaplatit za simulační model, který podle smlouvy

splňuje všechny stanovené body a zadavatel je musí dodržet, nebo klient může odstoupit. Samozřejmě, když se domluví na další změně k ochotě obou stran je to možné, ale pak je zvykem udělat buď novou smlouvu, nebo alespoň nějaké rozšíření, či dodatky.

Zadavatelé považují tento přístup za velice přitažlivý, protože nevyžaduje další práci a spolupráci na straně zadavatele, protože vše bylo zadáno a o vše ostatní se musí starat pouze dodavatel. Aby měl zadavatel možnost se k tomu takhle postavit je nutná určitá důvěra k dodavateli co má simulační model vytvořit. To je sice částečně ošetřeno díky smlouvě, ale i tak většina dodavatelů dává přednost dodavatelům s určitým reputací a historií. Protože odklady, problémy a nedodržování termínů, není jen výsadou stavebnictví. Spousta projektů co byly naplánováno bylo nutné odložit a termíny odevzdání byly posunuty, a věci spojené s informatikou nejsou výjimkou. Dobře to jde sledovat například na vývoji počítačových her, kde jejich uvedení na trh bývá často odloženo, nebo dokonce občas i úplně zrušeno.

Tento přístup se mi jako programátorovy líbí z důvodu toho, že nám poskytuje docela rozsáhlou volnost při tvorbě, ale může to komplikovat z důvodu, nutnosti upřesnění některých odborných specifikací ohledně oboru, pro který je tvořena. To může vyžadovat buď asistenci nějakého specialisty, nebo minimálně vyvinout snahu k získání dodatečných znalostí potřebných k dokončení simulačního modelu tak abychom si byli jistí, že funguje správně.

Mě osobně tento přístup velice nevyhovuje, zvláště pro dané simulační modely spojené s bakalářskou prací. Protože dávám přednost průběžnému upřesňování a konzultování možných přístupů. Samozřejmě některé specifické bakalářské projekty, zvláště které vyplívají přímo z práce ve firmě, nebo zadání přímo vymyšlené čistě přímo pro originální a specifické zadání, které má autor bakalářské práce přímo vylepšené. Tento přístup může také vyhovovat v případě, že buď autor bakalářské práce, či vedoucí nemají moc času na průběžné konzultace, a tak si pouze na začátku můžou vyhradit více času, aby specifikovaly zadání, jednotlivé etapy a body mnohem více dopodrobna, a zmenšily nutnost průběžných konzultací, které nemají k dispozici.

Závěrem bych tedy tuhle metodu doporučil, pro případ, kdy buď zadavatel nechce, nebo nemůže průběžně konzultovat možné řešení, ale dostatečně věří tvůrci simulačního modelu. Nebo tvůrci simulačního modelu, který rád vytváří projekt podle svojí představy, samozřejmě, ale se vším, co obsahuje zadání a dokáže si své řešení u zadavatele obhájit.

2.4.2 Modifikovaný přístup

Tento přístup se dá při vytváření chápat, tak že na vytváření simulačního modelu spolupracuje zadavatel s dodavatelem. Průběžně se konzultuje, zda cestou, kterou jsme se vydaly je správná a zda zadavatel je s tím spokojený a tou cestou, kterou jsme se vydaly souhlasí. Ve světě je tento přístup nepříliš populární, protože vyžaduje velice hodně času zadavatele a úzkou spolupráci s dodavatelem simulačního modelu. To obě strany velice vytěžuje a zpomaluje při práci. Jednoduše řečeno, když konzultují musí odložit co dělají a komunikovat mezi sebou, zatím co když nemusí spolu nic řešit tak obě strany mohou samostatně pracovat bez rušení a svým tempem. Většinou přístup, přesněji na konzultaci a spolupráci dojde částečně, až na konci, kdy je simulační model hotový a prezentuje se zadavateli. Během ní se občas dolaďuje a domlouvá se ještě dodatečné vylepšení. V některých případech je možné provést i drobné úpravy přímo na místě.

Modifikovaný přístup byl velice populární ve dřívějších dobách, kdy měli firmy vlastní programátory, kteří, vytvářely pro firmu originální simulační modely a samotní je udržovaly. To umožňovalo snadnou, plynulou a neustálou komunikaci, spolupráci a zpětnou vazbu od uživatelů simulačního modelu. Díky tomu může tvůrce simulačního modelu testovat a konzultovat jednotlivé části, a ne přímo celý simulační model. Například začíná s první tou nejvíce jednoduchou funkcí, nebo vzhledem a pak tam přidává další funkčnosti a rozšiřuje ji o další funkce. Například napřed vytvoří základní vzhled, kam se budou vkládat potřebná dat. Ať bude uživatelům vyhovovat. Vybereme jednu z funkcí, kterou bude simulační modelu se zadanými informacemi provést. Když to funguje pro standartní případy, tak například dále ošetří možné výjimky a chyby co by se mohly objevit. Když tohle všechno funguje, přidáme další funkčnost a opakujeme postup neustále dokola. Díky průběžné konzultaci neustále víme, o co je zájem a co je potřeba upravit či přidat. Samozřejmě, i zde je nutná určitá míra spolupráce. Nemůžeme nestále dělat změny v simulačním modelu, kvůli každé drobnost. Proto je dobré vždy, než něco předěláme nebo odstraníme nechat uživatele si to napřed pořádně prozkoušet a zvyknout si na to. Až je jisté že je to skutečně nutné předělat pak to tedy udělat. Výhodou tohoto přístupu je rychlá reakce na krize, z důvodu dostupnosti programátora a jeho znalosti kódu, protože ho sám vytvořil. Nevýhodou je nutnost starání se o dalšího pracovníka a nebezpečí spojené s jeho odchodem. Když odejde nejen, že je akutně nutné sehnat kvalifikovanou náhradu, ale daná náhrada musí být schopná se vžít s daným řešením a dále s ním pracovat. Tohle může být obtížné, protože každý dává předost vlastní

tvorbě, protože každý programuje rozdílně a každý kód má vlastní specifikace podle jeho autora, a to se projevuje čím dál více čím je simulační model složitější a její výsledný kód rozsáhlejší.

Popularita modifikovaného přístupu u zadavatelů, je velice rozdílná, a to podle druhu zadavatele a jeho vztah k tvůrci simulačního modelu. Protože vyžaduje častou spolupráci mezi zadavatelem, či vedoucím a programátorem, či dodavatelem musí si rozumět a umět se efektivně domluvit. Konec konců kvalita spolupráce nezáleží jen na schopnostech zúčastněných, ale i na jejich osobnosti a tomu, jak si mezi sebou rozumí. To může silně ovlivnit, zda spolupráce mezi nimi je příjemná a plynulá, nebo otravná a problematická. Navíc je důležité, aby si vedoucí byl jistý, že mu daný pracovník zůstane a v případě odchodu počká, než si seženou náhradu a následně ji zaučí a předají simulační model ve formě, která je pro nového pracovníka přehledná.

Pro programátory je tento přístup rozdílně populární. Stejně jako pro zadavatele u pro něj je velice důležité, s kým spolupracuje. Možná ještě více než v předchozím případě, a to z důvodu možné nerozhodnosti. K efektivní tvorbě programu je nutná určitá plynulost a jednotnost zadání. Neustále změny škodí celistvosti a výsledné struktuře programu, a kromě časové ztrátu mohou do budoucnosti způsobit rozsáhlé problémy z důvodu dodatečných chyb a nedokonalého propojení vytvořeného z neustálého přepisování kódu. Toto řešení je často také spojeno s tím, jakým druhem programátora chce dodavatel být. Protože programátor úplně nejen sám vytvoří úplně celý program, ale i sám se o něj neustále stará a rozvíjí ho. Oproti práce pro programátorskou firmu, kde pracujete v týmu a na rozdílných projektech.

Pro mě je tento přístup docela přijatelný, protože rád konzultuju s někým jiným a ubezpečuji se, zda to dělám správně a jak to zadavatel chce. Navíc je to pravděpodobně nejlepší řešení pro bakalářskou práci postavené na pravidelné konzultaci s vedoucím projektu a kontrola, zda postupujeme správně. Další výhodou je že to nutí programátora pracovat v kuse a neodkládat práci na později, z toho důvodu, že na každou konzultaci musíte mít nachystanou další část. Samozřejmě, plně využití tohohle přístupu vyžaduje, aby měli jak vedoucí, tak autor bakalářské práce průběžně dostatek času na konzultace.

Závěrem by se dalo říci, že modifikovaná metoda je perfektní, když zadavatel chce mít řešení přesně podle svojí představy, rád si průběžné kontroluje, jak vývoj postupuje a má kontrolu na dalším vývojem simulačního modelu, pro tohle všechno musí mít samozřejmě

schopného strůjce modelu. Pro strůjce modelu je vhodné, když mu nevadí (nebo preferuje) průběžnou kontrolu, nevadí mu komunikace s zadavatelem a případně plánuje budoucí další vývin a údržbu simulačního modelu.

2.4.3 Progresivní přístup

Progresivní přístup by fungoval tak, že další dodatečné změny v simulačním modelu provádí přímo pracovník ve firmě a na tuhle variaci jej připraví dodavatel simulačního modelu. Na tvorbě simulačního modelu se může pracovník klienta přímo i podílet, ale tak jako tak hlavní práci by měla provést externí firma a účelem tohoto přístupu je hlavně to, aby dokončení simulačního modelu byl vlastní pracovník klienta schopen provádět v daném simulačního modelu změny, či údržbu, a to bez nutnosti další konzultace s dodavatelem. To vyžaduje často školení, konzultace, nebo samotnou spolupráci na vývoji simulačního modelu mezi dodavatel a správcem modelu klienta. Aby tento postup fungoval je nutný, aby firma zaměstnávala vlastního programátora, IT specialistu, nebo jiného zaměstnance schopného vyznat se v psaní a úpravách kódu v jazyce ve kterém je daný simulační model napsán, je ochotný a schopný převzít výsledný simulační model. Dodavatel, zase musí být schopný nejen vytvořit simulační model, ale i jej efektivně prezentovat a objasnit jeho stavbu a vnitřní fungování, tak aby jej programátor klienta efektivně zvládl.

Takto chápaný progresivní přístup není příliš populární a využívaný u zadavatelů. Důvody jsou především takové, že aby tato metoda efektivně fungovala musí se splnit mnoho podmínek, a některé nejsou příliš populární. Například, když už máte ve firmě vlastního programátora schopného vyznat se v kódu, je často lepší ho nechat vytvořit si vlastní simulační model a napsat si tedy pro něj vlastní zdrojový kód ve kterém se díky tomu nadále dokáže velice dobře vyznat. Dále to vyžaduje nalezení dodavatele, který nejen dokáže vytvořit daný simulační model, ale i jej efektivně předat dalšímu programátorovi, a to tak aby ji plně pochopil a mohl na ní stavět. Proto, je tento přístup užitečný jen pro velice specifické případy. Například když potřebujete daný simulační model ve velmi krátkém časovém úseku, kdy by to vás pracovník sám nedokázal, ale chcete, aby nadále to byl je váš simulační model a věříte a chcete, aby ji pak nadále udržoval a rozšiřoval váš vlastní zaměstnanec.

Pro dodavatele je ještě méně populární, protože vyžaduje nejen vše, co vyžadoval klasický přístup, tedy vytvoření simulační model v termínu podle všech specifikací, ale i

zaškolení správce, který by se v ní měl plně vyznat a sám ji případně dále upravovat a rozšiřovat. Tahle práce navíc zvláště, když vyžaduje určité dodatečné komunikativní a pedagogické schopnosti může tvořit obrovskou překážku a problém, který je málo kdo ochotný přijmout. Samozřejmě, vyskytují se specifické případy, kdy i tento přístup může být žádaný a dodavatelů mohou být ochotni se k němu uchýlit. Očividnou motivací je finanční kompenzaci, s vidinou dostatečně velkého finančního ohodnocení je spousta dodavatelů a programátorů ochotná překousnout mnoho komplikací a postavit se vážným problémům. Dalším případem může být dělení práce v rozsáhlé firmě pro různá oddělení. Například, firma může mít hlavní sídlo, která má vlastní plně kvalifikovaný tým programátorů, ale má i pobočku na jiném místě, která má pouze jednoho IT pracovníka. Když ta pobočka potřebuji specifický simulační model, na kterou v rámci časového rozmezí jejich pracovník nestačí můžou požádat centrálu o zaúkolování jejich vlastního schopného IT týmu. Ten vytvoří simulační model a následně zaučí daného pracovníka, aby se o ni dál sám staral. Ten má ještě výhody, že když se objeví vážný problém, má teoreticky ještě šanci ho probrat s týmem z centrály.

Mě přijde tento přístup téměř nerealistický. Sice chápu potřebu umět předat svoji práci dál a umět si ji obhájit. Přijde mi tato cesta velice obtížná a pro mě nepříjemná, a to ať už jak na straně klienta, tak dodavatele. Jako klient by pro mě bylo obtížné přijmout komplexní cizí kód za vlastní a dokázat s výsledným kódem dále pracovat a dále na něm provádět úpravy a rozvíjet kód neustále. Dále Mnohem lépe a pohodlněji by se mi pracovalo se zdrojovým kódem, který jsem vytvořil od začátku, chápu ho díky tomu perfektně do hloubky a mám k němu i tedy i lepší vztah. Samozřejmě s ohledem na různé možné scénáře nemůžu vyloučit, že jednou nebudu mít na výběr, ale ta představa mi není moc příjemná. Minimálně bych rád už podílel na jeho vývinu jako člen týmu, než abych k němu přišel z ničeho nic. Jako dodavatel bych zase měl problémy vysvětlit jinému programátorovi všechny svoje myšlenky a postoje. U drobných projektů to docela jde, ale u rozsáhlý, kde je spousta možností kudy se vydat, jak jednotlivé části rozdělit, pojmenovat a využívat je to pak stále obtížnější. Neexistují lidé, co se dokonale shodnou, a tak nikdy váš kód nebude takový, aby někomu vyhovoval bez nejmenších připomínek. Navíc to silně závisí, i na povaze, myšlení a ochotě toho komu ho mám předat. Lidský faktor je často ten nejvíce problematický.

Závěrem bych chtěl označit progresivní přístup za velice málo používaný a často i nepohodlný. Je to často přístup, který se využívá, když prostě není jiná možnost, jak jsem

zmínil na příkladech. Určitá forma progresivního přístupu by se dala asi brát i střídání IT specialisty ve firmě, kde, když například předchozí zaměstnanec odchází do důchodu musí zaučit svého nástupce. Ale i tato nutnost je na ústupu. Stále více práce na simulačních modelech, ať už jejich zavedení, tak i výsledná údržba se v dnešní době předává externím firmám.

2.4.4 Přístup pro bakalářskou práci

Pro vybrání správného přístupu k bakalářské práci bylo velice důležité především zjistit stanovisko vedoucího, a to pana Fedorko. Kolik bude mít času, jak podrobně chce být informován o výsledné struktuře zdrojových kódů simulačního modelu a jak velkou volnost poskytne k vypracování bakalářské práce podle mých vlastních nápadů. Naštěstí byl pan Fedorko velice ochotný a chápající, proto s ním nebyl v žádném bodě výrazný problém.

Napřed bezesporu vyřadíme možnost progresivního přístupu, protože s výsledným kódem pracuji pouze já a neplánuji jeho další vývoj a spolupráci na něm s jiným programátorem. Samozřejmě během obhajoby bakalářské práce bude nutné objasnit fungování daného simulačního modelu a objasnit všechny části, které by mohli komisi zajímat, ale přesto neočekávám že by někdo ze zmíněné komise plánoval pokračovat v mém simulačním modelu.

Druhé dva přístupy, ale jsou mnohem vhodnější pro moji bakalářskou. Práci, především modifikovaný přístup. Díky ochotě vedoucího práce se pravidelně setkávat, aspoň jednou týdně, online přes TeamSpeak, jsme mohli pravidelně probírat vývin, nápady a případné změny v simulačním modelu k bakalářské práci a následně i k textové části. Díky tomu jsme si mohli ujasnit, že se shodneme a postup na práci mohl efektivně postupovat.

Také klasický přístup byl částečně použit při tvorbě bakalářské práce, a to zde hlavně ze dvou důležitých důvodů. Prvním je, že vedoucí pan Fedorko mi dal k vypracování bakalářské práce velkou volnost, a tedy mezi jednotlivými setkáními jsem mohl pracovně, efektivně sám a podle sebe, protože setkání spíš sloužily k ujasnění, že jsem zadání dobře pochopil a volby, cesty a metody co jsem si vybral jsou ne, jen pro vedoucího přijatelné, ale pro práci i vhodné.

Druhým důvodem, bylo že vedoucí pan Fedorko nemá takové zkušenosti s psaním simulačních modelů v Javě, a tedy mi dal pro vybrané řešení, velkou volnost a zajímal se

tedy až o případné řešení. Navíc většinou mu šlo především spíše o funkčnost a ověření správnosti, a tedy konkrétní řešení nechával z větší části na mě.

Věřím, že kombinace těchto dvou přístupů byla efektivní a dovolila pracovat dostatečně rychle, efektivně a zároveň být průběžně schopný ujistit se s vedoucím bakalářské práce panem Fedorko, takže jsem si byl jistý, že jdu správnou cestou. Je sice možné, že kdybych se vydal klasickou cestou a neměl tolik konzultací bylo by vypracování rychlejší, ale zase bych si nebyl tak jistý, což je pro mě, velice důležité.

3 Návrh zefektivnění tvorby simulačních modelů

V této kapitole bude ukázáno možné zefektivnění tvorby simulačních modelů pomocí aplikací vytvořených pomocí obecného programovacího jazyku Java a demonstrované na čtyřech typových příkladech. Ještě předtím, ale něco k tomu, co vše bylo pro práci na těchto příkladech využito a v jaké formě je prezentovaný výsledek.

Základem celé aplikace je Java, a je nutná jak k jejímu vytvoření, tak k samotnému spuštění. Naštěstí je v dnešní době Java prakticky v každém počítači a sama, se i aktualizuje. Přesto, pokud není k dispozici, z jakéhokoliv důvodu dostatečná verze může to způsobovat problémy se spuštění.

Naštěstí, její aktualizace, či dokonce instalace je velice jednoduchá, a dokonce je i zdarma. Navíc Javu využívá spousta aplikací a webových stránek, proto je tahle situace krajně nepravděpodobná. Zvláště když se vezme v úvahu, že se jedná o aplikaci vytvořenou pomocí nástrojům dostupných zdarma na internetu, a ne velkou firmou, či profesionálním týmem nevyžaduje, poslední verzi Javy ke svému používání.

Aplikace byla vytvořena ve vývojovém prostředí Eclipse IDE for Java Developers, která se dá stáhnou z jejich domovských stránek na https://www.eclipse.org/ kde poskytují jak verzi zdarma využitou pro vytvoření aplikace, tak placenou profesionálnější verzi.

Pro grafickou stránku aplikací (a to přesněji 2. - 4.), využíváme aplikační platformu JavaFX. Ta se načte a dá využít tak, že se knihovny pro ni načtou přímo do projektu pro konkrétní aplikaci. Knihovny se dají stáhnout ze stránky https://gluonhq.com/products/javafx/ kde si můžete vybrat verzi podle vaší platformy.

Všechny zdrojové kódy aplikací pro jednotlivé typové příklady se nachází na přiloženém CD ve složce Zdrojové kódy, dále se na CD vyskytují i komprimované JAR soubor daných aplikací, které se dají přímo spustit pomocí jejich BAT souborů ve složce Spustitelné aplikace. Dále se dají vidět důležité části zadání v přílohách A, B a C a zdrojové kódy v přílohách D, E, F a G.

3.1 První typový příklad

3.1.1 Zadání

První aplikace demonstruje především možnost vygenerovat kód pro tacnomatix a to díky konzolové aplikaci. Tato aplikace by měla uživatele provést zadáváním jednotlivých potřebných údajů a následně je využít pro vygenerování kódu. Navíc by měla využít výhody konzolové aplikace oproti grafickému řešení. Což je vytvoření velkého počtu proměnných. Tenhle kód nemá žádnou předlohu, z důvodu čistého experimentování a zda je vůbec možné tento projekt uskutečnit.

3.1.2 Algoritmizace

Celá tahle aplikace je postavena na cyklu. Na začátku se stanový počet vláčků a entit. Následně se aplikace zeptá na každý jednotlivý název, který musí uživatel zadat a na konec kolik entit, ještě kolik entit se stejným názvem se budeme potřebovat. Výhodou je tedy, že nejsme limitování pro počet entit prostorem, ale nevýhodou je menší přehlednost, protože je neustále nevidíme.

3.1.3 Vývojový diagram

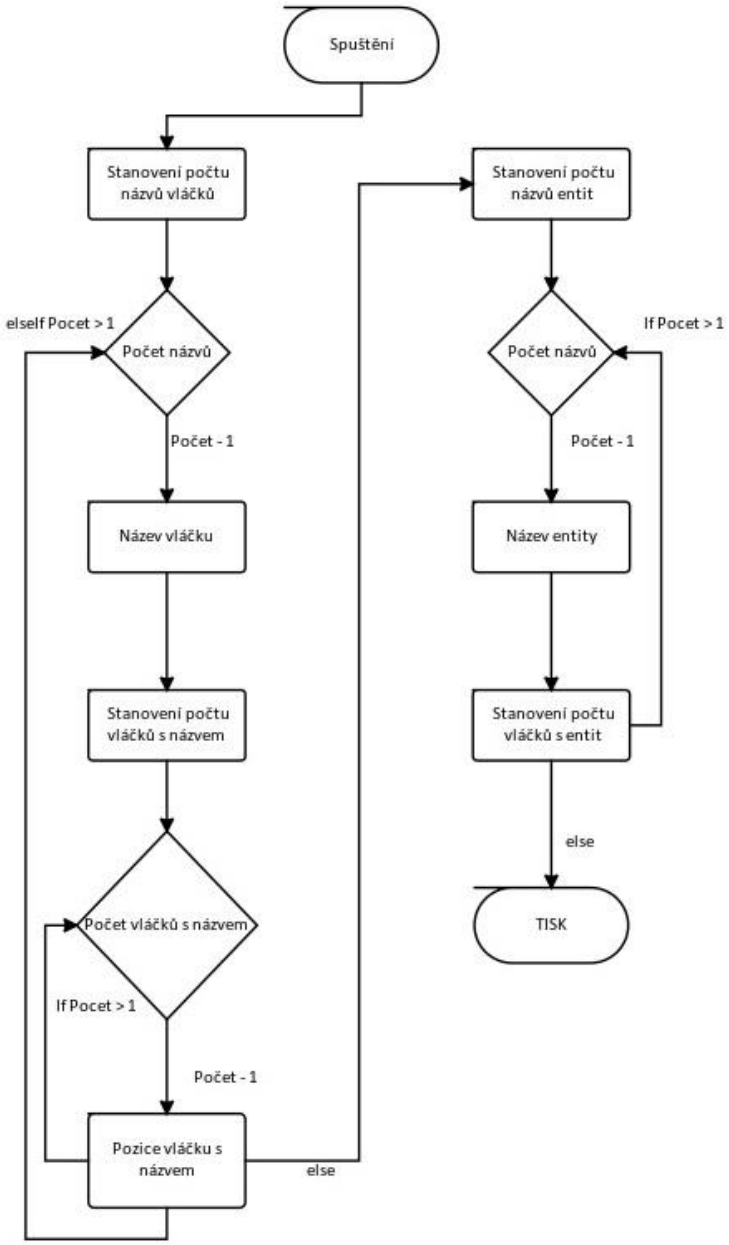


Diagram 3. 1 Vývojový diagram pro 1. typový příklad

3.1.4 Zajímavosti

První aplikace slouží tedy především pro ověření, zda je tahle forma generování kódu pro tacnomatix možná a praktická.

Největší specifikací je asi vytvoření textových souborů hned na začátku. A to přesněji z důvodu pro možné průběžné průběžného zapisování.

Takže textové soubory jsou během celého běhu programu otevřeny pro zápis a uzavřeny až úplně na konci. Takže průběžně jak získáváme všechny informace jako název entity a jejich počet tak je využijeme a zapíšeme do správného souboru.

Až skončí všechny cykly vytvořený zadaným počtem vláčků a entit, až tedy se výsledné textové soubory uzavřou.

3.1.5 Shrnutí

První příklad demonstruje možnosti tvorby kódu pro tacnomatix jinou cestou a zároveň demonstruje výhodu konzolové aplikace pro zadání většího počtu. Bohužel, musíme si uvědomit, že příjemné prostředí je v dnešní době nutnost, a proto uživatelé vyžadují často ne, jen plně funkční aplikaci, ale i takovou, která se jim bude graficky líbit a bude vysoce přehledná. Což konzolová aplikace plně nesplňuje a v dalších příkladech tedy přidáme i grafické rozhraní.

3.2 Druhý typový příklad

3.2.1 Zadání

Cílem dané aplikace je vytvořit kód pro program tacnomatix alternativní cestou. V tomhle případě pomocí Java aplikace s *JavaFX* frameworkem. Výsledný kód pro tři různé části programu se rozdělí mezi tři různé textové soubory. Všechny vzorové kódy, co by měla být schopná aplikace vygenerovat, lze vidět na Příloze A. V tomto případě bude zadávání potřebných proměnných (hodnot), jako jsou názvy entit, výsledných textových souborů a potřebné číselné hodnoty pomocí graficky vytvořeného formuláře.

3.2.2 Algoritmizace

Pro tuhle aplikaci bylo rozhodnuto, že se bude ovládat pomocí grafického formuláře vytvořené pomocí frameworku *JavaFX*. Všechny potřebné informace jako názvy entity, údaje a názvy výstupných textových souborů se stanovují pomocí rámců ve vytvořeném grafickém zobrazení. Po tom, co se vyplní vše potřebné se pak vše vytiskne do textových souborů pomocí tlačítka na jedno kliknutí.

3.2.3 Vývojový diagram

Diagram 3. 2 Vývojový diagram pro 2. typový příklad

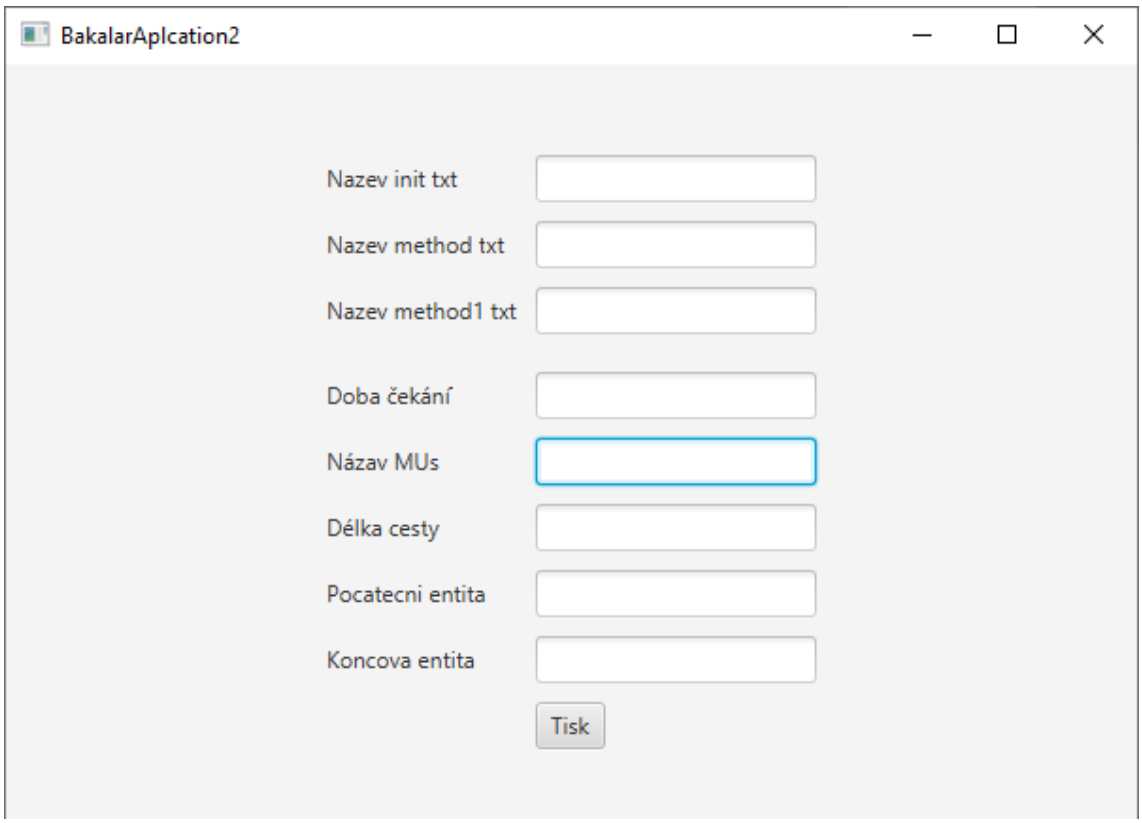
3.2.4 Zajímavosti

Pro tuhle aplikaci je asi nejzajímavější, že jsme poprvé vytvořily základní layout pro okno aplikace kde se všechny potřebné objekty vyskytují. Následně jsou tu poprvé vytvořené rámce *TextField* pro zadávání informací, které pak načítáme a na závěr tlačítko pro příkaz vytisknutí textových souborů pomocí informací obsažených v rámcích *TextField*.

3.2.5 Shrnutí

Výsledná aplikace splňuje svůj účel a dává tak možnost, alternativního vytvoření potřebného kódu to a díky možnosti vyplnění grafického formuláře. Tahle metoda vypadá

atraktivní z důvodu, že s formuláři má zkušenost většina pracovníku ať už se jedná o například různé formuláře, například daňové přiznání, nebo vyplňování dotazníku.



Obr. 3. 1 Ukázka druhého typového příkladu

3.3 Třetí typový příklad

3.3.1 Zadání

Tahle aplikace sloužila k demonstraci zlepšení a posunutí od předchozího návrhu, příkladu. A to tak aby uměla vše, co dokázala předchozí a nejlépe měla ještě nějaké dodatečné funkce. Tento program kvůli změně používá jiný vzorový kód pro tacnomatix, pro změnu a pro příklad na kterém se lépe demonstruje možnost zvolení různého počtu entit. Výsledný kód pro tři různé části programu se rozdělí mezi tři různé textové soubory. Všechny vzorové kódy, co by měla být schopná aplikace vygenerovat, lze vidět na Příloze B. Tento příklad vyžaduje tedy trochu flexibilnější zadávání, většina rámců je stále povinná, ale nutné rámce pro 1.Vozidlo až 5.Vozidlo stanový uživatel z číselníku Počet vozidel. Číselníkem v tomhle případě budeme zadávat i dobu čekání.

3.3.2 Algoritmizace

Pro tuhle aplikaci bylo rozhodnuto, že se bude ovládat pomocí grafického formuláře vytvořené pomocí frameworku *JavaFX*. Všechny potřebné informace jako názvy entity, údaje a názvy výstupných textových souborů se stanovují pomocí rámců v, vytvořeném grafickém zobrazení. Po tom, co se vyplní vše potřebné se pak vše vytiskne do textových souborů pomocí tlačítka na jedno kliknutí. Rozdílem bude oproti předchozímu programu, že přidáme některé funkce a program nebude mít přesně stanovený počet hodnot, ale pro parametr počet vozidel, a to pomocí třídy *ComboBox*. Všechny *TextField* které nebudeme potřebovat se následně díky získané informaci zablokují, či odblokují pomocí metody *setEditable(True/False).* Na závěr se vše podle stanovených hodnot vytiskne pomocí tlačítka Tisk.

3.3.3 Vývojový diagram

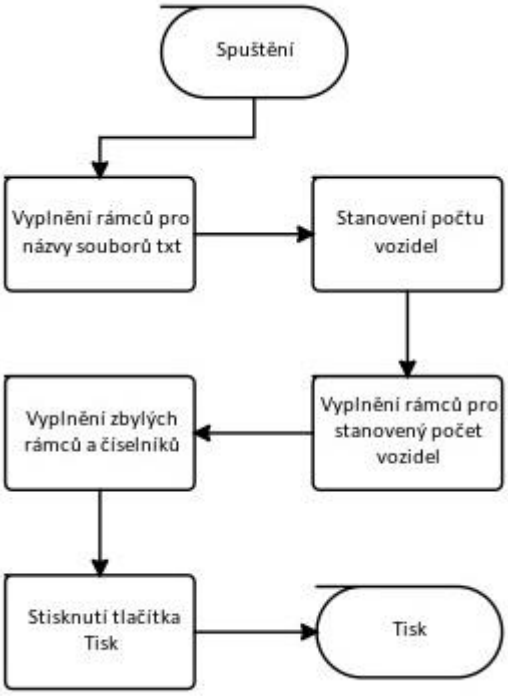


Diagram 3. 3 Vývojový diagram pro 3. typový příklad

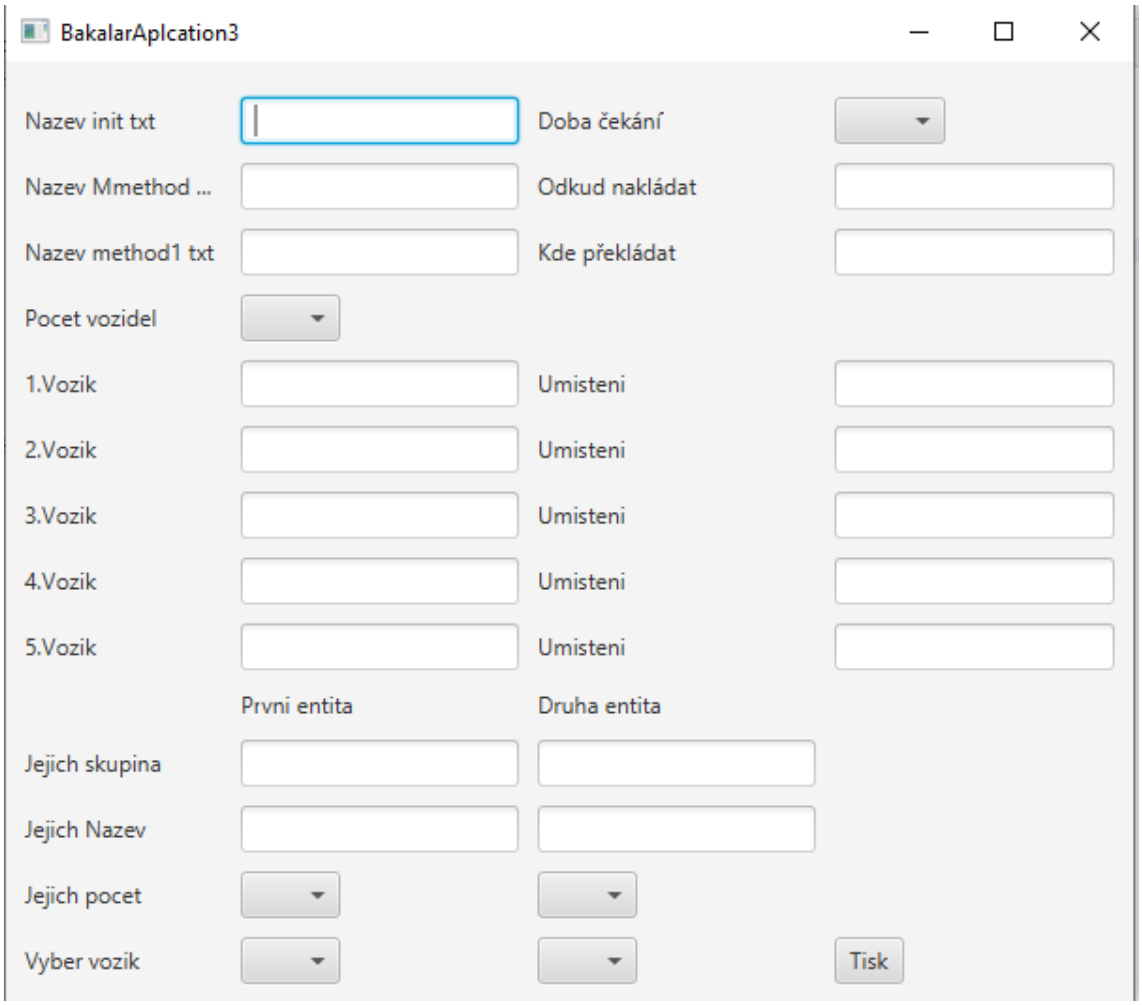
3.3.4 Zajímavosti

Tahle aplikace je v podstatě zlepšení té předchozí, jen s rozdílným zadáním. Obsahuje všechny funkce jako ta předchozí, a navíc má navíc několik vylepšení. Dovoluje vybrání konkrétní hodnoty skrze *ComboBox*, která slouží k vybrání počtu vozíků, či vozidel co budeme potřebovat.

Na nic navazuje další schopnost aplikace, a to je odblokování rámců pro počet vozidel co jsme vybraly a případné zablokování těch, co nebudeme potřebovat. Tomu slouží metoda *setEditable(True)* pro odblokování a *setEditable(False).* Pro vybrání správné kombinace se používá větvení pomocí switch. Na závěr po vyplenění všech potřebných rámců aplikace umožňuje vytisknutí výsledného kódu do textových souborů pomocí tlačítka Tisk.

3.3.5 Shrnutí

Účelem aplikace bylo vylepšení a přidání všech funkcí co měla předchozí aplikace a rozvinutí možností jejich funkcí, a to především díky zvolení hodnoty ze seznamu možných a blokování nepotřebných rámců. Na tom se krásně demonstruje neustále možnosti zlepšování a přidávání možných dodatečných funkcí.



Obr. 3. 2 Ukázka třetího typového příkladu

3.4 Čtvrtý typový příklad

3.4.1 Zadání

Účelem aplikace bylo vylepšení a přidání všech funkcí co měla předchozí aplikace a rozvinutí možností jejich funkcí, a to především díky zvolení hodnoty ze seznamu možných a blokování nepotřebných rámců. K tomuto bylo vybráno zadání, jehož všechny vzorové kódy, co by měla být schopná aplikace vygenerovat, při správné kombinaci zvolených hodnot se nachází v Příloze C. Na tom to zadání se krásně demonstruje neustále možnosti zlepšování a přidávání možných dodatečných funkcí. Dalším rozdílem je že vzorový kód pro *.Met_Sensor_Creation* neobsahuje žádný dodatečný text a proto si jediný co si bere z nastavených parametrů je název textového souboru obsahují výsledný kód.

Stejný případ je i pro *.Met_Sensor_Creation.*

Trošku více to ovlivňuje *.Met_Sensor_Identification* ze které si to bere dobu *Wait*, nebo jinak dobu čekání.

Nejvíce to ale ovlivňuje *.Met_Color_Setting.* Tam se nejen nastavuje název materiálu, a jednotlivé hodnoty pro RGB. Ale i samotný počet materiálů které chceme, a to od jednoho po sedm.

3.4.2 Algoritmizace

Aplikace na rozdíl od předchozí zobrazí a skryje rámce pro nepotřebné hodnoty zadané pomocné číselníku Počet materiálů. Také kvůli snadnější kontrole byly všem číselníkům nastavená defaultní hodnota.

Právě kontroly jsou největším vylepšením této verze. První je, už jen to že rámce pro RGB nepřijímají nečíselné znaky. Další kontroly jsou to, že jsme vybraly v číselníku aspoň jeden materiál, že jsme vyplnily před stisknutím tlačítka Tisk všechny rámce a že hodnoty pro barvy nejsou stanoveny na větší, než 255.

Po stisknutím Tisk může dojít, buď k vyvolání chybového hlášení s výpisem problémů, nebo přímo vytvořením očekávaných textových souborů.

3.4.3 Vývojový diagram

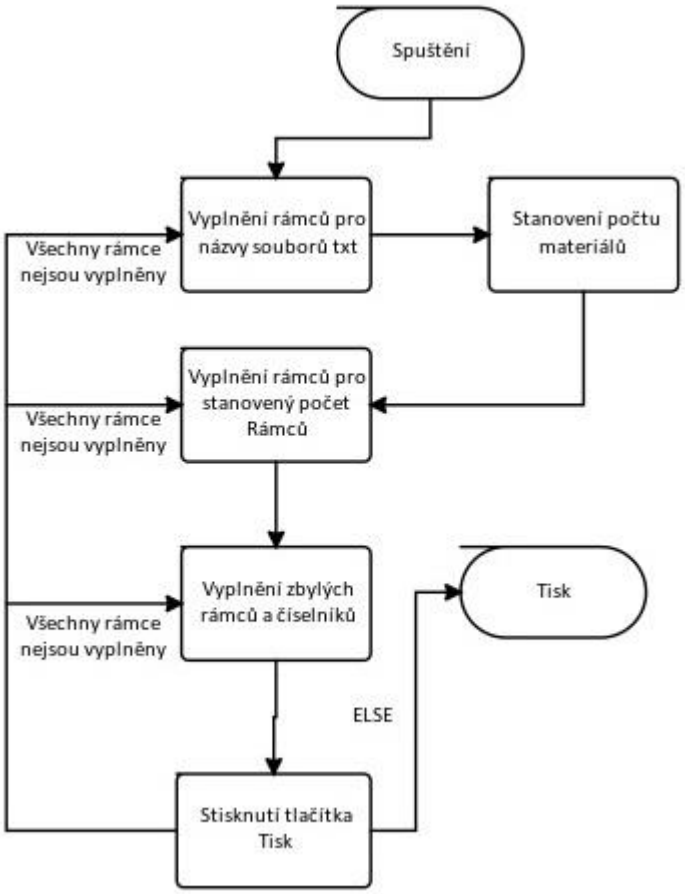


Diagram 3. 4 Vývojový diagram pro 4. typový příklad

3.4.4 Zajímavosti

Jak bylo řečeno tahle aplikace má být nejvíce komplexní. Proto mají jednotlivé *ComboBox* mají nastavené počáteční hodnoty, pro snadnější kontrolu a ovládání.

Dále aplikace místo blokování nepotřebných rámců, je přímo skrývá pomocí metody *setVisible(false)* a následně odkryje potřebné pomocí metody *setVisible(false).* Zobrazování a skrývání rámců je opět řešeno pomocí větvení switch.

Dalším vylepšením je, že do rámců, které potřebují pouze číselné hodnoty pro RGB, nelze psát jiné než číselné. Znaky pomocí specifického algoritmu.

Další možnou chybou je, kontrolu zda jsou všechny rámce vyplněny pomocí porovnávání rámců s hodnotou null a *trim().isEmpty(),* navíc pro jednotlivé barvy ještě dochází k porovnávání zda hodnota není větší než 255.

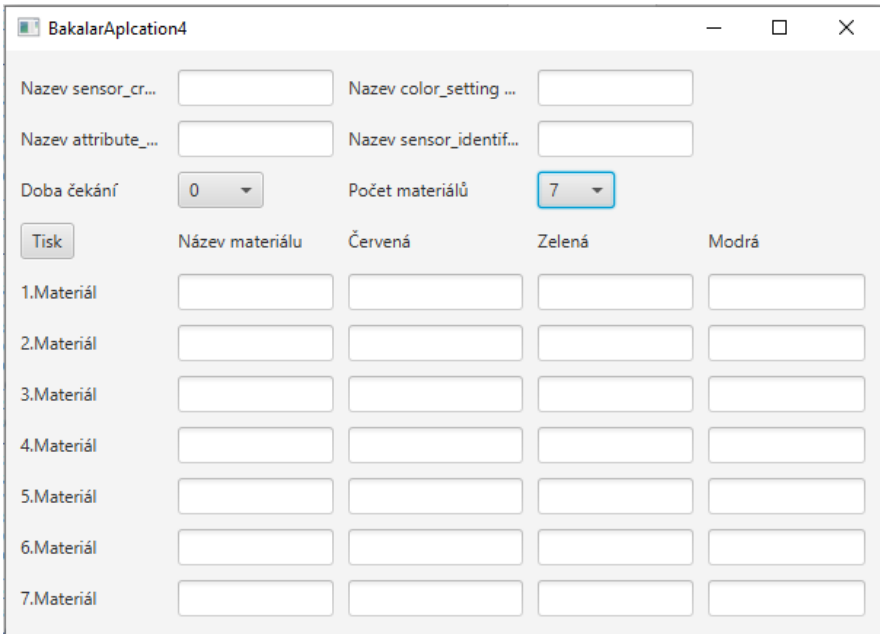
Všechny tyhle chyby se ukládají do specifického stringu (řetězce znaků) a pak nakonec je porovnán jeho obsah, zda je větší, než byl před začátkem kontrol a pokud ano tak se vyvolá chybové hlášení. Co vyvolá chybu.

Protože bylo do tlačítka pro Tisk (v kódu *btn*) obtížné zapojit jak kontrolu a vytváření seznamu chyb, tak vyvolávání chybového hlášení. Tak aplikace obsahuje neviditelné tlačítko *btn2*, které se na konci kontroly v případě že se objevily chyby zavolají pomocí příkazu *btn2.fire.*

Pak dochází k vyvolání výstrahy, které obsahuje i výpis všech problémů. Jinak dochází k vytvoření a nových textových souborů a jejich naplnění kódem, stejně jak tomu bylo v předchozích příkladech.

3.4.5 Shrnutí

Aplikace splňuje, co bylo zadáno a předepsáno. Většina možných problémů a chyb je vyřešena a další možné vylepšení a rozšíření by se muselo pořádně prokonzultovat a testovat. Největší rozšíření bych asi v budoucnu mohl být propojení různých typů cílových příkladů jakých jsme viděly v předchozích zadání do jednoho ať si můžeme vybrat ten, které chceme nyní zpracovat, a přesto využívat všech výhod a kontrol poslední aplikace. Úplně nové a dodatečné funkce by mohli být přidány, pouze za předpokladu, že nenaruší současnou funkčnost a bude s ním zadavatel souhlasit.



Obr. 3. 3 Ukázka čtvrtého typového příkladu

4 Zhodnocení návrhů

Na jednotlivých typových příkladech jsou názorně demonstrovány možné cesty rozšíření a zlepšení možných cest tvorby simulačních modelů, přesněji jeho kódu. Zefektivnění je tady především bráno z pohledu uživatele. Aby mi práce a tvorba simulačního modelu lépe vyhovovala a nevyžadovalo tak velké technické schopnosti a znalosti.

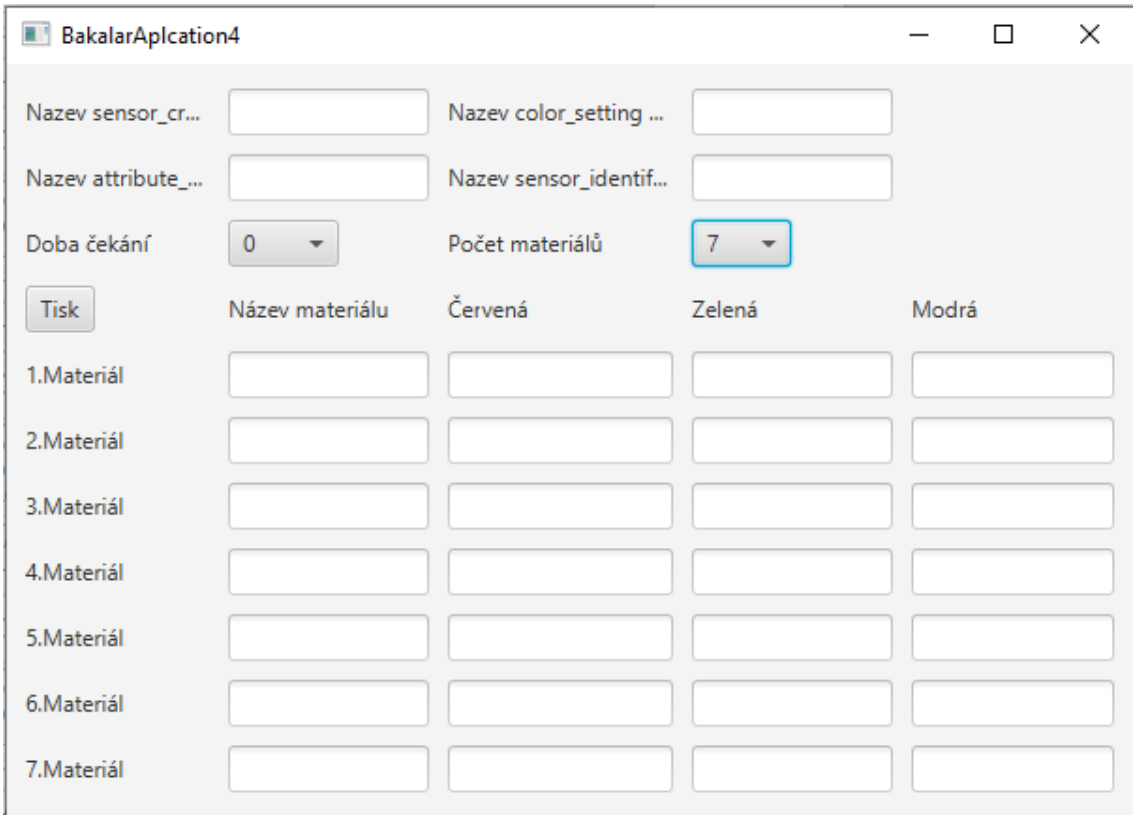
První příklad jen demonstruje možnost vytvořit výsledný kódu simulačního modelu, přes co nejsnadnější programátorskou cestou, tedy přes konzoli. Je to určité zlepšení, protože uživatele přímo vede a žádá od něj vždy co právě potřebuje. Bohužel tohle řešení není kvůli absenci vizualizace a nepříliš příjemnému prostředí přijatelné a jen demonstrovalo, že se jde dostat k cíli mnohem jednodušší cestou.

Další příklady jsou mnohem zajímavější. Pro tento projekt bylo rozhodnuto, že cestou, kterou se vydáme bude formulářová. Ta je podle mě velice vhodná, protože by mohla díky svému vzhledu a rozložení podobné tištěným formulářům vyhovovat i uživatelům nenavyklých na komplikované ovládací prvky. Dalo by se říci, že uživatel si skrze uživatelské rozhraní, co vypadá jako formulář objedná simulaci podle stanovených specifikací.

Příklady dva až čtyři, demonstrují na různých zadání, které lze vidět v přílohách A, B a C, různé úrovně komplexnosti řešení množství přidaných, funkcí a kontrol. Ukazují, že každé řešení se dá neustále zlepšovat a posouvat dále. Dobře to lze vidět i na zdrojových kódech vyskytujících se v přílohách (D pro první příklad,) E, F a G, kde lze názorně sledovat stále větší velikost zdrojového kódu nepřímo úměrnou pouze rozsahu formuláře. Zvláště patrné je to vidět na čtvrtém a posledním typovém příkladu, kde kromě dodatečných funkcí se kód výrazně rozrostl i kvůli vnitřním kontrolám a skrytým funkcím.

Jestli je tahle cesta vytváření simulačního modelu efektivnější, než původní skutečně nevím. Ale, když vezmeme v úvahu, kolik zdrojů a času bylo vloženo k tvorbě daného simulačního program Tecnomatix Plant Simulation, tak by přímé srovnání nebylo asi velice férové. Bezesporu jsme, ale ukázaly zajímavou, odlišnou a graficky příjemnou cestu, která by lidem dávající přednost práci s dokumenty mohla připadat atraktivní. Zvláště kdyby se do vývoje tohoto principu vložilo více času a zdrojů. Závěrem tedy věřím, že koncept je atraktivní a má skutečně potenciál, být efektivní cestou, nebo

minimálně být rozšířením simulačního programu pro možnost alternativní tvorby modelu.



Obr. 4. 1 Závěrečná aplikace

Závěr

Cílem této bakalářské práce bylo navrhnout možnou změnu, jak generovat kód pro tacnomatix, a to jinou cestou než pro aplikaci přímo pro to určenou. V první část bakalářská práce je věnována teoretické části a vysvětlení důležitých pojmů spojených s logistickou. Hlavním účelem této části je objasnit všechny pojmy s pojené s logistikou a díky tomu objasnit čtenáři důležitost logistiky a programů s logistickou spojenými. Díky tomu by měl čtenář pochopit kde vše se logistika využívá a jak je pro ni informatická část důležitá. Důležité, je také zmínit, že důvodem, proč je tak často zvolený stejný zdroj pro parafrázi, je ten že jsem zvolil část největší a první zdroj. Ves skutečnosti, kdyby to bylo naprosto přesné bylo by tam minimálně pět odkazů, ke každému, protože informace byly získány z vícero zdrojů. Což velice dobře demonstruje velké množství zdrojů, a je docela pravděpodobně, že jsem si něco odnesl i z dalších, když jsem procházel internet a další zdroje. Tohle jsou jen ty poslední a asi nejvýraznější.

V další části bakalářská práce rozebírá možnosti různých cest spolupráce mezi zadavatelem a tvůrcem simulačního model. Na začátku jsou navrhnuté tři různé a pravděpodobné možnosti přístupy ke spolupráci a postupu vytvoření simulačního modelu a na závěr jsou podrobně vysvětlen zvolený přístup využitý pro tuto bakalářskou práci a důvody proč byl právě tento, a ne jiný zvolen.

Poslední část bakalářské části je věnována především rozboru a podrobnému objasnění vytvořených aplikací pro zadané typové příklady dodané vedoucím bakalářské práce profesorovi Gabrieli Fedorkovi. Samotná práce obsahuje jednoduší vysvětlení zadání, vysvětlení jeho fungování, a to nejen textovou formou, ale využívá i vývojový diagram a samozřejmě výsledné shrnutí a případně obrázek ukazující výsledný vzhled. Protože první příklad byla pouze konzolová aplikace, tak nebo považováno za nutné ukazovat jeho grafické rozhraní. Nejdůležitějším příkladem je bezesporu čtvrtý, na kterým ve výsledku demonstruji všechny, co jsem si na předchozích příkladech ověřily a dodaly další funkce a kontroly, aby skutečně demonstrovaly možné realistické a praktické řešení. Důležité je také propojení této části na přílohy. Protože přílohy A, B a C obsahují zadání jednotlivých příkladů, především ukázku původních výstupních kódu tacnomatixu, které by měly aplikace schopné vytvořit. První typový příklad byl jenom experimentální a zkoušelo se na něm jen experimentální možnosti a zda je tenhle projekt vůbec realistický, a proto neměl kompletní zadání. Důležité jsou i přílohy D, E, F a G, které obsahují

samotné zdrojové kódy aplikací řešící stanovené typové příklady. Celé soubory pro zadání typových příkladů v složce Zadání, zdrojové kódy programu Java ve složce Zdrojové kódy, společně s komprimovanými soubory daných aplikací JAR, které se dají spustit pomocí BAT souborů, které jsou ve stejné složce Spustitelné aplikace, se nachází na přiloženém CD.

Bakalářská práce se snaží být logicky uspořádána, aby postupně jak jejím obsahem čtenář prochází, tak se mu rozvíjely znalosti a pochopení a mohl si na konci vytvořit komplexní pohled a názor. Takže po přečtení bakalářské práce by, jste měli být schopni nejen pochopit přínos bakalářské práce v demonstraci alternativní cesty k vytvoření potřebného kódu pro tacnomatix, ale i důvod proč je to užitečné a kde všude se podobné simulace využívají. Vytvořené aplikace nejen demonstrují konkrétní možnou cestu k vytvoření kódu, ale i potenciál přidávání dalších cest k vytvoření výsledných kódů. Ukazuje tedy, že je tedy dost možné vytvořit rozšíření, aplikace, či funkce co dovolí aplikacím vytvoření stejného kódu jinou cestou, což umožňuje dát uživatelům více možností tvorby simulací, a tedy zvětšují popularitu daného programu. Konec konců, každému uživatele může vyhovovat jiný přístup k tvorbě výsledné simulace.

Seznam zdrojů

[1] BANGSOW, Steffen. Tecnomatix Plan Simulation: Modeling and Programming by Means of Examples – Second Edition. Springer Nature Switzerland AG, 2020. ISBN 978-3-030-41544-0

[2] BANGSOW, Steffen. Tecnomatix Plan Simulation: Modeling and Programming by Means of Examples. Springer Nature Switzerland AG, 2015. ISBN 978-3-319-19503-2

[3] KEOGH, James. Java bez předchozích znalostí: průvodce pro samouky. Computer Press Brno 2012. ISBN 978-80-251-0839-0

[4] SCHILDT, Herbert. Java7: Výukový kurz. Computer Press Brno 2012. ISBN 978-80-251-3748-2

[5] GROSS Ivan, BARANČÍK Ivan, ČUJAN Zdeněk. Velká kniha logistiky. VŠCHT Praha (1. vydání, 2016). ISBN 978-80-7080-952-5

[6] WELLING, Luke. Mistrovství – PHP a MySQL: Kompletní průvodce vývojáře. Computer Press Brno 2017. ISBN 978-80-251-4892-1

[7] Nové trendy v oblasti logistiky | topvision.cz. Top Vision - vzdělávací kurzy a diskusní fóra pro manažery | topvision.cz [online]. Copyright © 2015 [cit. 01.05.2022]. Dostupné z: https://www.topvision.cz/blog/nove-trendy-v-oblasti-logistiky

[8] Just in Time: Co to vlastně je? | Průmyslové Inženýrství.cz | [online]. Copyright © [cit. 01.05.2022]. Dostupné z: https://www.prumysloveinzenyrstvi.cz/just-in-time-co-to-vlastne-je/

[9] Kanban – výroba tahem. SystemOnLine.cz - ekonomické a informační systémy v praxi [online]. Copyright © 2001 [cit. 01.05.2022]. Dostupné z: https://m.systemonline.cz/rizeni-vyroby/kanban-vyroba-tahem.htm

[10] QR kód slaví 25 let: Víte, co všechno dokáže? - Computerworld. Computerworld [online]. Copyright © jamdesign [cit. 01.05.2022]. Dostupné z: https://www.computerworld.cz/clanky/qr-kod-slavi-25-let-vite-co-vsechno-dokaze/

[11] Černé díry v červených číslech 5 – příliš velké letadlo – Kechlibar.net. Kechlibar.net – Komentáře ke světovému dění [online]. Copyright © 2022 [cit. 01.05.2022]. Dostupné z: https://kechlibar.net/2019/02/27/cerne-diry-v-cervenych-cislech-5-prilis-velke-letadlo/

[12] Cross Docking -. D&V Southern Logistics Company | Trucking | Warehousing | Philippines [online]. Copyright © [cit. 01.05.2022]. Dostupné z: https://www.dnvslogistics.com/cross-docking_services/

[13] Průmysl 4.0 | Lean Industry. Průmysl 4.0 | Štíhlá výroba | Management | Lean Industry [online]. Copyright © 2022 leanindustry.cz. Vytvořilo reklamní a grafické studio [cit. 01.05.2022]. Dostupné z: https://www.leanindustry.cz/prumysl-4-0/

[14] What is Digital Twin Technology and How Does it Work? - TWI. Joining Innovation with Expertise - TWI [online]. Copyright © 2022 TWI Ltd. All rights reserved. [cit. 28.04.2022]. Dostupné z: https://www.twi-global.com/technical-knowledge/faqs/what-is-digital-twin

[15] Gartner: 10 strategických IoT technologií a trendů | RESELLER CHANNEL NETWORK NEWS. RESELLER CHANNEL NETWORK NEWS [online]. Copyright © 2022 [cit. 01.05.2022]. Dostupné z: https://www.rc-nn.com/gartner-10-strategickych-iot-technologii-a-trendu/

[16] Industrial IoT Solutions to Discover New Business Opportunities. An End-To-End IoT Solutions Provider | Biz4intellia [online]. [cit. 01.05.2022]. Dostupné z: https://www.biz4intellia.com/industrial-iot/

[17] Stock fotografie Cps Konceptuální Abstraktní Obraz Vizuální – iStock. [online]. Copyright © 2022 iStockphoto LP. [cit. 01.05.2022]. Dostupné z: https://www.istockphoto.com/cs/fotografie/cps-konceptu%C3%A1ln%C3%AD-abstraktn%C3%AD-obraz-vizu%C3%A1ln%C3%AD-gm612623626-105559187

[18] Digitální výroba 2020: Kolektivní inteligence v chytré výrobě | ANASOFT. Development of customized software, IT services: ANASOFT [online]. Copyright © Anasoft [cit. 01.05.2022]. Dostupné z: https://www.anasoft.com/emans/cz/home/Novinky-blog/Blog/Digitalni-vyroba-2020-kolektivni-inteligence-v-chytre-vyrobe/

[19] Vysočina 4.0 hub. Vysočina 4.0 hub [online]. [cit. 01.05.2022]. Dostupné z: http://www.vysocina40hub.cz/compas

[20] What is Simulation? What Does it Mean? (Definition and Examples) - TWI. Joining Innovation with Expertise - TWI [online]. Copyright © 2022 TWI Ltd. All rights reserved. [cit. 01.05.2022]. Dostupné z: https://www.twi-global.com/technical-knowledge/faqs/faq-what-is-simulation#HowSimulationWorks

[21] What are compilers, translators, interpreters, and assemblers?. Tips on coding, designing, and embedding with microcontrollers [online]. Dostupné z: https://www.microcontrollertips.com/compilers-translators-interpreters-assemblers-faq/

[22] History and development of JIT manufacturing. UK Essays | UKEssays [online]. Copyright © 2003 [cit. 01.05.2022]. Dostupné z: https://www.ukessays.com/essays/management/history-and-development-of-jit-manufacturing-management-essay.php

[23] Just-in-Time for Real Life - Snowflake Blog. Snowflake Data Cloud | Enable the Most Critical Workloads [online]. Copyright © 2022 Snowflake Inc. All Rights Reserved [cit. 01.05.2022]. Dostupné z: https://www.snowflake.com/blog/just-in-time-logistics-for-the-real-world/

[24] What Is Assembly Language (With an Example) | Indeed.com. Job Search | Indeed [online]. Copyright © 2022 Indeed [cit. 01.05.2022]. Dostupné z: https://www.indeed.com/career-advice/career-development/what-is-assembly-language

[25] Live Life Just in Time - LifeEdited. - LifeEdited [online]. Copyright © 2022 LifeEdited, All Rights Reserved. [cit. 01.05.2022]. Dostupné z: https://lifeedited.com/live-life-just-in-time/

[26] Just-In-Time with Examples – StudiousGuy. StudiousGuy – Your Study Buddy [online]. Copyright © 2022. [cit. 01.05.2022]. Dostupné z: https://studiousguy.com/just-in-time-with-examples/

[27] Rozdělení logistiky - Logistika. Logistika - Vše co student potřebuje vědět [online]. Copyright © 2022. Všechna práva vyhrazena. [cit. 01.05.2022]. Dostupné z: https://logistika-cz.studentske.cz/2009/05/rozdeleni-logistiky.html

[28] Global Shortages During Coronavirus Reveal Failings of Just in Time Manufacturing - The New York Times. The New York Times - Breaking News, US News, World News and Videos [online]. Copyright © [cit. 01.05.2022]. Dostupné z: https://www.nytimes.com/2021/06/01/business/coronavirus-global-shortages.html

[29] Rethinking Your Just-in-Time Supply Chain - Wharton. Executive Education at The Wharton School - Executive Programs [online]. Copyright © [cit. 01.05.2022]. Dostupné z: https://executiveeducation.wharton.upenn.edu/thought-leadership/wharton-at-work/2021/08/rethinking-your-supply-chain/

[30] Just-in-time supply chains after the Covid-19 crisis | VOX, CEPR Policy Portal. VOX, CEPR Policy Portal [online]. [cit. 01.05.2022]. Dostupné z: https://voxeu.org/article/just-time-supply-chains-after-covid-19-crisis

[31] Pros & Cons of the JIT Inventory System | Small Business - Chron.com. Small Business - Chron.com [online]. Copyright © 2022 Hearst [cit. 01.05.2022]. Dostupné z: https://smallbusiness.chron.com/pros-cons-jit-inventory-system-3195.html

[32] Advantages and Disadvantages of Just-In-Time (JIT) Manufacturing. APS - Advanced Planning Scheduling Software & System | PlanetTogether [online]. Copyright ©2022 PlanetTogether [cit. 01.05.2022]. Dostupné z: https://www.planettogether.com/blog/advantages-and-disadvantages-of-just-in-time-jit-manufacturing

[33] JIT Just-in-Time manufacturing. Institute for Manufacturing (IfM) [online]. Copyright © University of Cambridge 2016 [cit. 01.05.2022]. Dostupné z: https://www.ifm.eng.cam.ac.uk/research/dstools/jit-just-in-time-manufacturing/

[34] Co je tlumočník a kde se používá? . Domů - puntomarinero.com [online]. Copyright © 2018 [cit. 01.05.2022]. Dostupné z: https://cs.puntomarinero.com/what-is-the-interpreter-and/

[35] History and development of JIT manufacturing - 2022 Essay (Analysis). Oessays - The Biggest Essays Database of 2022 [online]. Copyright © 2022 oessays.com [cit. 01.05.2022]. Dostupné z: https://oessays.com/document/history-and-development-of-jit-manufacturing/

[36] Co je kompilátor kódu a co dělá?. CS.EFERRIT.COM [online]. Copyright © 2022 cs.eferrit.com [cit. 01.05.2022]. Dostupné z: https://cs.eferrit.com/definice-a-ucel-prekladace/

[37] Interpreter Vs Compiler : Differences Between Interpreter and Compiler. Programiz: Learn to Code for Free [online]. Copyright © Parewa Labs Pvt. Ltd. All rights reserved. [cit. 01.05.2022]. Dostupné z: https://www.programiz.com/article/difference-compiler-interpreter

[38] A Brief History of Logistics - Universal Cargo. Universal Cargo | Freight Forwarder & International Logistics [online]. Copyright © 2019 Universal Cargo Management. All Rights Reserved. [cit. 01.05.2022]. Dostupné z: https://www.universalcargo.com/a-brief-history-of-logistics/

[39] History of Logistics and Its Progression in eCommerce -Shiprocket. eCommerce Logistics & Shipping Solutions: Courier Aggregator India - Shiprocket [online]. [cit. 01.05.2022]. Dostupné z: https://www.shiprocket.in/blog/history-of-logistics-and-its-progression-in-ecommerce/

[40] ECR - Efficient Consumer Response. GS1 | GS1 Switzerland [online]. Copyright © GS1 Switzerland 2021. [cit. 01.05.2022]. Dostupné z: https://www.gs1.ch/en/home/topics/ecr---efficient-consumer-response

[41] What is Just-in-Time (JIT)? | Just-in-Time Inventory management. Zoho - Cloud Software Suite and SaaS Applications for Businesses [online]. [cit. 01.05.2022]. Dostupné z: https://www.zoho.com/inventory/guides/what-is-just-in-time.html

[42] Cross Docking Definition: How Does Cross-Docking Work. BlueCart | The Leader in Hospitality Technology [online]. [cit. 01.05.2022]. Dostupné z: https://www.bluecart.com/blog/cross-docking-definition

[43] Introduction to Kanban Guide – What is Kanban? | Planview. Planview Portfolio Management and Work Management Software [online]. Copyright © 2022. Planview, Inc. All Rights Reserved. [cit. 01.05.2022]. Dostupné z: https://www.planview.com/resources/guide/introduction-to-kanban/

[44] The Origins of Just-In-Time | Quality and Innovation. Quality and Innovation | Performance Excellence & Data Science for Digital Transformation in Industry 4.0 [online]. Copyright ©2008 [cit. 01.05.2022]. Dostupné z: https://qualityandinnovation.com/2010/10/13/the-origins-of-just-in-time/

[45] A Short History of Logistics Technology – Logmore Blog. Logmore – Data logging for any industry or scale [online]. Copyright © 2020 [cit. 01.05.2022]. Dostupné z: https://www.logmore.com/post/short-history-logistics-tech

[46] Logistics Definition - What is Logistics. Start and grow your e-commerce business - 14-Day Free Trial [online]. Dostupné z: https://www.shopify.com/encyclopedia/logistics

[47] Examples of Successful JIT Systems - Toyota, Dell, and Harley Davidson - BrightHub Project Management. BrightHub Project Management [online]. Copyright © 2021 Bright Hub PM. All Rights Reserved. [cit. 01.05.2022]. Dostupné z: https://www.brighthubpm.com/methods-strategies/71540-real-life-examples-of-successful-jit-systems/

[48] Just-in-Time Inventory (JIT) Explained: A Guide | NetSuite. [online]. Copyright © [cit. 01.05.2022]. Dostupné z: https://www.netsuite.com/portal/resource/articles/inventory-management/just-in-time-inventory.shtml

[49] Just in time | ALTAXO. ★ Zakládání a prodej ready made společností, virtuální sídla, vedení účetnictví - ALTAXO [online]. Copyright © 2019, ALTAXO SE [cit. 01.05.2022]. Dostupné z: https://www.altaxo.cz/provoz-firmy/management/rady-pro-manazery/just-in-time

[50] Quick Response | Computerworld. IT news, careers, business technology, reviews | Computerworld [online]. Copyright © 2001 IDG Communications, Inc. [cit. 01.05.2022]. Dostupné z: https://www.computerworld.com/article/2591779/quick-response.html

[51] ECR - Efficient Consumer Response. GS1 | GS1 Switzerland [online]. Copyright © GS1 Switzerland 2021. [cit. 01.05.2022]. Dostupné z: https://www.gs1.ch/en/home/topics/ecr---efficient-consumer-response

[52] Hub-and-spoke arrangements in competition - OECD. Home page - OECD [online]. Copyright © Organisation for Economic [cit. 01.05.2022]. Dostupné z: https://www.oecd.org/competition/hub-and-spoke-arrangements.htm

[53] What is Cloud Computing? Definition, Examples, & Uses. Managed IT Services from Network Coverage | NetCov.com [online]. Copyright © 2018 [cit. 01.05.2022]. Dostupné z: https://www.netcov.com/what-is-cloud-computing/

[54] What is the Internet of Things, and how does it work?. [online]. Copyright © Copyright IBM Corp. 2022 [cit. 01.05.2022]. Dostupné z: https://www.ibm.com/blogs/internet-of-things/what-is-the-iot/

[55] What is the Industrial Internet of Things (IIoT)? - Definition from Techopedia. Techopedia: Educating IT Professionals To Make Smarter Decisions [online]. Copyright © 2022 [cit. 01.05.2022]. Dostupné z: https://www.techopedia.com/definition/33015/industrial-internet-of-things-iiot

[56] What are Cyber-Physical Systems?; Cyber-Physical Lab; Newcastle University. research.ncl.ac.uk; ; Newcastle University [online]. [cit. 01.05.2022]. Dostupné z: https://research.ncl.ac.uk/cplab/aboutthelab/whatarecyber-physicalsystems/

[57] What is Smart Manufacturing? -. Instructions de travail digitales pour l'industrie 4.0 - Picomto [online]. Copyright © 2021 [cit. 01.05.2022]. Dostupné z: https://www.picomto.com/en/what-is-smart-manufacturing/

[58] What is the Industrial Internet of Things (IIoT)? - Definition from Techopedia. Techopedia: Educating IT Professionals To Make Smarter Decisions [online]. Copyright © 2022 [cit. 01.05.2022]. Dostupné z: https://www.techopedia.com/definition/33015/industrial-internet-of-things-iiot

[59] What Is A Smart Factory? (And What It Means For You) | Tulip. Tulip | The Industry's Leading Frontline Operations Platform [online]. [cit. 01.05.2022]. Dostupné z: https://tulip.co/glossary/what-is-a-smart-factory-and-what-it-means-for-you/

[60] Just-in-time in the Real World of Inventory Management - Unleashed Software. Inventory Management Software - Unleashed Software [online]. Copyright © 2022 Unleashed Software. All Rights reserved. [cit. 01.05.2022]. Dostupné z: https://www.unleashedsoftware.com/blog/just-time-real-world-inventory-management

[61] The Center - Advanced Manufacturing with Accelerating Technology. The Center - Home [online]. Copyright © 2022 THE CENTER. ALL RIGHTS RESERVED. [cit. 01.05.2022]. Dostupné z: https://www.the-center.org/Our-Services/Advanced-Technology/Industry-4-0/Industry-4-0-Simulation

[62] Tutorial Kart - Best Online Learning Site for Free Tutorials, Online Training, Courses, Materials and Interview Questions - SAP Tutorial, Salesforce Tutorial, Informatica Tutorial, Kotlin Tutorial, Kotlin Android Tutorial, Spark Tutorial, Kafka Tutorial, Node JS Tutorial. Tutorial Kart - Best Online Learning Site for Free Tutorials, Online Training, Courses, Materials and Interview Questions - SAP Tutorial, Salesforce Tutorial, Informatica Tutorial, Kotlin Tutorial, Kotlin Android Tutorial, Spark Tutorial, Kafka Tutorial, Node JS Tutorial [online]. Copyright ©Copyright [cit. 01.05.2022]. Dostupné z: https://www.tutorialkart.com/

[63] The Fourth Industrial Revolution: what it means and how to respond | World Economic Forum. The World Economic Forum [online]. Copyright © [cit. 01.05.2022]. Dostupné z: https://www.weforum.org/agenda/2016/01/the-fourth-industrial-revolution-what-it-means-and-how-to-respond/

[64] Planning for Industry 4.0 with Simulation | SIMUL8. Simul8 | Fast, Intuitive Simulation Software for Desktop and Web [online]. Copyright © [cit. 01.05.2022]. Dostupné z: https://www.simul8.com/applications/manufacturing/implementing-industry-4-0-with-simulation

[65] What is a Digital Factory? | TIBCO Software. Global Leader in Integration and Analytics Software | TIBCO Software [online]. Copyright © 2022 TIBCO Software Inc. [cit. 01.05.2022]. Dostupné z: https://www.tibco.com/reference-center/what-is-a-digital-factory

[66] What is a digital factory, and how do they relate to digital twins?. Bridging the gap between the physical and digital world | NavVis [online]. Copyright © 2019 MyFonts Inc [cit. 01.05.2022]. Dostupné z: https://www.navvis.com/blog/what-is-a-digital-factory-and-how-do-they-relate-to-digital-twins

[67] What is the Difference Between a Simulation and a Digital Twin?. EXOR International | Industrial Automation [online]. Copyright ©2019 Exor International [cit. 01.05.2022]. Dostupné z: https://www.exorint.com/en/blog/what-is-the-difference-between-a-simulation-and-a-digital-twin

[68] What is Industry 4.0 and What Does it Mean for My Manufacturing?. [online]. Copyright © 2018 Saint Clair Systems All rights reserved [cit. 01.05.2022]. Dostupné z: https://blog.viscosity.com/blog/what-is-industry-4.0-and-what-does-it-mean-for-my-manufacturing

[69] What is Industry 4.0? How Does it Work? (A Beginners Guide) - TWI. Joining Innovation with Expertise - TWI [online]. Copyright © 2022 TWI Ltd. All rights reserved. [cit. 01.05.2022]. Dostupné z: https://www.twi-global.com/what-we-do/research-and-technology/technologies/industry-4-0

[70] What is Industry 4.0? | The Industrial Internet of Things | Epicor. Object moved [online]. Copyright © Epicor Software Corporation 2022. [cit. 01.05.2022]. Dostupné z: https://www.epicor.com/en/blog/learn/what-is-industry-4-0/¨

[71] Industry 4.0 and the fourth industrial revolution explained. i-SCOOP | Digital business and transformation hub [online]. Copyright © Copyright i [cit. 01.05.2022]. Dostupné z: https://www.i-scoop.eu/industry-4-0/

[72] Industry 4.0: At the Front of the Fourth Industrial Revolution | ICL Group. ICL Group: Global manufacturer of Specialty Minerals and Fertilizers [online]. Copyright © 2021 [cit. 01.05.2022]. Dostupné z: https://www.icl-group.com/blog/industry-4-0-at-the-front-of-the-fourth-industrial-revolution/

[73] itnetwork.cz - Učíme národ IT. itnetwork.cz - Učíme národ IT [online]. Copyright © 2022 itnetwork.cz. Veškerý obsah webu [cit. 01.05.2022]. Dostupné z: https://www.itnetwork.cz/

[74] Tecnomatix Plant Simulation Foundation - Siemens Digital Industries Software. Document Moved [online]. Copyright ©2022 Siemens [cit. 01.05.2022]. Dostupné z: https://trials.sw.siemens.com/tecnomatix-plant-simulation-foundation/

Seznam grafických objektů

Seznam obrázků

Seznam diagramů

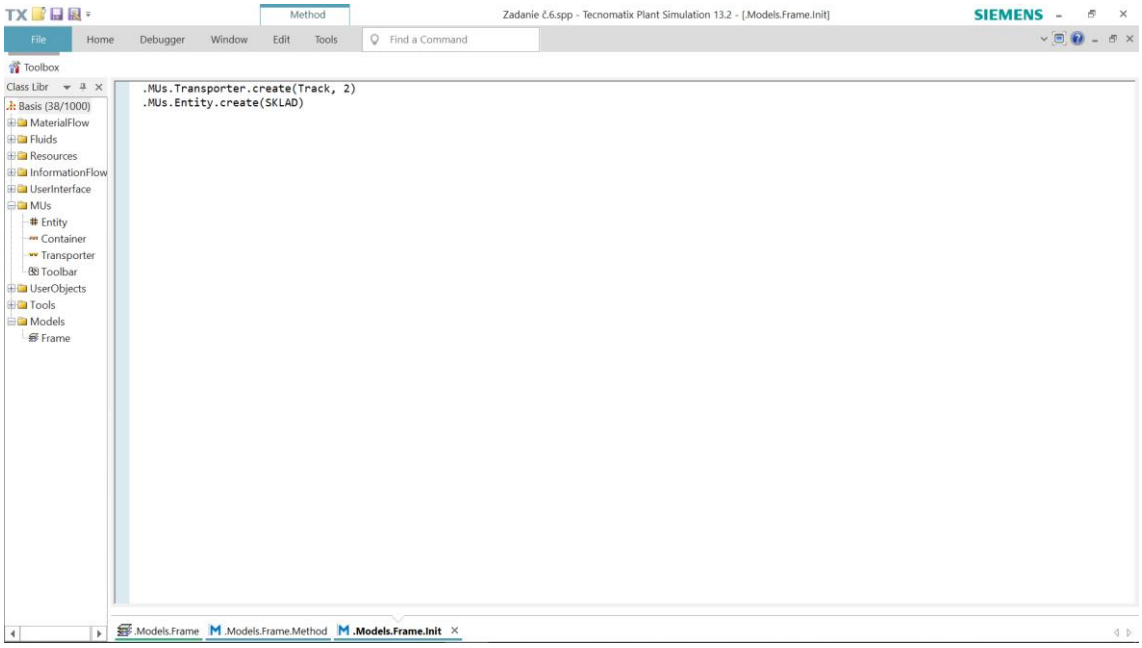
Seznam zkratek

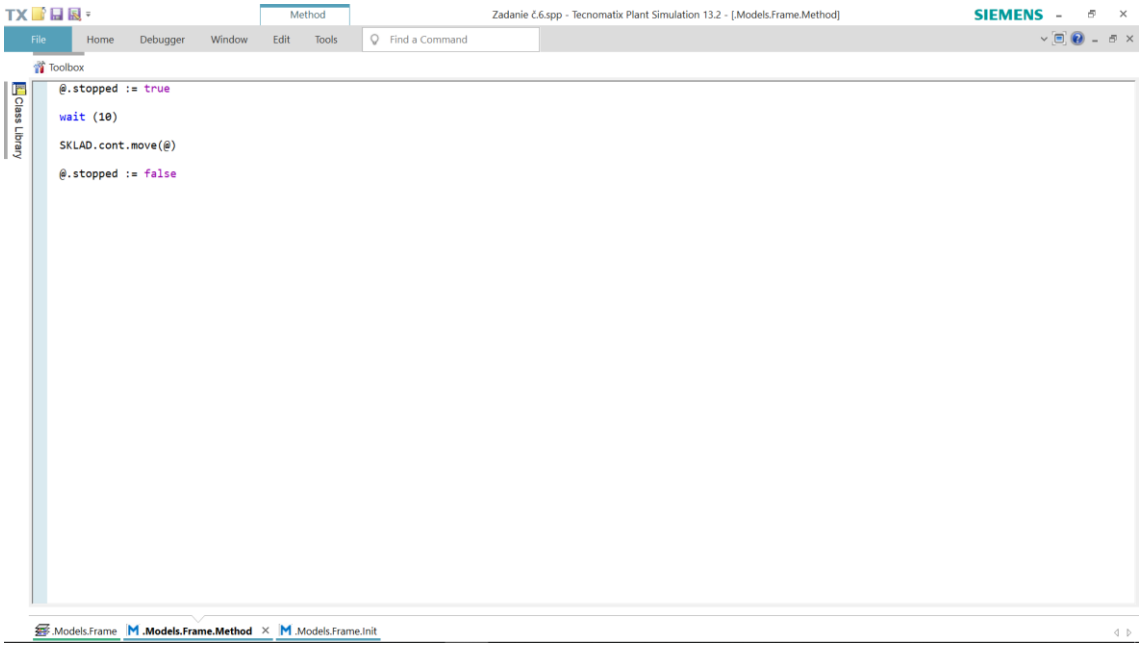
CPS	Cyber-Physical Systems
ECR	Efficent Consumer Response
H&S	Hub and Spoke
IIoT	Industrial Internet of Things
IoT	Internet of Things
JIC	Just-In-Case
JIT	Just-In-Time
QR	Quick Response

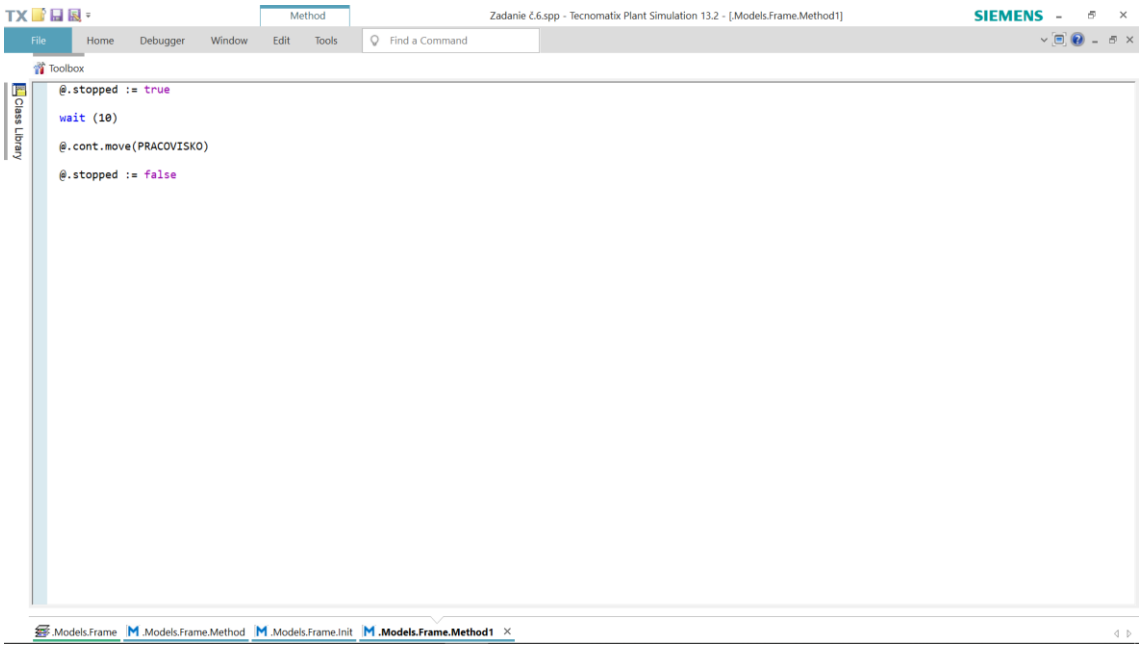
Seznam příloh

Příloha A

Zadání druhého typového příkladu

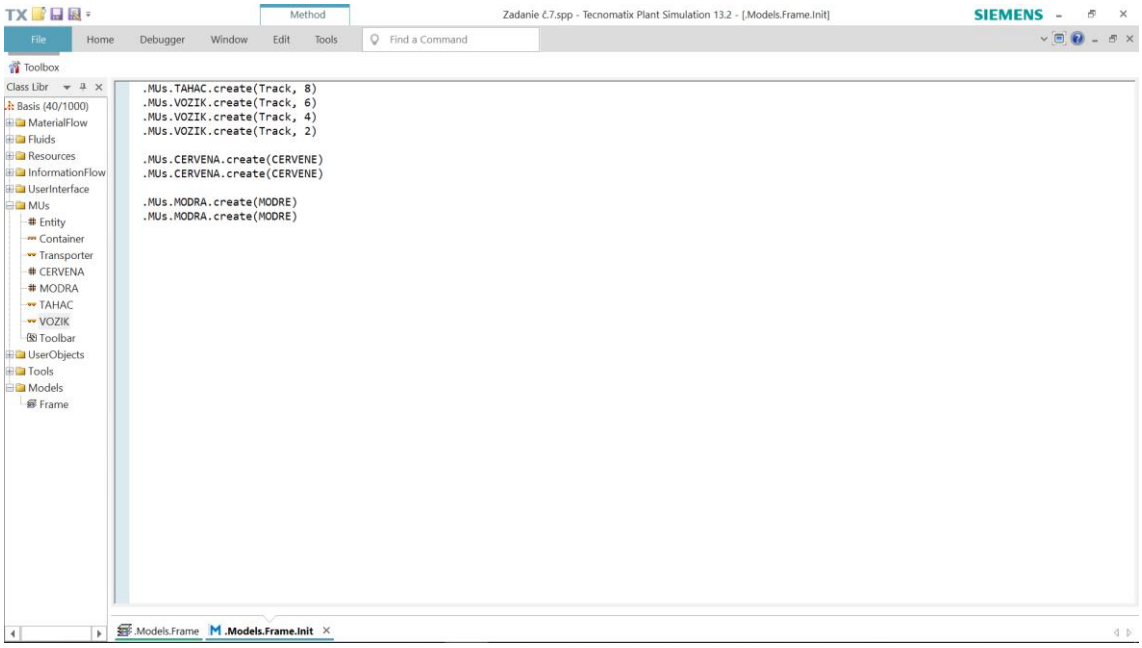


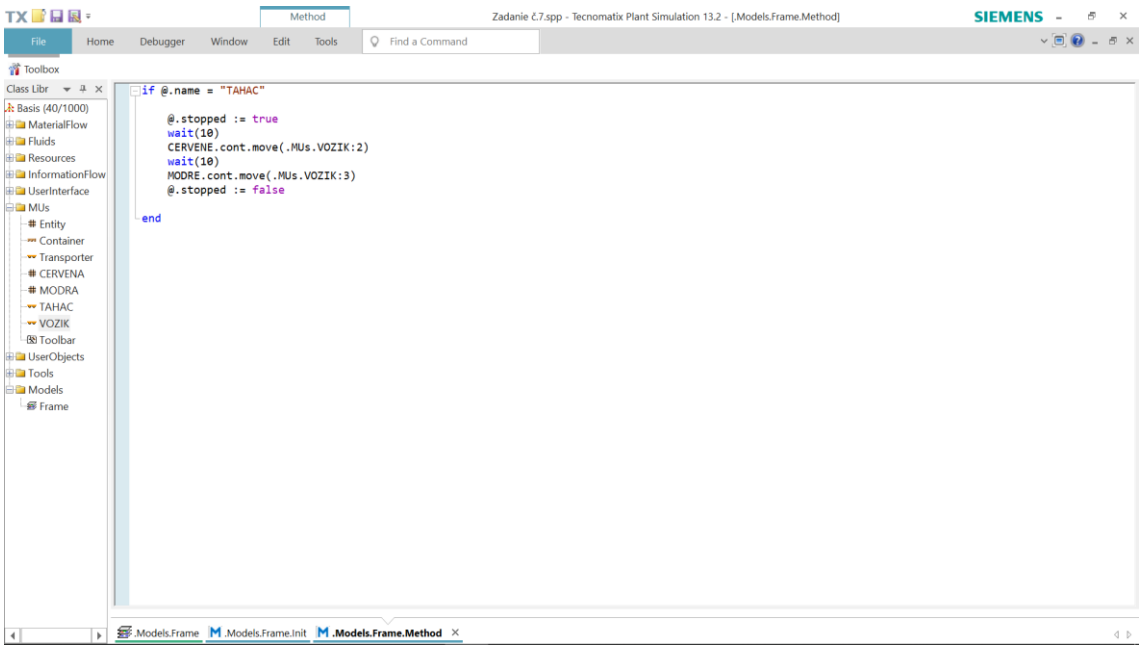


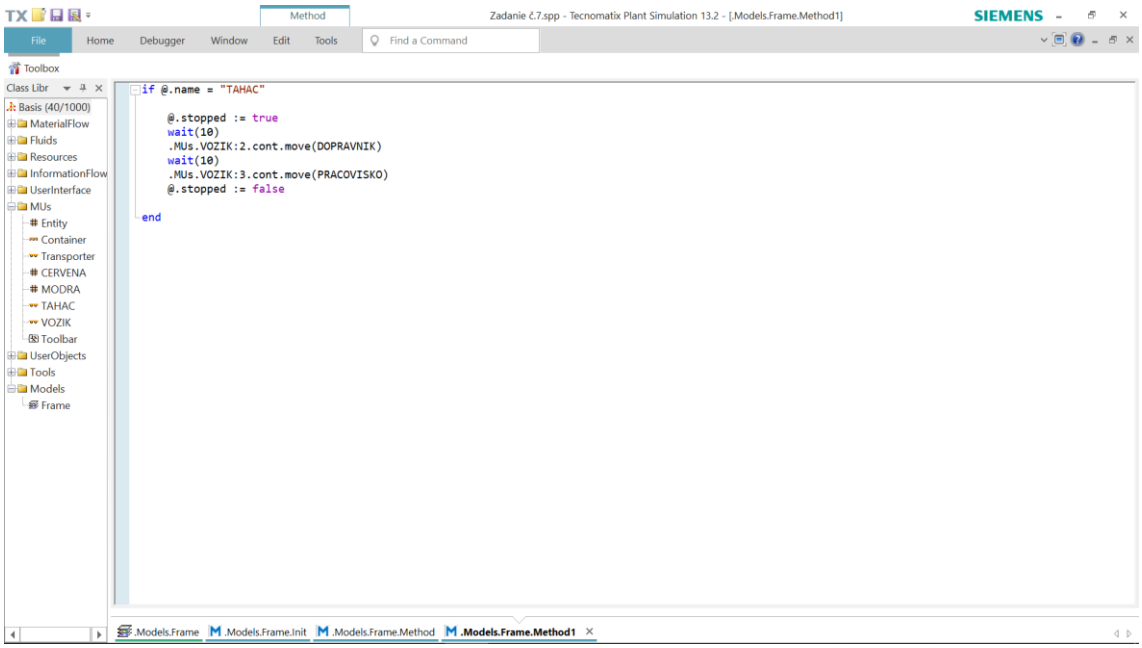

Method
Zadanie č.6.spp - Tecnomatix Plant Simulation 13.2 - [.Models.Frame.Method1]
SIEMENS
File
Home
Debugger
Window
Edit
Tools
Find a Command
Toolbox
Class Library
@.stopped := true
wait (10)
@.cont.move(PRACOVISKO)
@.stopped := false
.Models.Frame
.Models.Frame.Method
.Models.Frame.Init
.Models.Frame.Method1

Příloha B

Zadání třetího typového příkladu






Method
Zadanie č.7.spp - Tecnomatix Plant Simulation 13.2 - [.Models.Frame.Method1]
SIEMENS
File
Home
Debugger
Window
Edit
Tools
Find a Command
Toolbox
Class Libr
Basis (40/1000)
MaterialFlow
Fluids
Resources
InformationFlow
UserInterface
MUs
Entity
Container
Transporter
CERVENA
MODRA
TAHAC
VOZIK
Toolbar
UserObjects
Tools
Models
Frame
if @.name = "TAHAC"
@.stopped := true
wait(10)
.MUs.VOZIK:2.cont.move(DOPRAVNIK)
wait(10)
.MUs.VOZIK:3.cont.move(PRACOVISKO)
@.stopped := false
end
.Models.Frame
.Models.Frame.Init
.Models.Frame.Method
.Models.Frame.Method1

Příloha C

Zadání čtvrtého typového příkladu

...Tab_Sensors

4

	integer 1
string	Sensor Position
1	4
2	9
3	14
4	19
5	24
6	
7	
8	
9	
10	
11	
12	

.Models.Model.Met_Sensor_Creation

```
var i : integer
var Line : object

Line := Track -- Set Line for sensors creation

for i := 1 to Line.numsensors -- Delete all sensors on line

    Line.deletesensor(i) -- Delete sensor

next

for i := 1 to Tab_Sensors.ydim -- Create all sensors according table

    Line.createsensor(Tab_Sensors[1,i]) -- Create sensor according table
    Line.sensorID(i).ctrl:=&Met_Sensor_Identification -- Assign method to created sensor

next
```

.Models.Model.Met_Attributes_Setting

```
@.createAttr("Storage","object") -- Assign attribute Storage to Forklift
@.createAttr("Sensor","integer") -- Assign attribute Sensor to Forklift
```

.Models.Model.Tab_Materials

.UserObjects.Material

	object 1	real 2	integer 3	string 4	table 5
string	MU	Portion	Number	Name	Attributes
1	.UserObjects.Material	20.00		Red_Box	
2	.UserObjects.Material	20.00		Blue_Box	
3	.UserObjects.Material	20.00		Green_Box	
4	.UserObjects.Material	20.00		Orange_Box	
5	.UserObjects.Material	20.00		Pink_Box	
6					
7					
8					
9					
10					
11					
12					
13					
14					
15					
16					
17					

.Models.Model.Met_Color_Setting

```
-- ==== Box coloring (RGB Values) according name
if @.name = "Red_Box"

    @.vectorgraphicscolor := makeRGBValue(255,0,0)

elseif @.name = "Blue_Box"

    @.vectorgraphicscolor := makeRGBValue(0,0,255)

elseif @.name = "Green_Box"

    @.vectorgraphicscolor := makeRGBValue(0,255,0)

elseif @.name = "Orange_Box"

    @.vectorgraphicscolor := makeRGBValue(255,190,0)

elseif @.name = "Pink_Box"

    @.vectorgraphicscolor := makeRGBValue(255,0,255)

end
```

.Models.Model.Met_Sensor_Identification

```
param SensorID: integer, Front: boolean, BookPos: boolean

if @.Sensor = SensorID -- Stop when you are on good position in warehouse

    @.stopped := true -- Stop

    wait 5 -- Process time

    @.cont.move(@.Storage) -- Unload box to storage

    @.Storage := "" -- Reset attribute
    @.Sensor := 0 -- Reset attribute

    @.stopped := false -- Move

end
```

Příloha D

Zdrojový kód prvního typového příkladu

```
import java.io.File;  // Import the File class
import java.io.FileWriter;   // Import the FileWriter class
import java.io.IOException;  // Import the IOException class to handle errors
import java.io.*;

public class MailClass {

        public static void main(String[] args) {
                String NazevVlacku = "";
                int PocetNazvuVlacku = 0;
                int PocetVlacku;
                int PocetEntit;
                int PocetTypuEntit;
                String NazevEntity = "";
            InputStreamReader streamReader = new InputStreamReader(System.in);
            BufferedReader bufferedReader = new BufferedReader(streamReader);

            try {
                File myObj = new File("seminarka.txt");
                if (myObj.createNewFile()) {
                  System.out.println("File created: " + myObj.getName());
                } else {
                  System.out.println("File already exists.");
                }
              } catch (IOException e) {
                System.out.println("An error occurred.");
                e.printStackTrace();
              }

            try {
                FileWriter myWriter2 = new FileWriter("seminarka_presun.txt");
                FileWriter myWriter1 = new FileWriter("seminarka_inicializace.txt");

                System.out.println("Počet názvů vláčků");
                String pocet1 = bufferedReader.readLine();
                PocetNazvuVlacku = Integer.parseInt(pocet1);
                myWriter2.write("@.stopped := true");
                myWriter2.write(System.getProperty("line.separator"));
                myWriter2.write("waint(10)");
                myWriter2.write(System.getProperty("line.separator"));

                for (int i = 0; i < PocetNazvuVlacku; i++)
                {
                    System.out.println("Název vláčků");
                    NazevVlacku = bufferedReader.readLine();
                        myWriter2.write(NazevVlacku + ".cont.move(@)");
                        myWriter2.write(System.getProperty("line.separator"));
                    System.out.println("Počet vláčků s názvem " + NazevVlacku);
                    String pocet2 = bufferedReader.readLine();
                    PocetVlacku = Integer.parseInt(pocet2);
                    for (int j = 0; j < PocetVlacku; j++)
                    {
                        System.out.println("Pozice " + (j+1) + ". vláčku s názvem " +
NazevVlacku);
                        String pocet = bufferedReader.readLine();
                        PocetNazvuVlacku = Integer.parseInt(pocet);
                        myWriter1.write(".MUs."+NazevVlacku+".create(Track, "+pocet+")");
                        myWriter2.write(System.getProperty("line.separator"));

                    }
                }
                System.out.println("Pocet názvů entit");
                String pocet3 = bufferedReader.readLine();
                PocetTypuEntit = Integer.parseInt(pocet3);
                for (int i = 0; i < PocetTypuEntit; i++)
                {
                    System.out.println("Název entity");
                    NazevEntity = bufferedReader.readLine();
                        myWriter2.write("@.cont.move(" + NazevEntity + ")");
```

```
                    myWriter2.write(System.getProperty("line.separator"));
                System.out.println("Počet entit s názvem " + NazevEntity);
                String pocet4 = bufferedReader.readLine();
                PocetEntit = Integer.parseInt(pocet4);
                for (int j = 0; j < PocetEntit; j++)
                {

                    myWriter1.write(".MUs."+NazevEntity+".create("+NazevEntity+")");
                    myWriter2.write(System.getProperty("line.separator"));
                }
            }
            myWriter2.write("@.stopped := false");
            myWriter2.write(System.getProperty("line.separator"));
            myWriter1.close();
            myWriter2.close();
            System.out.println("Successfully wrote to the file.");
          } catch (IOException e) {
            System.out.println("An error occurred.");
            e.printStackTrace();
          }
        }

}
```

Příloha E

Zdrojový kód druhého typového příkladu

```
import javafx.geometry.Insets;

import java.io.File;
import java.io.FileWriter;
import java.io.IOException;
import javafx.application.Application;
import javafx.geometry.Pos;
import javafx.scene.Scene;
import javafx.scene.control.*;
import javafx.scene.layout.GridPane;

import javafx.stage.Stage;

public class Main extends Application{
        Stage window;
    public void start(Stage primaryStage) throws Exception {
        int pocet = 400;

        GridPane grid =new GridPane();
        grid.setAlignment(Pos.CENTER);
        grid.setVgap(10);
        grid.setHgap(10);
        grid.setPadding(new Insets(10));
        //ComboBox comboBox;
        Button btn1=new Button("Tisk");

        Label lblMUs = new Label("Názav MUs");
        grid.add(lblMUs,0,6);
        Label lblDis = new Label("Délka cesty");
        grid.add(lblDis,0,7);
        Label lblOdkud = new Label("Pocatecni entita");
        grid.add(lblOdkud,0,8);
        Label lblKam = new Label("Koncova entita");
        grid.add(lblKam,0,9);

        Label lblInit = new Label("Nazev init txt");
        grid.add(lblInit,0,1);
        Label lblMethod = new Label("Nazev method txt");
        grid.add(lblMethod,0,2);
        Label lblMethod2 = new Label("Nazev method1 txt");
        grid.add(lblMethod2,0,3);
        Label lblWait = new Label("Doba čekání");
        grid.add(lblWait,0,5);

        TextField txtMUs = new TextField();
        grid.add(txtMUs,1,6);
        TextField txtDis = new TextField();
        grid.add(txtDis,1,7);

        TextField txtOdkud = new TextField();
        grid.add(txtOdkud,1,8);
        TextField txtKam = new TextField();
        grid.add(txtKam,1,9);
        TextField txtInit = new TextField();
        grid.add(txtInit,1,1);
        TextField txtMethod = new TextField();
        grid.add(txtMethod,1,2);
        TextField txtMethod2 = new TextField();
        grid.add(txtMethod2,1,3);
        grid.add(btn1,1,10);
        TextField txtWait = new TextField();
        grid.add(txtWait,1,5);

        btn1.setOnAction(action -> {

           String     MUs=txtMUs.getText();
           String     Dis=txtDis.getText();
           String     Odkud=txtOdkud.getText();
           String     Kam=txtKam.getText();
```

```
String     Init=txtInit.getText();
Init = Init+".txt";
String     Method=txtMethod.getText();
Method = Method+".txt";
String     Method2=txtMethod2.getText();
Method2 = Method2+".txt";
String     WaitInt=txtWait.getText();

try {
    File myObj = new File(Init);
    if (myObj.createNewFile()) {
      System.out.println("File created: " + myObj.getName());
    } else {
      System.out.println("File already exists.");
    }
  } catch (IOException e) {
    System.out.println("An error occurred.");
    e.printStackTrace();
  }

try {
    File myObj = new File(Method);
    if (myObj.createNewFile()) {
      System.out.println("File created: " + myObj.getName());
    } else {
      System.out.println("File already exists.");
    }
  } catch (IOException e) {
    System.out.println("An error occurred.");
    e.printStackTrace();
  }

try {
    File myObj = new File(Method2);
    if (myObj.createNewFile()) {
      System.out.println("File created: " + myObj.getName());
    } else {
      System.out.println("File already exists.");
    }
  } catch (IOException e) {
    System.out.println("An error occurred.");
    e.printStackTrace();
  }
try {
    FileWriter myWriter1 = new FileWriter(Init);
    FileWriter myWriter2 = new FileWriter(Method);
    FileWriter myWriter3 = new FileWriter(Method2);

   myWriter1.write(".MUs."+MUs+".create(Track, "+Dis+")");
    myWriter1.write(System.getProperty("line.separator"));
    myWriter1.write(".MUs.Entity.create("+Dis+")");
    myWriter1.write(System.getProperty("line.separator"));

    myWriter2.write("@.stopped := true");
    myWriter2.write(System.getProperty("line.separator"));
    myWriter3.write("waint("+WaitInt+")");
    myWriter2.write(System.getProperty("line.separator"));
    myWriter2.write(Odkud+".cont.move(@)");
    myWriter2.write(System.getProperty("line.separator"));
    myWriter2.write("@.stopped := false");
    myWriter2.write(System.getProperty("line.separator"));

    myWriter3.write("@.stopped := true");
    myWriter3.write(System.getProperty("line.separator"));
    myWriter3.write("waint("+WaitInt+")");
    myWriter3.write(System.getProperty("line.separator"));
    myWriter3.write("@.cont.move("+Kam+")");
    myWriter3.write(System.getProperty("line.separator"));
    myWriter3.write("@.stopped := false");
    myWriter3.write(System.getProperty("line.separator"));

    myWriter1.close();
    myWriter2.close();
    myWriter3.close();
```

```
            } catch (IOException e) {
              System.out.println("An error occurred.");
              e.printStackTrace();
            }

        });
        Scene scene=new Scene(grid,600,pocet);
        primaryStage.setTitle("BakalarAplcation2");
        primaryStage.setScene(scene);
        primaryStage.show();
    }
    public static void main (String[] args)
    {
        launch(args);
    }
}
```

Zdrojový kód třetího typového příkladu

```
import javafx.geometry.Insets;

import java.io.File;
import java.io.FileWriter;
import java.io.IOException;

import javafx.application.Application;
import javafx.geometry.Pos;
import javafx.scene.Scene;
import javafx.scene.control.*;
import javafx.scene.layout.GridPane;
import javafx.stage.Stage;

public class Main  extends Application{
        Stage window;String     Vozidel; int VN;
    public void start(Stage primaryStage) throws Exception {

      Button btn1=new Button("Tisk");
        int pocet = 500;

        GridPane grid =new GridPane();
        grid.setAlignment(Pos.CENTER);
        grid.setVgap(10);
        grid.setHgap(10);
        grid.setPadding(new Insets(10));

       Label lblInit = new Label("Nazev init txt");
        grid.add(lblInit,0,1);
        Label lblMethod = new Label("Nazev Mmethod txt");
        grid.add(lblMethod,0,2);
       Label lblMethod2 = new Label("Nazev method1 txt");
        grid.add(lblMethod2,0,3);

        TextField txtInit = new TextField();
        grid.add(txtInit,1,1);
        TextField txtMethod = new TextField();
        grid.add(txtMethod,1,2);
        TextField txtMethod2 = new TextField();
        grid.add(txtMethod2,1,3);
        grid.add(btn1,3,14);
      final ComboBox PocetBox = new ComboBox();
        PocetBox.getItems().addAll(
            0,
            1,
            2,
            3,
            4,
            5,
            6,
            7,
            8,
            9,
            10
        );
        final ComboBox EntityBox = new ComboBox();
        EntityBox.getItems().addAll(
            0,
            1,
            2,
            3,
            4,
            5
        );

        Label lblWait = new Label("Doba čekání");
        grid.add(lblWait,2,1);
        grid.add(PocetBox,3,1);
        Label lblNaklad = new Label("Odkud nakládat");
        grid.add(lblNaklad,2,2);
```

```
        TextField txtNaklad = new TextField();
        grid.add(txtNaklad,3,2);
       Label lblPreklad = new Label("Kde překládat");
        grid.add(lblPreklad,2,3);
        TextField txtPreklad = new TextField();
        grid.add(txtPreklad,3,3);
        //comboBox.getSelectionMode().selectFirst();

        Label lblEntity = new Label("Pocet vozidel");
        grid.add(lblEntity,0,4);
        grid.add(EntityBox,1,4);
        //Label lblTahac = new Label("Pocet entit");
        //grid.add(lblTahac,2,4);
        //grid.add(TahacBox,3,4);

        Label lblVozik1 = new Label("1.Vozik");
        grid.add(lblVozik1,0,5);
        Label lblVozik2 = new Label("2.Vozik");
        grid.add(lblVozik2,0,6);
        Label lblVozik3 = new Label("3.Vozik");
        grid.add(lblVozik3,0,7);
        Label lblVozik4 = new Label("4.Vozik");
        grid.add(lblVozik4,0,8);
        Label lblVozik5 = new Label("5.Vozik");
        grid.add(lblVozik5,0,9);

        TextField txtVozik1 = new TextField();
        grid.add(txtVozik1,1,5);
        txtVozik1.setEditable(false);
        TextField txtVozik2 = new TextField();
        grid.add(txtVozik2,1,6);
        txtVozik2.setEditable(false);
        TextField txtVozik3 = new TextField();
        grid.add(txtVozik3,1,7);
        txtVozik3.setEditable(false);
        TextField txtVozik4 = new TextField();
        grid.add(txtVozik4,1,8);
        txtVozik4.setEditable(false);
        TextField txtVozik5 = new TextField();
        grid.add(txtVozik5,1,9);
        txtVozik5.setEditable(false);

        Label lblVozik1b = new Label("Umisteni");
        grid.add(lblVozik1b,2,5);
        TextField txtVozik1b = new TextField();
        grid.add(txtVozik1b,3,5);
        txtVozik1b.setEditable(false);
        Label lblVozik2b = new Label("Umisteni");
        grid.add(lblVozik2b,2,6);
        TextField txtVozik2b = new TextField();
        grid.add(txtVozik2b,3,6);
        txtVozik2b.setEditable(false);
        Label lblVozik3b = new Label("Umisteni");
        grid.add(lblVozik3b,2,7);
        TextField txtVozik3b = new TextField();
        grid.add(txtVozik3b,3,7);
        txtVozik3b.setEditable(false);
        Label lblVozik4b = new Label("Umisteni");
        grid.add(lblVozik4b,2,8);
        TextField txtVozik4b = new TextField();
        grid.add(txtVozik4b,3,8);
        txtVozik4b.setEditable(false);
        Label lblVozik5b = new Label("Umisteni");
        grid.add(lblVozik5b,2,9);
        TextField txtVozik5b = new TextField();
        grid.add(txtVozik5b,3,9);
        txtVozik5b.setEditable(false);

        EntityBox.setOnAction(action -> {

            //   Combo=PocetBox.getValue();
          Vozidel=PocetBox.getValue().toString();
            int VozidelNumber=Integer.parseInt(Vozidel);
            VN=VozidelNumber;
```

```
            switch (VozidelNumber) {
            case 0:
                txtVozik1.setEditable(false);
                txtVozik1b.setEditable(false);
                txtVozik2.setEditable(false);
                txtVozik2b.setEditable(false);
                txtVozik3.setEditable(false);
                txtVozik3b.setEditable(false);
                txtVozik4.setEditable(false);
                txtVozik4b.setEditable(false);
                txtVozik5.setEditable(false);
                txtVozik5b.setEditable(false);
                break;
            case 1:
                txtVozik1.setEditable(true);
                txtVozik1b.setEditable(true);
                txtVozik2.setEditable(false);
                txtVozik2b.setEditable(false);
                txtVozik3.setEditable(false);
                txtVozik3b.setEditable(false);
                txtVozik4.setEditable(false);
                txtVozik4b.setEditable(false);
                txtVozik5.setEditable(false);
                txtVozik5b.setEditable(false);
                break;
            case 2:
                txtVozik1.setEditable(true);
                txtVozik1b.setEditable(true);
                txtVozik2.setEditable(true);
                txtVozik2b.setEditable(true);
                txtVozik3.setEditable(false);
                txtVozik3b.setEditable(false);
                txtVozik4.setEditable(false);
                txtVozik4b.setEditable(false);
                txtVozik5.setEditable(false);
                txtVozik5b.setEditable(false);
                break;
            case 3:
                txtVozik1.setEditable(true);
                txtVozik1b.setEditable(true);
                txtVozik2.setEditable(true);
                txtVozik2b.setEditable(true);
                txtVozik3.setEditable(true);
                txtVozik3b.setEditable(true);
                txtVozik4.setEditable(false);
                txtVozik4b.setEditable(false);
                txtVozik5.setEditable(false);
                txtVozik5b.setEditable(false);
                break;
            case 4:
                txtVozik1.setEditable(true);
                txtVozik1b.setEditable(true);
                txtVozik2.setEditable(true);
                txtVozik2b.setEditable(true);
                txtVozik3.setEditable(true);
                txtVozik3b.setEditable(true);
                txtVozik4.setEditable(true);
                txtVozik4b.setEditable(true);
                txtVozik5.setEditable(false);
                txtVozik5b.setEditable(false);
                break;
            case 5:
                txtVozik1.setEditable(true);
                txtVozik1b.setEditable(true);
                txtVozik2.setEditable(true);
                txtVozik2b.setEditable(true);
                txtVozik3.setEditable(true);
                txtVozik3b.setEditable(true);
                txtVozik4.setEditable(true);
                txtVozik4b.setEditable(true);
                txtVozik5.setEditable(true);
                txtVozik5b.setEditable(true);
                break;
```

```
                }
    });

        TextField txtSkupinaA = new TextField();
        grid.add(txtSkupinaA,1,11);
        TextField txtSkupinaB = new TextField();
        grid.add(txtSkupinaB,2,11);
        Label lblEntitaA = new Label("Prvni entita");
        grid.add(lblEntitaA,1,10);
        TextField txtEntitaA = new TextField();
        grid.add(txtEntitaA,1,12);
        Label lblSkupina = new Label("Jejich skupina");
        grid.add(lblSkupina,0,11);
        Label lblEntitaApocet = new Label("Jejich Nazev");
        grid.add(lblEntitaApocet,0,12);
        Label lblNazev = new Label("Jejich pocet");
        grid.add(lblNazev,0,13);
        final ComboBox EntitaABox = new ComboBox();
        EntitaABox.getItems().addAll(
            0,
            1,
            2,
            3,
            4,
            5
        );
        grid.add(EntitaABox,1,13);
        Label lblEntitaA2 = new Label("Vyber vozik");
        grid.add(lblEntitaA2,0,14);
        final ComboBox EntitaA2Box = new ComboBox();
        EntitaA2Box.getItems().addAll(
            0,
            1,
            2,
            3,
            4
        );
        grid.add(EntitaA2Box,1,14);

        Label lblEntitaB = new Label("Druha entita");
        grid.add(lblEntitaB,2,10);
        TextField txtEntitaB = new TextField();
        grid.add(txtEntitaB,2,12);
        final ComboBox EntitaBBox = new ComboBox();
        EntitaBBox.getItems().addAll(
            0,
            1,
            2,
            3,
            4,
            5
        );
        grid.add(EntitaBBox,2,13);
        final ComboBox EntitaB2Box = new ComboBox();
        EntitaB2Box.getItems().addAll(
            0,
            1,
            2,
            3,
            4
        );
        grid.add(EntitaB2Box,2,14);

         btn1.setOnAction(action -> {

            String     Init=txtInit.getText();
            Init = Init+".txt";
            String     Method=txtMethod.getText();
            Method = Method+".txt";
            String     Method2=txtMethod2.getText();
            Method2 = Method2+".txt";

            try {
                File myObj = new File(Init);
```

```
    if (myObj.createNewFile()) {
      System.out.println("File created: " + myObj.getName());
    } else {
      System.out.println("File already exists.");
    }
  } catch (IOException e) {
    System.out.println("An error occurred.");
    e.printStackTrace();
  }

try {
    File myObj = new File(Method);
    if (myObj.createNewFile()) {
      System.out.println("File created: " + myObj.getName());
    } else {
      System.out.println("File already exists.");
    }
  } catch (IOException e) {
    System.out.println("An error occurred.");
    e.printStackTrace();
  }

try {
    File myObj = new File(Method2);
    if (myObj.createNewFile()) {
      System.out.println("File created: " + myObj.getName());
    } else {
      System.out.println("File already exists.");
    }
  } catch (IOException e) {
    System.out.println("An error occurred.");
    e.printStackTrace();
  }
try {
    FileWriter myWriter1 = new FileWriter(Init);
    FileWriter myWriter2 = new FileWriter(Method);
    FileWriter myWriter3 = new FileWriter(Method2);



    switch (VN) {
    case 1:
       myWriter1.write(".MUs."+txtVozik1+".create(Track, "+txtVozik1b+")");
    myWriter1.write(System.getProperty("line.separator"));
        break;
    case 2:
       myWriter1.write(".MUs."+txtVozik1+".create(Track, "+txtVozik1b+")");
    myWriter1.write(System.getProperty("line.separator"));
   myWriter1.write(".MUs."+txtVozik2+".create(Track, "+txtVozik2b+")");
    myWriter1.write(System.getProperty("line.separator"));
        break;
    case 3:
       myWriter1.write(".MUs."+txtVozik1+".create(Track, "+txtVozik1b+")");
    myWriter1.write(System.getProperty("line.separator"));
   myWriter1.write(".MUs."+txtVozik2+".create(Track, "+txtVozik2b+")");
    myWriter1.write(System.getProperty("line.separator"));
   myWriter1.write(".MUs."+txtVozik3+".create(Track, "+txtVozik3b+")");
    myWriter1.write(System.getProperty("line.separator"));
        break;
    case 4:
       myWriter1.write(".MUs."+txtVozik1+".create(Track, "+txtVozik1b+")");
    myWriter1.write(System.getProperty("line.separator"));
   myWriter1.write(".MUs."+txtVozik2+".create(Track, "+txtVozik2b+")");
    myWriter1.write(System.getProperty("line.separator"));
   myWriter1.write(".MUs."+txtVozik3+".create(Track, "+txtVozik3b+")");
    myWriter1.write(System.getProperty("line.separator"));
   myWriter1.write(".MUs."+txtVozik4+".create(Track, "+txtVozik4b+")");
    myWriter1.write(System.getProperty("line.separator"));
        break;
    case 5:
       myWriter1.write(".MUs."+txtVozik1+".create(Track, "+txtVozik1b+")");
    myWriter1.write(System.getProperty("line.separator"));
   myWriter1.write(".MUs."+txtVozik2+".create(Track, "+txtVozik2b+")");
    myWriter1.write(System.getProperty("line.separator"));
```

```
            myWriter1.write(".MUs."+txtVozik3+".create(Track, "+txtVozik3b+")");
             myWriter1.write(System.getProperty("line.separator"));
            myWriter1.write(".MUs."+txtVozik4+".create(Track, "+txtVozik4b+")");
             myWriter1.write(System.getProperty("line.separator"));
            myWriter1.write(".MUs."+txtVozik5+".create(Track, "+txtVozik5b+")");
             myWriter1.write(System.getProperty("line.separator"));
                 break;
                 }
             String     ABox=EntitaABox.getValue().toString();
             int ABoxNumber=Integer.parseInt(ABox);

             String     BBox=EntitaBBox.getValue().toString();
             int BBoxNumber=Integer.parseInt(BBox);

             for (int i=0; i < ABoxNumber; i++)
             {
             myWriter1.write(".MUs."+txtSkupinaA+".create("+lblEntitaA+")");
             myWriter1.write(System.getProperty("line.separator"));
             }

             for (int i=0; i < BBoxNumber; i++)
             {
             myWriter1.write(".MUs."+txtSkupinaA+".create("+lblEntitaA+")");
             myWriter1.write(System.getProperty("line.separator"));
             }

             String     Pozice1=EntitaA2Box.getValue().toString();
             int PoziceNum1=Integer.parseInt(Pozice1);
             String     Pozice2=EntitaB2Box.getValue().toString();
             int PoziceNum2=Integer.parseInt(Pozice2);

             myWriter2.write("if @.name = \""+txtVozik1+"\"");
             myWriter2.write(System.getProperty("line.separator"));
             myWriter2.write("@.stopped := true");
             myWriter2.write(System.getProperty("line.separator"));
             myWriter2.write("waint("+PocetBox+")");
             myWriter2.write(System.getProperty("line.separator"));
             myWriter2.write(lblEntitaA+".cont.move(.MUs."+txtVozik2+":"+PoziceNum1+")");
             myWriter2.write(System.getProperty("line.separator"));
             myWriter2.write("waint("+PocetBox+")");
             myWriter2.write(System.getProperty("line.separator"));
             myWriter2.write(lblEntitaA+".cont.move(.MUs."+txtVozik2+":"+PoziceNum2+")");
             myWriter2.write(System.getProperty("line.separator"));
             myWriter2.write("@.stopped := false");
             myWriter2.write(System.getProperty("line.separator"));
             myWriter2.write("end");

             myWriter3.write("if @.name = \""+txtVozik1+"\"");
             myWriter3.write(System.getProperty("line.separator"));
             myWriter3.write("@.stopped := true");
             myWriter3.write(System.getProperty("line.separator"));
             myWriter2.write("waint("+PocetBox+")");
             myWriter3.write(System.getProperty("line.separator"));
             myWriter3.write("MUs."+txtVozik2+":"+PoziceNum1+".cont.move("+txtNaklad+")");
             myWriter2.write(System.getProperty("line.separator"));
             myWriter2.write("waint("+PocetBox+")");
             myWriter3.write(System.getProperty("line.separator"));
             myWriter3.write("MUs."+txtVozik2+":"+PoziceNum2+".cont.move("+txtPreklad+")");
             myWriter3.write(System.getProperty("line.separator"));
             myWriter3.write("@.stopped := false");
             myWriter3.write(System.getProperty("line.separator"));
             myWriter3.write("end");

             myWriter1.close();
             myWriter2.close();
             myWriter3.close();

           } catch (IOException e) {
             System.out.println("An error occurred.");
             e.printStackTrace();
           }

});
```

```
        Scene scene=new Scene(grid,600,pocet);
        primaryStage.setTitle("BakalarAplcation3");
        primaryStage.setScene(scene);
        primaryStage.show();
    }
    public static void main (String[] args)
    {
        launch(args);
    }

}
```

Zdrojový kód čtvrtého typového příkladu

```
import javafx.geometry.Insets;
import java.io.File;
import java.io.FileWriter;
import java.io.IOException;
import javafx.application.Application;
import javafx.beans.value.ChangeListener;
import javafx.beans.value.ObservableValue;
import javafx.event.ActionEvent;
import javafx.event.EventHandler;
import javafx.geometry.Pos;
import javafx.scene.Scene;
import javafx.scene.control.*;
import javafx.scene.layout.GridPane;
import javafx.scene.control.Alert;
import javafx.scene.control.Alert.AlertType;
import javafx.stage.DirectoryChooser;

import javafx.stage.Stage;

public class Main extends Application{
        Stage window;
        String Pocet;
        int PocetNumber = 0;
        String Chyba = ("Objevily se tihle problémy:");

        private void showAlertWithHeaderText() {
                Alert alert = new Alert(AlertType.INFORMATION);
                alert.setTitle("Chybové hlášení");
                alert.setHeaderText("Objevily se tihle problémy:");
                alert.setContentText(Chyba);

                alert.showAndWait();
        }

    public void start(Stage primaryStage) throws Exception {
        GridPane grid =new GridPane();
        grid.setAlignment(Pos.CENTER);
        grid.setVgap(10);
        grid.setHgap(10);
        grid.setPadding(new Insets(10));


        Label lblCreatTxt = new Label("Nazev sensor_creation txt");
        grid.add(lblCreatTxt,0,0);
       Label lblSetTxt = new Label("Nazev attribute_setting txt");
        grid.add(lblSetTxt,0,1);
       Label lblColorTxt = new Label("Nazev color_setting txt");
        grid.add(lblColorTxt,2,0);
        Label lblIdTxt = new Label("Nazev sensor_identification txt");
        grid.add(lblIdTxt,2,1);

        TextField CreateTxt = new TextField();
        grid.add(CreateTxt,1,0);
        TextField SetTxt = new TextField();
        grid.add(SetTxt,1,1);
        TextField ColorTxt = new TextField();
        grid.add(ColorTxt,3,0);
        TextField IdTxt = new TextField();
        grid.add(IdTxt,3,1);

        Button btn=new Button("Tisk");
        final ComboBox WaitBox = new ComboBox();
        WaitBox.getItems().addAll(
            0,
            1,
            2,
            3,
            4,
            5,
```

```
            6,
            7,
            8,
            9,
            10
        );
        final ComboBox PocetBox = new ComboBox();
        PocetBox.getItems().addAll(
            0,
            1,
            2,
            3,
            4,
            5,
            6,
            7
        );

        Label lblWait = new Label("Doba čekání");
        grid.add(lblWait,0,2);
        grid.add(WaitBox,1,2);
        WaitBox.getSelectionModel().selectFirst();
        Label lblPocet = new Label("Počet materiálů");
        grid.add(lblPocet,2,2);
        grid.add(PocetBox,3,2);
        PocetBox.getSelectionModel().selectFirst();
        grid.add(btn,0,3);

        Label lblMaterialNazev = new Label("Název materiálu");
        grid.add(lblMaterialNazev,1,3);
        Label lblMaterialRed = new Label("Červená");
        grid.add(lblMaterialRed,2,3);
        Label lblMaterialGreen = new Label("Zelená");
        grid.add(lblMaterialGreen,3,3);
        Label lblMaterialBlue = new Label("Modrá");
        grid.add(lblMaterialBlue,4,3);

        Label lblMaterial1 = new Label("1.Materiál");
        grid.add(lblMaterial1,0,4);
        Label lblMaterial2 = new Label("2.Materiál");
        grid.add(lblMaterial2,0,5);
        Label lblMaterial3 = new Label("3.Materiál");
        grid.add(lblMaterial3,0,6);
        Label lblMaterial4 = new Label("4.Materiál");
        grid.add(lblMaterial4,0,7);
        Label lblMaterial5 = new Label("5.Materiál");
        grid.add(lblMaterial5,0,8);
        Label lblMaterial6 = new Label("6.Materiál");
        grid.add(lblMaterial6,0,9);
        Label lblMaterial7 = new Label("7.Materiál");
        grid.add(lblMaterial7,0,10);

        lblMaterial1.setVisible(false);
        lblMaterial2.setVisible(false);
        lblMaterial3.setVisible(false);
        lblMaterial4.setVisible(false);
        lblMaterial5.setVisible(false);
        lblMaterial6.setVisible(false);
        lblMaterial7.setVisible(false);

        TextField txtMaterial1 = new TextField();
        grid.add(txtMaterial1,1,4);
        TextField txtMaterial2 = new TextField();
        grid.add(txtMaterial2,1,5);
        TextField txtMaterial3 = new TextField();
        grid.add(txtMaterial3,1,6);
        TextField txtMaterial4 = new TextField();
        grid.add(txtMaterial4,1,7);
        TextField txtMaterial5 = new TextField();
        grid.add(txtMaterial5,1,8);
        TextField txtMaterial6 = new TextField();
        grid.add(txtMaterial6,1,9);
        TextField txtMaterial7 = new TextField();
        grid.add(txtMaterial7,1,10);
```

```
txtMaterial1.setVisible(false);
txtMaterial2.setVisible(false);
txtMaterial3.setVisible(false);
txtMaterial4.setVisible(false);
txtMaterial5.setVisible(false);
txtMaterial6.setVisible(false);
txtMaterial7.setVisible(false);

TextField txtRed1 = new TextField();
grid.add(txtRed1,2,4);
TextField txtRed2 = new TextField();
grid.add(txtRed2,2,5);
TextField txtRed3 = new TextField();
grid.add(txtRed3,2,6);
TextField txtRed4 = new TextField();
grid.add(txtRed4,2,7);
TextField txtRed5 = new TextField();
grid.add(txtRed5,2,8);
TextField txtRed6 = new TextField();
grid.add(txtRed6,2,9);
TextField txtRed7 = new TextField();
grid.add(txtRed7,2,10);

txtRed1.setVisible(false);
txtRed2.setVisible(false);
txtRed3.setVisible(false);
txtRed4.setVisible(false);
txtRed5.setVisible(false);
txtRed6.setVisible(false);
txtRed7.setVisible(false);

TextField txtGreen1 = new TextField();
grid.add(txtGreen1,3,4);
TextField txtGreen2 = new TextField();
grid.add(txtGreen2,3,5);
TextField txtGreen3 = new TextField();
grid.add(txtGreen3,3,6);
TextField txtGreen4 = new TextField();
grid.add(txtGreen4,3,7);
TextField txtGreen5 = new TextField();
grid.add(txtGreen5,3,8);
TextField txtGreen6 = new TextField();
grid.add(txtGreen6,3,9);
TextField txtGreen7 = new TextField();
grid.add(txtGreen7,3,10);

txtGreen1.setVisible(false);
txtGreen2.setVisible(false);
txtGreen3.setVisible(false);
txtGreen4.setVisible(false);
txtGreen5.setVisible(false);
txtGreen6.setVisible(false);
txtGreen7.setVisible(false);

TextField txtBlue1 = new TextField();
grid.add(txtBlue1,4,4);
TextField txtBlue2 = new TextField();
grid.add(txtBlue2,4,5);
TextField txtBlue3 = new TextField();
grid.add(txtBlue3,4,6);
TextField txtBlue4 = new TextField();
grid.add(txtBlue4,4,7);
TextField txtBlue5 = new TextField();
grid.add(txtBlue5,4,8);
TextField txtBlue6 = new TextField();
grid.add(txtBlue6,4,9);
TextField txtBlue7 = new TextField();
grid.add(txtBlue7,4,10);

txtBlue1.setVisible(false);
txtBlue2.setVisible(false);
txtBlue3.setVisible(false);
txtBlue4.setVisible(false);
```

```
txtBlue5.setVisible(false);
txtBlue6.setVisible(false);
txtBlue7.setVisible(false);

ChangeListener<String> Listener = new ChangeListener<String>() {
    @Override
    public void changed(ObservableValue<? extends String> observable, String oldValue,
        String newValue) {
        if (!newValue.matches("\\d*")) {
        txtRed1.setText(newValue.replaceAll("[^\\d]", ""));
        }
        if (!newValue.matches("\\d*")) {
        txtRed2.setText(newValue.replaceAll("[^\\d]", ""));
        }
        if (!newValue.matches("\\d*")) {
        txtRed3.setText(newValue.replaceAll("[^\\d]", ""));
        }
        if (!newValue.matches("\\d*")) {
        txtRed4.setText(newValue.replaceAll("[^\\d]", ""));
        }
        if (!newValue.matches("\\d*")) {
        txtRed5.setText(newValue.replaceAll("[^\\d]", ""));
        }
        if (!newValue.matches("\\d*")) {
        txtRed6.setText(newValue.replaceAll("[^\\d]", ""));
        }
        if (!newValue.matches("\\d*")) {
        txtRed7.setText(newValue.replaceAll("[^\\d]", ""));
        }
        if (!newValue.matches("\\d*")) {
        txtGreen1.setText(newValue.replaceAll("[^\\d]", ""));
        }
        if (!newValue.matches("\\d*")) {
        txtGreen2.setText(newValue.replaceAll("[^\\d]", ""));
        }
        if (!newValue.matches("\\d*")) {
        txtGreen3.setText(newValue.replaceAll("[^\\d]", ""));
        }
        if (!newValue.matches("\\d*")) {
        txtGreen4.setText(newValue.replaceAll("[^\\d]", ""));
        }
        if (!newValue.matches("\\d*")) {
        txtGreen5.setText(newValue.replaceAll("[^\\d]", ""));
        }
        if (!newValue.matches("\\d*")) {
        txtGreen6.setText(newValue.replaceAll("[^\\d]", ""));
        }
        if (!newValue.matches("\\d*")) {
        txtGreen7.setText(newValue.replaceAll("[^\\d]", ""));
        }
        if (!newValue.matches("\\d*")) {
        txtBlue1.setText(newValue.replaceAll("[^\\d]", ""));
        }
        if (!newValue.matches("\\d*")) {
        txtBlue2.setText(newValue.replaceAll("[^\\d]", ""));
        }
        if (!newValue.matches("\\d*")) {
        txtBlue3.setText(newValue.replaceAll("[^\\d]", ""));
        }
        if (!newValue.matches("\\d*")) {
        txtBlue4.setText(newValue.replaceAll("[^\\d]", ""));
        }
        if (!newValue.matches("\\d*")) {
        txtBlue5.setText(newValue.replaceAll("[^\\d]", ""));
        }
        if (!newValue.matches("\\d*")) {
        txtBlue6.setText(newValue.replaceAll("[^\\d]", ""));
        }
        if (!newValue.matches("\\d*")) {
        txtBlue7.setText(newValue.replaceAll("[^\\d]", ""));
        }
    }
};
txtRed1.textProperty().addListener(Listener);
```

```
        txtRed2.textProperty().addListener(Listener);
        txtRed3.textProperty().addListener(Listener);
        txtRed4.textProperty().addListener(Listener);
        txtRed5.textProperty().addListener(Listener);
        txtRed6.textProperty().addListener(Listener);
        txtRed7.textProperty().addListener(Listener);

        txtGreen1.textProperty().addListener(Listener);
        txtGreen2.textProperty().addListener(Listener);
        txtGreen3.textProperty().addListener(Listener);
        txtGreen4.textProperty().addListener(Listener);
        txtGreen5.textProperty().addListener(Listener);
        txtGreen6.textProperty().addListener(Listener);
        txtGreen7.textProperty().addListener(Listener);

        txtBlue1.textProperty().addListener(Listener);
        txtBlue2.textProperty().addListener(Listener);
        txtBlue3.textProperty().addListener(Listener);
        txtBlue4.textProperty().addListener(Listener);
        txtBlue5.textProperty().addListener(Listener);
        txtBlue6.textProperty().addListener(Listener);
        txtBlue7.textProperty().addListener(Listener);


        PocetBox.setOnAction(action -> {

            //   Combo=PocetBox.getValue();
            Pocet=PocetBox.getValue().toString();
             PocetNumber=Integer.parseInt(Pocet);

             switch (PocetNumber) {
             case 0:
                 lblMaterial1.setVisible(false);
                 lblMaterial2.setVisible(false);
                 lblMaterial3.setVisible(false);
                 lblMaterial4.setVisible(false);
                 lblMaterial5.setVisible(false);
                 lblMaterial6.setVisible(false);
                 lblMaterial7.setVisible(false);

                 txtMaterial1.setVisible(false);
                 txtMaterial2.setVisible(false);
                 txtMaterial3.setVisible(false);
                 txtMaterial4.setVisible(false);
                 txtMaterial5.setVisible(false);
                 txtMaterial6.setVisible(false);
                 txtMaterial7.setVisible(false);

                 txtRed1.setVisible(false);
                 txtRed2.setVisible(false);
                 txtRed3.setVisible(false);
                 txtRed4.setVisible(false);
                 txtRed5.setVisible(false);
                 txtRed6.setVisible(false);
                 txtRed7.setVisible(false);

                 txtGreen1.setVisible(false);
                 txtGreen2.setVisible(false);
                 txtGreen3.setVisible(false);
                 txtGreen4.setVisible(false);
                 txtGreen5.setVisible(false);
                 txtGreen6.setVisible(false);
                 txtGreen7.setVisible(false);

                 txtBlue1.setVisible(false);
                 txtBlue2.setVisible(false);
                 txtBlue3.setVisible(false);
                 txtBlue4.setVisible(false);
                 txtBlue5.setVisible(false);
                 txtBlue6.setVisible(false);
                 txtBlue7.setVisible(false);
                 break;
             case 1:
                 lblMaterial1.setVisible(true);
```

```
                lblMaterial2.setVisible(false);
                lblMaterial3.setVisible(false);
                lblMaterial4.setVisible(false);
                lblMaterial5.setVisible(false);
                lblMaterial6.setVisible(false);
                lblMaterial7.setVisible(false);

                txtMaterial1.setVisible(true);
                txtMaterial2.setVisible(false);
                txtMaterial3.setVisible(false);
                txtMaterial4.setVisible(false);
                txtMaterial5.setVisible(false);
                txtMaterial6.setVisible(false);
                txtMaterial7.setVisible(false);

                txtRed1.setVisible(true);
                txtRed2.setVisible(false);
                txtRed3.setVisible(false);
                txtRed4.setVisible(false);
                txtRed5.setVisible(false);
                txtRed6.setVisible(false);
                txtRed7.setVisible(false);

                txtGreen1.setVisible(true);
                txtGreen2.setVisible(false);
                txtGreen3.setVisible(false);
                txtGreen4.setVisible(false);
                txtGreen5.setVisible(false);
                txtGreen6.setVisible(false);
                txtGreen7.setVisible(false);

                txtBlue1.setVisible(true);
                txtBlue2.setVisible(false);
                txtBlue3.setVisible(false);
                txtBlue4.setVisible(false);
                txtBlue5.setVisible(false);
                txtBlue6.setVisible(false);
                txtBlue7.setVisible(false);
                break;
            case 2:
                lblMaterial1.setVisible(true);
                lblMaterial2.setVisible(true);
                lblMaterial3.setVisible(false);
                lblMaterial4.setVisible(false);
                lblMaterial5.setVisible(false);
                lblMaterial6.setVisible(false);
                lblMaterial7.setVisible(false);

                txtMaterial1.setVisible(true);
                txtMaterial2.setVisible(true);
                txtMaterial3.setVisible(false);
                txtMaterial4.setVisible(false);
                txtMaterial5.setVisible(false);
                txtMaterial6.setVisible(false);
                txtMaterial7.setVisible(false);

                txtRed1.setVisible(true);
                txtRed2.setVisible(true);
                txtRed3.setVisible(false);
                txtRed4.setVisible(false);
                txtRed5.setVisible(false);
                txtRed6.setVisible(false);
                txtRed7.setVisible(false);

                txtGreen1.setVisible(true);
                txtGreen2.setVisible(true);
                txtGreen3.setVisible(false);
                txtGreen4.setVisible(false);
                txtGreen5.setVisible(false);
                txtGreen6.setVisible(false);
                txtGreen7.setVisible(false);

                txtBlue1.setVisible(true);
                txtBlue2.setVisible(true);
```

```
                txtBlue3.setVisible(false);
                txtBlue4.setVisible(false);
                txtBlue5.setVisible(false);
                txtBlue6.setVisible(false);
                txtBlue7.setVisible(false);
                break;
            case 3:
                lblMaterial1.setVisible(true);
                lblMaterial2.setVisible(true);
                lblMaterial3.setVisible(true);
                lblMaterial4.setVisible(false);
                lblMaterial5.setVisible(false);
                lblMaterial6.setVisible(false);
                lblMaterial7.setVisible(false);

                txtMaterial1.setVisible(true);
                txtMaterial2.setVisible(true);
                txtMaterial3.setVisible(true);
                txtMaterial4.setVisible(false);
                txtMaterial5.setVisible(false);
                txtMaterial6.setVisible(false);
                txtMaterial7.setVisible(false);

                txtRed1.setVisible(true);
                txtRed2.setVisible(true);
                txtRed3.setVisible(true);
                txtRed4.setVisible(false);
                txtRed5.setVisible(false);
                txtRed6.setVisible(false);
                txtRed7.setVisible(false);

                txtGreen1.setVisible(true);
                txtGreen2.setVisible(true);
                txtGreen3.setVisible(true);
                txtGreen4.setVisible(false);
                txtGreen5.setVisible(false);
                txtGreen6.setVisible(false);
                txtGreen7.setVisible(false);

                txtBlue1.setVisible(true);
                txtBlue2.setVisible(true);
                txtBlue3.setVisible(true);
                txtBlue4.setVisible(false);
                txtBlue5.setVisible(false);
                txtBlue6.setVisible(false);
                txtBlue7.setVisible(false);
                break;
            case 4:
                lblMaterial1.setVisible(true);
                lblMaterial2.setVisible(true);
                lblMaterial3.setVisible(true);
                lblMaterial4.setVisible(true);
                lblMaterial5.setVisible(false);
                lblMaterial6.setVisible(false);
                lblMaterial7.setVisible(false);

                txtMaterial1.setVisible(true);
                txtMaterial2.setVisible(true);
                txtMaterial3.setVisible(true);
                txtMaterial4.setVisible(true);
                txtMaterial5.setVisible(false);
                txtMaterial6.setVisible(false);
                txtMaterial7.setVisible(false);

                txtRed1.setVisible(true);
                txtRed2.setVisible(true);
                txtRed3.setVisible(true);
                txtRed4.setVisible(true);
                txtRed5.setVisible(false);
                txtRed6.setVisible(false);
                txtRed7.setVisible(false);

                txtGreen1.setVisible(true);
                txtGreen2.setVisible(true);
```

```
                txtGreen3.setVisible(true);
                txtGreen4.setVisible(true);
                txtGreen5.setVisible(false);
                txtGreen6.setVisible(false);
                txtGreen7.setVisible(false);

                txtBlue1.setVisible(true);
                txtBlue2.setVisible(true);
                txtBlue3.setVisible(true);
                txtBlue4.setVisible(true);
                txtBlue5.setVisible(false);
                txtBlue6.setVisible(false);
                txtBlue7.setVisible(false);
                break;
            case 5:
                lblMaterial1.setVisible(true);
                lblMaterial2.setVisible(true);
                lblMaterial3.setVisible(true);
                lblMaterial4.setVisible(true);
                lblMaterial5.setVisible(true);
                lblMaterial6.setVisible(false);
                lblMaterial7.setVisible(false);

                txtMaterial1.setVisible(true);
                txtMaterial2.setVisible(true);
                txtMaterial3.setVisible(true);
                txtMaterial4.setVisible(true);
                txtMaterial5.setVisible(true);
                txtMaterial6.setVisible(false);
                txtMaterial7.setVisible(false);

                txtRed1.setVisible(true);
                txtRed2.setVisible(true);
                txtRed3.setVisible(true);
                txtRed4.setVisible(true);
                txtRed5.setVisible(true);
                txtRed6.setVisible(false);
                txtRed7.setVisible(false);

                txtGreen1.setVisible(true);
                txtGreen2.setVisible(true);
                txtGreen3.setVisible(true);
                txtGreen4.setVisible(true);
                txtGreen5.setVisible(true);
                txtGreen6.setVisible(false);
                txtGreen7.setVisible(false);

                txtBlue1.setVisible(true);
                txtBlue2.setVisible(true);
                txtBlue3.setVisible(true);
                txtBlue4.setVisible(true);
                txtBlue5.setVisible(true);
                txtBlue6.setVisible(false);
                txtBlue7.setVisible(false);
                break;
            case 6:
                lblMaterial1.setVisible(true);
                lblMaterial2.setVisible(true);
                lblMaterial3.setVisible(true);
                lblMaterial4.setVisible(true);
                lblMaterial5.setVisible(true);
                lblMaterial6.setVisible(true);
                lblMaterial7.setVisible(false);

                txtMaterial1.setVisible(true);
                txtMaterial2.setVisible(true);
                txtMaterial3.setVisible(true);
                txtMaterial4.setVisible(true);
                txtMaterial5.setVisible(true);
                txtMaterial6.setVisible(true);
                txtMaterial7.setVisible(false);

                txtRed1.setVisible(true);
                txtRed2.setVisible(true);
```

```
                txtRed3.setVisible(true);
                txtRed4.setVisible(true);
                txtRed5.setVisible(true);
                txtRed6.setVisible(true);
                txtRed7.setVisible(false);

                txtGreen1.setVisible(true);
                txtGreen2.setVisible(true);
                txtGreen3.setVisible(true);
                txtGreen4.setVisible(true);
                txtGreen5.setVisible(true);
                txtGreen6.setVisible(true);
                txtGreen7.setVisible(false);

                txtBlue1.setVisible(true);
                txtBlue2.setVisible(true);
                txtBlue3.setVisible(true);
                txtBlue4.setVisible(true);
                txtBlue5.setVisible(true);
                txtBlue6.setVisible(true);
                txtBlue7.setVisible(false);
                break;
            case 7:
                lblMaterial1.setVisible(true);
                lblMaterial2.setVisible(true);
                lblMaterial3.setVisible(true);
                lblMaterial4.setVisible(true);
                lblMaterial5.setVisible(true);
                lblMaterial6.setVisible(true);
                lblMaterial7.setVisible(true);

                txtMaterial1.setVisible(true);
                txtMaterial2.setVisible(true);
                txtMaterial3.setVisible(true);
                txtMaterial4.setVisible(true);
                txtMaterial5.setVisible(true);
                txtMaterial6.setVisible(true);
                txtMaterial7.setVisible(true);

                txtRed1.setVisible(true);
                txtRed2.setVisible(true);
                txtRed3.setVisible(true);
                txtRed4.setVisible(true);
                txtRed5.setVisible(true);
                txtRed6.setVisible(true);
                txtRed7.setVisible(true);

                txtGreen1.setVisible(true);
                txtGreen2.setVisible(true);
                txtGreen3.setVisible(true);
                txtGreen4.setVisible(true);
                txtGreen5.setVisible(true);
                txtGreen6.setVisible(true);
                txtGreen7.setVisible(true);

                txtBlue1.setVisible(true);
                txtBlue2.setVisible(true);
                txtBlue3.setVisible(true);
                txtBlue4.setVisible(true);
                txtBlue5.setVisible(true);
                txtBlue6.setVisible(true);
                txtBlue7.setVisible(true);
                break;
                }
    });


        Button btn2=new Button("Invisible");
        btn2.setVisible(false);
        grid.add(btn2,0,0);

         btn.setOnAction(action -> {

                int TestPocet;
```

```
                        Chyba = ("Objevily se tihle problémy:");

                if (CreateTxt.getText() == null || CreateTxt.getText().trim().isEmpty()) {
                        Chyba = String.join("\n"
                                , "Chybí zadání názvu sensor_creation txt");
                        }
                if (SetTxt.getText() == null || SetTxt.getText().trim().isEmpty()) {
                        Chyba = String.join("\n"
                                , "Chybí zadání názvu attribute_setting txt");
                        }
                if (ColorTxt.getText() == null || ColorTxt.getText().trim().isEmpty()) {
                        Chyba = String.join("\n"
                                , "Chybí zadání názvu color_setting txt");
                        }
                if (IdTxt.getText() == null || IdTxt.getText().trim().isEmpty()) {
                        Chyba = String.join("\n"
                                , "Chybí zadání názvu sensor_identification txt");
                        }

        switch (PocetNumber) {
        case 0:
                        Chyba = String.join("\n"
                                , "Počet materiálů nesmí být 0");
            break;
        case 1:
                if (txtMaterial1.getText() == null || txtMaterial1.getText().trim().isEmpty()) {
                        Chyba = String.join("\n"
                                , "Chybí zadání názvu 1.materálu");
                        }
                if (txtRed1.getText() == null || txtRed1.getText().trim().isEmpty()) {
                        Chyba = String.join("\n"
                                , "Chybí stanoveni červené barvy v RGB u 1.materiálu");
                        }
                else {
                        TestPocet = Integer.parseInt(txtRed1.getText());

                        if (TestPocet > 255) {
                                Chyba = String.join("\n"
                                        , "Stanoveniá hodnota červené barvy v RGB pro
1.materiál jet větší než 255");
                                }
                }
                if (txtGreen1.getText() == null || txtGreen1.getText().trim().isEmpty()) {
                        Chyba = String.join("\n"
                                , "Chybí stanoveni zelené barvy v RGB u 1.materiálu");
                        }
                else {
                        TestPocet = Integer.parseInt(txtGreen1.getText());

                        if (TestPocet > 255) {
                                Chyba = String.join("\n"
                                        , "Stanoveniá hodnota zelené barvy v RGB pro 1.materiál
jet větší než 255");
                                }
                }
                if (txtBlue1.getText() == null || txtBlue1.getText().trim().isEmpty()) {
                        Chyba = String.join("\n"
                                , "Chybí stanoveni modré barvy v RGB u 1.materiálu");
                        }
                else {
                        TestPocet = Integer.parseInt(txtBlue1.getText());

                        if (TestPocet > 255) {
                                Chyba = String.join("\n"
                                , "Stanoveniá hodnota modré barvy v RGB pro 1.materiál jet
větší než 255");
                                }
                }
            break;
        case 2:
                if (txtMaterial1.getText() == null || txtMaterial1.getText().trim().isEmpty()) {
                        Chyba = String.join("\n"
                                , "Chybí zadání názvu 1.materálu");
                        }
```

```
                if (txtRed1.getText() == null || txtRed1.getText().trim().isEmpty()) {
                        Chyba = String.join("\n"
                                , "Chybí stanoveni červené barvy v RGB u 1.materiálu");
                        }
                else {
                        TestPocet = Integer.parseInt(txtRed1.getText());

                        if (TestPocet > 255) {
                                Chyba = String.join("\n"
                                        , "Stanoveniá hodnota červené barvy v RGB pro
1.materiál jet větší než 255");
                                }
                }
                if (txtGreen1.getText() == null || txtGreen1.getText().trim().isEmpty()) {
                        Chyba = String.join("\n"
                                , "Chybí stanoveni zelené barvy v RGB u 1.materiálu");
                        }
                else {
                        TestPocet = Integer.parseInt(txtGreen1.getText());

                        if (TestPocet > 255) {
                                Chyba = String.join("\n"
                                        , "Stanoveniá hodnota zelené barvy v RGB pro 1.materiál
jet větší než 255");
                                }
                }
                if (txtBlue1.getText() == null || txtBlue1.getText().trim().isEmpty()) {
                        Chyba = String.join("\n"
                                , "Chybí stanoveni modré barvy v RGB u 1.materiálu");
                        }
                else {
                        TestPocet = Integer.parseInt(txtBlue1.getText());

                        if (TestPocet > 255) {
                                Chyba = String.join("\n"
                                , "Stanoveniá hodnota modré barvy v RGB pro 1.materiál jet
větší než 255");
                                }
                }
                if (txtMaterial2.getText() == null || txtMaterial2.getText().trim().isEmpty()) {
                        Chyba = String.join("\n"
                                , "Chybí zadání názvu 2.materálu");
                        }
                if (txtRed2.getText() == null || txtRed2.getText().trim().isEmpty()) {
                        Chyba = String.join("\n"
                                , "Chybí stanoveni červené barvy v RGB u 2.materiálu");
                        }
                else {
                        TestPocet = Integer.parseInt(txtRed2.getText());

                        if (TestPocet > 255) {
                                Chyba = String.join("\n"
                                        , "Stanoveniá hodnota červené barvy v RGB pro
2.materiál jet větší než 255");
                                }
                }
                if (txtGreen2.getText() == null || txtGreen2.getText().trim().isEmpty()) {
                        Chyba = String.join("\n"
                                , "Chybí stanoveni zelené barvy v RGB u 2.materiálu");
                        }
                else {
                        TestPocet = Integer.parseInt(txtGreen2.getText());

                        if (TestPocet > 255) {
                                Chyba = String.join("\n"
                                        , "Stanoveniá hodnota zelené barvy v RGB pro 2.materiál
jet větší než 255");
                                }
                }
                if (txtBlue2.getText() == null || txtBlue2.getText().trim().isEmpty()) {
                        Chyba = String.join("\n"
                                , "Chybí stanoveni modré barvy v RGB u 2.materiálu");
                        }
                else {
```

```
                        TestPocet = Integer.parseInt(txtBlue2.getText());

                        if (TestPocet > 255) {
                                Chyba = String.join("\n"
                                 , "Stanovenià hodnota modré barvy v RGB pro 2.materiál jet
větší než 255");
                                }
                }
            break;
        case 3:
                if (txtMaterial1.getText() == null || txtMaterial1.getText().trim().isEmpty()) {
                        Chyba = String.join("\n"
                                 , "Chybí zadání názvu 1.materálu");
                        }
                if (txtRed1.getText() == null || txtRed1.getText().trim().isEmpty()) {
                        Chyba = String.join("\n"
                                 , "Chybí stanoveni červené barvy v RGB u 1.materiálu");
                        }
                else {
                        TestPocet = Integer.parseInt(txtRed1.getText());

                        if (TestPocet > 255) {
                                Chyba = String.join("\n"
                                         , "Stanovenià hodnota červené barvy v RGB pro
1.materiál jet větší než 255");
                                }
                }
                if (txtGreen1.getText() == null || txtGreen1.getText().trim().isEmpty()) {
                        Chyba = String.join("\n"
                                 , "Chybí stanoveni zelené barvy v RGB u 1.materiálu");
                        }
                else {
                        TestPocet = Integer.parseInt(txtGreen1.getText());

                        if (TestPocet > 255) {
                                Chyba = String.join("\n"
                                         , "Stanovenià hodnota zelené barvy v RGB pro 1.materiál
jet větší než 255");
                                }
                }
                if (txtBlue1.getText() == null || txtBlue1.getText().trim().isEmpty()) {
                        Chyba = String.join("\n"
                                 , "Chybí stanoveni modré barvy v RGB u 1.materiálu");
                        }
                else {
                        TestPocet = Integer.parseInt(txtBlue1.getText());

                        if (TestPocet > 255) {
                                Chyba = String.join("\n"
                                 , "Stanovenià hodnota modré barvy v RGB pro 1.materiál jet
větší než 255");
                                }
                }
                if (txtMaterial2.getText() == null || txtMaterial2.getText().trim().isEmpty()) {
                        Chyba = String.join("\n"
                                 , "Chybí zadání názvu 2.materálu");
                        }
                if (txtRed2.getText() == null || txtRed2.getText().trim().isEmpty()) {
                        Chyba = String.join("\n"
                                 , "Chybí stanoveni červené barvy v RGB u 2.materiálu");
                        }
                else {
                        TestPocet = Integer.parseInt(txtRed2.getText());

                        if (TestPocet > 255) {
                                Chyba = String.join("\n"
                                         , "Stanovenià hodnota červené barvy v RGB pro
2.materiál jet větší než 255");
                                }
                }
                if (txtGreen2.getText() == null || txtGreen2.getText().trim().isEmpty()) {
                        Chyba = String.join("\n"
                                 , "Chybí stanoveni zelené barvy v RGB u 2.materiálu");
                        }
```

```
                else {
                        TestPocet = Integer.parseInt(txtGreen2.getText());

                        if (TestPocet > 255) {
                                Chyba = String.join("\n"
                                        , "Stanoveniá hodnota zelené barvy v RGB pro 2.materiál
jet větší než 255");
                                }
                }
                if (txtBlue2.getText() == null || txtBlue2.getText().trim().isEmpty()) {
                        Chyba = String.join("\n"
                                , "Chybí stanoveni modré barvy v RGB u 2.materiálu");
                        }
                else {
                        TestPocet = Integer.parseInt(txtBlue2.getText());

                        if (TestPocet > 255) {
                                Chyba = String.join("\n"
                                , "Stanoveniá hodnota modré barvy v RGB pro 2.materiál jet
větší než 255");
                                }
                }
                if (txtMaterial3.getText() == null || txtMaterial3.getText().trim().isEmpty()) {
                        Chyba = String.join("\n"
                                , "Chybí zadání názvu 3.materálu");
                        }
                if (txtRed3.getText() == null || txtRed3.getText().trim().isEmpty()) {
                        Chyba = String.join("\n"
                                , "Chybí stanoveni červené barvy v RGB u 3.materiálu");
                        }
                else {
                        TestPocet = Integer.parseInt(txtRed3.getText());

                        if (TestPocet > 255) {
                                Chyba = String.join("\n"
                                        , "Stanoveniá hodnota červené barvy v RGB pro
3.materiál jet větší než 255");
                                }
                }
                if (txtGreen3.getText() == null || txtGreen3.getText().trim().isEmpty()) {
                        Chyba = String.join("\n"
                                , "Chybí stanoveni zelené barvy v RGB u 3.materiálu");
                        }
                else {
                        TestPocet = Integer.parseInt(txtGreen3.getText());

                        if (TestPocet > 255) {
                                Chyba = String.join("\n"
                                        , "Stanoveniá hodnota zelené barvy v RGB pro 3.materiál
jet větší než 255");
                                }
                }
                if (txtBlue3.getText() == null || txtBlue3.getText().trim().isEmpty()) {
                        Chyba = String.join("\n"
                                , "Chybí stanoveni modré barvy v RGB u 3.materiálu");
                        }
                else {
                        TestPocet = Integer.parseInt(txtBlue3.getText());

                        if (TestPocet > 255) {
                                Chyba = String.join("\n"
                                , "Stanoveniá hodnota modré barvy v RGB pro 3.materiál jet
větší než 255");
                                }
                }
            break;
        case 4:
                if (txtMaterial1.getText() == null || txtMaterial1.getText().trim().isEmpty()) {
                        Chyba = String.join("\n"
                                , "Chybí zadání názvu 1.materálu");
                        }
                if (txtRed1.getText() == null || txtRed1.getText().trim().isEmpty()) {
                        Chyba = String.join("\n"
                                , "Chybí stanoveni červené barvy v RGB u 1.materiálu");
```

```
                        }
                else {
                        TestPocet = Integer.parseInt(txtRed1.getText());

                        if (TestPocet > 255) {
                                Chyba = String.join("\n"
                                        , "Stanoveniá hodnota červené barvy v RGB pro
1.materiál jet větší než 255");
                                }
                }
                if (txtGreen1.getText() == null || txtGreen1.getText().trim().isEmpty()) {
                        Chyba = String.join("\n"
                                , "Chybí stanoveni zelené barvy v RGB u 1.materiálu");
                        }
                else {
                        TestPocet = Integer.parseInt(txtGreen1.getText());

                        if (TestPocet > 255) {
                                Chyba = String.join("\n"
                                        , "Stanoveniá hodnota zelené barvy v RGB pro 1.materiál
jet větší než 255");
                                }
                }
                if (txtBlue1.getText() == null || txtBlue1.getText().trim().isEmpty()) {
                        Chyba = String.join("\n"
                                , "Chybí stanoveni modré barvy v RGB u 1.materiálu");
                        }
                else {
                        TestPocet = Integer.parseInt(txtBlue1.getText());

                        if (TestPocet > 255) {
                                Chyba = String.join("\n"
                                , "Stanoveniá hodnota modré barvy v RGB pro 1.materiál jet
větší než 255");
                                }
                }
                if (txtMaterial2.getText() == null || txtMaterial2.getText().trim().isEmpty()) {
                        Chyba = String.join("\n"
                                , "Chybí zadání názvu 2.materálu");
                        }
                if (txtRed2.getText() == null || txtRed2.getText().trim().isEmpty()) {
                        Chyba = String.join("\n"
                                , "Chybí stanoveni červené barvy v RGB u 2.materiálu");
                        }
                else {
                        TestPocet = Integer.parseInt(txtRed2.getText());

                        if (TestPocet > 255) {
                                Chyba = String.join("\n"
                                        , "Stanoveniá hodnota červené barvy v RGB pro
2.materiál jet větší než 255");
                                }
                }
                if (txtGreen2.getText() == null || txtGreen2.getText().trim().isEmpty()) {
                        Chyba = String.join("\n"
                                , "Chybí stanoveni zelené barvy v RGB u 2.materiálu");
                        }
                else {
                        TestPocet = Integer.parseInt(txtGreen2.getText());

                        if (TestPocet > 255) {
                                Chyba = String.join("\n"
                                        , "Stanoveniá hodnota zelené barvy v RGB pro 2.materiál
jet větší než 255");
                                }
                }
                if (txtBlue2.getText() == null || txtBlue2.getText().trim().isEmpty()) {
                        Chyba = String.join("\n"
                                , "Chybí stanoveni modré barvy v RGB u 2.materiálu");
                        }
                else {
                        TestPocet = Integer.parseInt(txtBlue2.getText());

                        if (TestPocet > 255) {
```

```
                                Chyba = String.join("\n"
                                 , "Stanoveniá hodnota modré barvy v RGB pro 2.materiál jet
větší než 255");
                                }
                }
                if (txtMaterial3.getText() == null || txtMaterial3.getText().trim().isEmpty()) {
                        Chyba = String.join("\n"
                                 , "Chybí zadání názvu 3.materálu");
                        }
                if (txtRed3.getText() == null || txtRed3.getText().trim().isEmpty()) {
                        Chyba = String.join("\n"
                                 , "Chybí stanoveni červené barvy v RGB u 3.materiálu");
                        }
                else {
                        TestPocet = Integer.parseInt(txtRed3.getText());

                        if (TestPocet > 255) {
                                Chyba = String.join("\n"
                                         , "Stanoveniá hodnota červené barvy v RGB pro
3.materiál jet větší než 255");
                                }
                }
                if (txtGreen3.getText() == null || txtGreen3.getText().trim().isEmpty()) {
                        Chyba = String.join("\n"
                                 , "Chybí stanoveni zelené barvy v RGB u 3.materiálu");
                        }
                else {
                        TestPocet = Integer.parseInt(txtGreen3.getText());

                        if (TestPocet > 255) {
                                Chyba = String.join("\n"
                                         , "Stanoveniá hodnota zelené barvy v RGB pro 3.materiál
jet větší než 255");
                                }
                }
                if (txtBlue3.getText() == null || txtBlue3.getText().trim().isEmpty()) {
                        Chyba = String.join("\n"
                                 , "Chybí stanoveni modré barvy v RGB u 3.materiálu");
                        }
                else {
                        TestPocet = Integer.parseInt(txtBlue3.getText());

                        if (TestPocet > 255) {
                                Chyba = String.join("\n"
                                 , "Stanoveniá hodnota modré barvy v RGB pro 3.materiál jet
větší než 255");
                                }
                }
                if (txtMaterial4.getText() == null || txtMaterial4.getText().trim().isEmpty()) {
                        Chyba = String.join("\n"
                                 , "Chybí zadání názvu 4.materálu");
                        }
                if (txtRed4.getText() == null || txtRed4.getText().trim().isEmpty()) {
                        Chyba = String.join("\n"
                                 , "Chybí stanoveni červené barvy v RGB u 4.materiálu");
                        }
                else {
                        TestPocet = Integer.parseInt(txtRed4.getText());

                        if (TestPocet > 255) {
                                Chyba = String.join("\n"
                                         , "Stanoveniá hodnota červené barvy v RGB pro
4.materiál jet větší než 255");
                                }
                }
                if (txtGreen4.getText() == null || txtGreen4.getText().trim().isEmpty()) {
                        Chyba = String.join("\n"
                                 , "Chybí stanoveni zelené barvy v RGB u 4.materiálu");
                        }
                else {
                        TestPocet = Integer.parseInt(txtGreen4.getText());

                        if (TestPocet > 255) {
                                Chyba = String.join("\n"
```

```
                                        , "Stanoveniá hodnota zelené barvy v RGB pro 4.materiál
jet větší než 255");
                                }
                        }
                        if (txtBlue4.getText() == null || txtBlue4.getText().trim().isEmpty()) {
                                Chyba = String.join("\n"
                                         , "Chybí stanoveni modré barvy v RGB u 4.materiálu");
                                }
                        else {
                                TestPocet = Integer.parseInt(txtBlue4.getText());

                                if (TestPocet > 255) {
                                        Chyba = String.join("\n"
                                         , "Stanoveniá hodnota modré barvy v RGB pro 4.materiál jet
větší než 255");
                                        }
                        }
                    break;
                case 5:
                        if (txtMaterial1.getText() == null || txtMaterial1.getText().trim().isEmpty()) {
                                Chyba = String.join("\n"
                                         , "Chybí zadání názvu 1.materálu");
                                }
                        if (txtRed1.getText() == null || txtRed1.getText().trim().isEmpty()) {
                                Chyba = String.join("\n"
                                         , "Chybí stanoveni červené barvy v RGB u 1.materiálu");
                                }
                        else {
                                TestPocet = Integer.parseInt(txtRed1.getText());

                                if (TestPocet > 255) {
                                        Chyba = String.join("\n"
                                                 , "Stanoveniá hodnota červené barvy v RGB pro
1.materiál jet větší než 255");
                                        }
                        }
                        if (txtGreen1.getText() == null || txtGreen1.getText().trim().isEmpty()) {
                                Chyba = String.join("\n"
                                         , "Chybí stanoveni zelené barvy v RGB u 1.materiálu");
                                }
                        else {
                                TestPocet = Integer.parseInt(txtGreen1.getText());

                                if (TestPocet > 255) {
                                        Chyba = String.join("\n"
                                                 , "Stanoveniá hodnota zelené barvy v RGB pro 1.materiál
jet větší než 255");
                                        }
                        }
                        if (txtBlue1.getText() == null || txtBlue1.getText().trim().isEmpty()) {
                                Chyba = String.join("\n"
                                         , "Chybí stanoveni modré barvy v RGB u 1.materiálu");
                                }
                        else {
                                TestPocet = Integer.parseInt(txtBlue1.getText());

                                if (TestPocet > 255) {
                                        Chyba = String.join("\n"
                                         , "Stanoveniá hodnota modré barvy v RGB pro 1.materiál jet
větší než 255");
                                        }
                        }
                        if (txtMaterial2.getText() == null || txtMaterial2.getText().trim().isEmpty()) {
                                Chyba = String.join("\n"
                                         , "Chybí zadání názvu 2.materálu");
                                }
                        if (txtRed2.getText() == null || txtRed2.getText().trim().isEmpty()) {
                                Chyba = String.join("\n"
                                         , "Chybí stanoveni červené barvy v RGB u 2.materiálu");
                                }
                        else {
                                TestPocet = Integer.parseInt(txtRed2.getText());

                                if (TestPocet > 255) {
```

```
                                Chyba = String.join("\n"
                                        , "Stanoveniá hodnota červené barvy v RGB pro
2.materiál jet větší než 255");
                                }
                }
                if (txtGreen2.getText() == null || txtGreen2.getText().trim().isEmpty()) {
                        Chyba = String.join("\n"
                                , "Chybí stanoveni zelené barvy v RGB u 2.materiálu");
                        }
                else {
                        TestPocet = Integer.parseInt(txtGreen2.getText());

                        if (TestPocet > 255) {
                                Chyba = String.join("\n"
                                        , "Stanoveniá hodnota zelené barvy v RGB pro 2.materiál
jet větší než 255");
                                }
                }
                if (txtBlue2.getText() == null || txtBlue2.getText().trim().isEmpty()) {
                        Chyba = String.join("\n"
                                , "Chybí stanoveni modré barvy v RGB u 2.materiálu");
                        }
                else {
                        TestPocet = Integer.parseInt(txtBlue2.getText());

                        if (TestPocet > 255) {
                                Chyba = String.join("\n"
                                , "Stanoveniá hodnota modré barvy v RGB pro 2.materiál jet
větší než 255");
                                }
                }
                if (txtMaterial3.getText() == null || txtMaterial3.getText().trim().isEmpty()) {
                        Chyba = String.join("\n"
                                , "Chybí zadání názvu 3.materálu");
                        }
                if (txtRed3.getText() == null || txtRed3.getText().trim().isEmpty()) {
                        Chyba = String.join("\n"
                                , "Chybí stanoveni červené barvy v RGB u 3.materiálu");
                        }
                else {
                        TestPocet = Integer.parseInt(txtRed3.getText());

                        if (TestPocet > 255) {
                                Chyba = String.join("\n"
                                        , "Stanoveniá hodnota červené barvy v RGB pro
3.materiál jet větší než 255");
                                }
                }
                if (txtGreen3.getText() == null || txtGreen3.getText().trim().isEmpty()) {
                        Chyba = String.join("\n"
                                , "Chybí stanoveni zelené barvy v RGB u 3.materiálu");
                        }
                else {
                        TestPocet = Integer.parseInt(txtGreen3.getText());

                        if (TestPocet > 255) {
                                Chyba = String.join("\n"
                                        , "Stanoveniá hodnota zelené barvy v RGB pro 3.materiál
jet větší než 255");
                                }
                }
                if (txtBlue3.getText() == null || txtBlue3.getText().trim().isEmpty()) {
                        Chyba = String.join("\n"
                                , "Chybí stanoveni modré barvy v RGB u 3.materiálu");
                        }
                else {
                        TestPocet = Integer.parseInt(txtBlue3.getText());

                        if (TestPocet > 255) {
                                Chyba = String.join("\n"
                                , "Stanoveniá hodnota modré barvy v RGB pro 3.materiál jet
větší než 255");
                                }
                }
```

```
                if (txtMaterial4.getText() == null || txtMaterial4.getText().trim().isEmpty()) {
                        Chyba = String.join("\n"
                                 , "Chybí zadání názvu 4.materálu");
                        }
                if (txtRed4.getText() == null || txtRed4.getText().trim().isEmpty()) {
                        Chyba = String.join("\n"
                                 , "Chybí stanoveni červené barvy v RGB u 4.materiálu");
                        }
                else {
                        TestPocet = Integer.parseInt(txtRed4.getText());

                        if (TestPocet > 255) {
                                Chyba = String.join("\n"
                                         , "Stanoveniá hodnota červené barvy v RGB pro
4.materiál jet větší než 255");
                                }
                }
                if (txtGreen4.getText() == null || txtGreen4.getText().trim().isEmpty()) {
                        Chyba = String.join("\n"
                                 , "Chybí stanoveni zelené barvy v RGB u 4.materiálu");
                        }
                else {
                        TestPocet = Integer.parseInt(txtGreen4.getText());

                        if (TestPocet > 255) {
                                Chyba = String.join("\n"
                                         , "Stanoveniá hodnota zelené barvy v RGB pro 4.materiál
jet větší než 255");
                                }
                }
                if (txtBlue4.getText() == null || txtBlue4.getText().trim().isEmpty()) {
                        Chyba = String.join("\n"
                                 , "Chybí stanoveni modré barvy v RGB u 4.materiálu");
                        }
                else {
                        TestPocet = Integer.parseInt(txtBlue4.getText());

                        if (TestPocet > 255) {
                                Chyba = String.join("\n"
                                 , "Stanoveniá hodnota modré barvy v RGB pro 4.materiál jet
větší než 255");
                                }
                }
                if (txtMaterial5.getText() == null || txtMaterial5.getText().trim().isEmpty()) {
                        Chyba = String.join("\n"
                                 , "Chybí zadání názvu 5.materálu");
                        }
                if (txtRed5.getText() == null || txtRed5.getText().trim().isEmpty()) {
                        Chyba = String.join("\n"
                                 , "Chybí stanoveni červené barvy v RGB u 5.materiálu");
                        }
                else {
                        TestPocet = Integer.parseInt(txtRed5.getText());

                        if (TestPocet > 255) {
                                Chyba = String.join("\n"
                                         , "Stanoveniá hodnota červené barvy v RGB pro
5.materiál jet větší než 255");
                                }
                }
                if (txtGreen5.getText() == null || txtGreen5.getText().trim().isEmpty()) {
                        Chyba = String.join("\n"
                                 , "Chybí stanoveni zelené barvy v RGB u 5.materiálu");
                        }
                else {
                        TestPocet = Integer.parseInt(txtGreen5.getText());

                        if (TestPocet > 255) {
                                Chyba = String.join("\n"
                                         , "Stanoveniá hodnota zelené barvy v RGB pro 5.materiál
jet větší než 255");
                                }
                }
                if (txtBlue5.getText() == null || txtBlue5.getText().trim().isEmpty()) {
```

```
                        Chyba = String.join("\n"
                                , "Chybí stanoveni modré barvy v RGB u 5.materiálu");
                        }
                else {
                        TestPocet = Integer.parseInt(txtBlue5.getText());

                        if (TestPocet > 255) {
                                Chyba = String.join("\n"
                                , "Stanovenià hodnota modré barvy v RGB pro 5.materiál jet
větší než 255");
                                }
                }
            break;
        case 6:
                if (txtMaterial1.getText() == null || txtMaterial1.getText().trim().isEmpty()) {
                        Chyba = String.join("\n"
                                , "Chybí zadání názvu 1.materálu");
                        }
                if (txtRed1.getText() == null || txtRed1.getText().trim().isEmpty()) {
                        Chyba = String.join("\n"
                                , "Chybí stanoveni červené barvy v RGB u 1.materiálu");
                        }
                else {
                        TestPocet = Integer.parseInt(txtRed1.getText());

                        if (TestPocet > 255) {
                                Chyba = String.join("\n"
                                        , "Stanovenià hodnota červené barvy v RGB pro
1.materiál jet větší než 255");
                                }
                }
                if (txtGreen1.getText() == null || txtGreen1.getText().trim().isEmpty()) {
                        Chyba = String.join("\n"
                                , "Chybí stanoveni zelené barvy v RGB u 1.materiálu");
                        }
                else {
                        TestPocet = Integer.parseInt(txtGreen1.getText());

                        if (TestPocet > 255) {
                                Chyba = String.join("\n"
                                        , "Stanovenià hodnota zelené barvy v RGB pro 1.materiál
jet větší než 255");
                                }
                }
                if (txtBlue1.getText() == null || txtBlue1.getText().trim().isEmpty()) {
                        Chyba = String.join("\n"
                                , "Chybí stanoveni modré barvy v RGB u 1.materiálu");
                        }
                else {
                        TestPocet = Integer.parseInt(txtBlue1.getText());

                        if (TestPocet > 255) {
                                Chyba = String.join("\n"
                                , "Stanovenià hodnota modré barvy v RGB pro 1.materiál jet
větší než 255");
                                }
                }
                if (txtMaterial2.getText() == null || txtMaterial2.getText().trim().isEmpty()) {
                        Chyba = String.join("\n"
                                , "Chybí zadání názvu 2.materálu");
                        }
                if (txtRed2.getText() == null || txtRed2.getText().trim().isEmpty()) {
                        Chyba = String.join("\n"
                                , "Chybí stanoveni červené barvy v RGB u 2.materiálu");
                        }
                else {
                        TestPocet = Integer.parseInt(txtRed2.getText());

                        if (TestPocet > 255) {
                                Chyba = String.join("\n"
                                        , "Stanovenià hodnota červené barvy v RGB pro
2.materiál jet větší než 255");
                                }
                }
```

```
                if (txtGreen2.getText() == null || txtGreen2.getText().trim().isEmpty()) {
                        Chyba = String.join("\n"
                                , "Chybí stanoveni zelené barvy v RGB u 2.materiálu");
                        }
                else {
                        TestPocet = Integer.parseInt(txtGreen2.getText());

                        if (TestPocet > 255) {
                                Chyba = String.join("\n"
                                        , "Stanoveniá hodnota zelené barvy v RGB pro 2.materiál
jet větší než 255");
                                }
                }
                if (txtBlue2.getText() == null || txtBlue2.getText().trim().isEmpty()) {
                        Chyba = String.join("\n"
                                , "Chybí stanoveni modré barvy v RGB u 2.materiálu");
                        }
                else {
                        TestPocet = Integer.parseInt(txtBlue2.getText());

                        if (TestPocet > 255) {
                                Chyba = String.join("\n"
                                , "Stanoveniá hodnota modré barvy v RGB pro 2.materiál jet
větší než 255");
                                }
                }
                if (txtMaterial3.getText() == null || txtMaterial3.getText().trim().isEmpty()) {
                        Chyba = String.join("\n"
                                , "Chybí zadání názvu 3.materálu");
                        }
                if (txtRed3.getText() == null || txtRed3.getText().trim().isEmpty()) {
                        Chyba = String.join("\n"
                                , "Chybí stanoveni červené barvy v RGB u 3.materiálu");
                        }
                else {
                        TestPocet = Integer.parseInt(txtRed3.getText());

                        if (TestPocet > 255) {
                                Chyba = String.join("\n"
                                        , "Stanoveniá hodnota červené barvy v RGB pro
3.materiál jet větší než 255");
                                }
                }
                if (txtGreen3.getText() == null || txtGreen3.getText().trim().isEmpty()) {
                        Chyba = String.join("\n"
                                , "Chybí stanoveni zelené barvy v RGB u 3.materiálu");
                        }
                else {
                        TestPocet = Integer.parseInt(txtGreen3.getText());

                        if (TestPocet > 255) {
                                Chyba = String.join("\n"
                                        , "Stanoveniá hodnota zelené barvy v RGB pro 3.materiál
jet větší než 255");
                                }
                }
                if (txtBlue3.getText() == null || txtBlue3.getText().trim().isEmpty()) {
                        Chyba = String.join("\n"
                                , "Chybí stanoveni modré barvy v RGB u 3.materiálu");
                        }
                else {
                        TestPocet = Integer.parseInt(txtBlue3.getText());

                        if (TestPocet > 255) {
                                Chyba = String.join("\n"
                                , "Stanoveniá hodnota modré barvy v RGB pro 3.materiál jet
větší než 255");
                                }
                }
                if (txtMaterial4.getText() == null || txtMaterial4.getText().trim().isEmpty()) {
                        Chyba = String.join("\n"
                                , "Chybí zadání názvu 4.materálu");
                        }
                if (txtRed4.getText() == null || txtRed4.getText().trim().isEmpty()) {
```

```
                        Chyba = String.join("\n"
                                , "Chybí stanoveni červené barvy v RGB u 4.materiálu");
                        }
                else {
                        TestPocet = Integer.parseInt(txtRed4.getText());

                        if (TestPocet > 255) {
                                Chyba = String.join("\n"
                                        , "Stanoveniá hodnota červené barvy v RGB pro
4.materiál jet větší než 255");
                                }
                }
                if (txtGreen4.getText() == null || txtGreen4.getText().trim().isEmpty()) {
                        Chyba = String.join("\n"
                                , "Chybí stanoveni zelené barvy v RGB u 4.materiálu");
                        }
                else {
                        TestPocet = Integer.parseInt(txtGreen4.getText());

                        if (TestPocet > 255) {
                                Chyba = String.join("\n"
                                        , "Stanoveniá hodnota zelené barvy v RGB pro 4.materiál
jet větší než 255");
                                }
                }
                if (txtBlue4.getText() == null || txtBlue4.getText().trim().isEmpty()) {
                        Chyba = String.join("\n"
                                , "Chybí stanoveni modré barvy v RGB u 4.materiálu");
                        }
                else {
                        TestPocet = Integer.parseInt(txtBlue4.getText());

                        if (TestPocet > 255) {
                                Chyba = String.join("\n"
                                , "Stanoveniá hodnota modré barvy v RGB pro Č.materiál jet
větší než 255");
                                }
                }
                if (txtMaterial5.getText() == null || txtMaterial5.getText().trim().isEmpty()) {
                        Chyba = String.join("\n"
                                , "Chybí zadání názvu 5.materálu");
                        }
                if (txtRed5.getText() == null || txtRed5.getText().trim().isEmpty()) {
                        Chyba = String.join("\n"
                                , "Chybí stanoveni červené barvy v RGB u 5.materiálu");
                        }
                else {
                        TestPocet = Integer.parseInt(txtRed5.getText());

                        if (TestPocet > 255) {
                                Chyba = String.join("\n"
                                        , "Stanoveniá hodnota červené barvy v RGB pro
5.materiál jet větší než 255");
                                }
                }
                if (txtGreen5.getText() == null || txtGreen5.getText().trim().isEmpty()) {
                        Chyba = String.join("\n"
                                , "Chybí stanoveni zelené barvy v RGB u 5.materiálu");
                        }
                else {
                        TestPocet = Integer.parseInt(txtGreen5.getText());

                        if (TestPocet > 255) {
                                Chyba = String.join("\n"
                                        , "Stanoveniá hodnota zelené barvy v RGB pro 5.materiál
jet větší než 255");
                                }
                }
                if (txtBlue5.getText() == null || txtBlue5.getText().trim().isEmpty()) {
                        Chyba = String.join("\n"
                                , "Chybí stanoveni modré barvy v RGB u 5.materiálu");
                        }
                else {
```

```
                        TestPocet = Integer.parseInt(txtBlue5.getText());

                        if (TestPocet > 255) {
                                Chyba = String.join("\n"
                                 , "Stanoveniá hodnota modré barvy v RGB pro 5.materiál jet
větší než 255");
                                }
                }
                if (txtMaterial6.getText() == null || txtMaterial6.getText().trim().isEmpty()) {
                        Chyba = String.join("\n"
                                 , "Chybí zadání názvu 6.materálu");
                        }
                if (txtRed6.getText() == null || txtRed6.getText().trim().isEmpty()) {
                        Chyba = String.join("\n"
                                 , "Chybí stanoveni červené barvy v RGB u 6.materiálu");
                        }
                else {
                        TestPocet = Integer.parseInt(txtRed6.getText());

                        if (TestPocet > 255) {
                                Chyba = String.join("\n"
                                         , "Stanoveniá hodnota červené barvy v RGB pro
6.materiál jet větší než 255");
                                }
                }
                if (txtGreen6.getText() == null || txtGreen6.getText().trim().isEmpty()) {
                        Chyba = String.join("\n"
                                 , "Chybí stanoveni zelené barvy v RGB u 6.materiálu");
                        }
                else {
                        TestPocet = Integer.parseInt(txtGreen6.getText());

                        if (TestPocet > 255) {
                                Chyba = String.join("\n"
                                         , "Stanoveniá hodnota zelené barvy v RGB pro 6.materiál
jet větší než 255");
                                }
                }
                if (txtBlue6.getText() == null || txtBlue6.getText().trim().isEmpty()) {
                        Chyba = String.join("\n"
                                 , "Chybí stanoveni modré barvy v RGB u 6.materiálu");
                        }
                else {
                        TestPocet = Integer.parseInt(txtBlue6.getText());

                        if (TestPocet > 255) {
                                Chyba = String.join("\n"
                                 , "Stanoveniá hodnota modré barvy v RGB pro 6.materiál jet
větší než 255");
                                }
                }
            break;
        case 7:
                if (txtMaterial1.getText() == null || txtMaterial1.getText().trim().isEmpty()) {
                        Chyba = String.join("\n"
                                 , "Chybí zadání názvu 1.materálu");
                        }
                if (txtRed1.getText() == null || txtRed1.getText().trim().isEmpty()) {
                        Chyba = String.join("\n"
                                 , "Chybí stanoveni červené barvy v RGB u 1.materiálu");
                        }
                else {
                        TestPocet = Integer.parseInt(txtRed1.getText());

                        if (TestPocet > 255) {
                                Chyba = String.join("\n"
                                         , "Stanoveniá hodnota červené barvy v RGB pro
1.materiál jet větší než 255");
                                }
                }
                if (txtGreen1.getText() == null || txtGreen1.getText().trim().isEmpty()) {
                        Chyba = String.join("\n"
                                 , "Chybí stanoveni zelené barvy v RGB u 1.materiálu");
                        }
```

```
                else {
                        TestPocet = Integer.parseInt(txtGreen1.getText());

                        if (TestPocet > 255) {
                                Chyba = String.join("\n"
                                        , "Stanoveniá hodnota zelené barvy v RGB pro 1.materiál
jet větší než 255");
                                }
                }
                if (txtBlue1.getText() == null || txtBlue1.getText().trim().isEmpty()) {
                        Chyba = String.join("\n"
                                , "Chybí stanoveni modré barvy v RGB u 1.materiálu");
                }
                else {
                        TestPocet = Integer.parseInt(txtBlue1.getText());

                        if (TestPocet > 255) {
                                Chyba = String.join("\n"
                                , "Stanoveniá hodnota modré barvy v RGB pro 1.materiál jet
větší než 255");
                                }
                }
                if (txtMaterial2.getText() == null || txtMaterial2.getText().trim().isEmpty()) {
                        Chyba = String.join("\n"
                                , "Chybí zadání názvu 2.materálu");
                }
                if (txtRed2.getText() == null || txtRed2.getText().trim().isEmpty()) {
                        Chyba = String.join("\n"
                                , "Chybí stanoveni červené barvy v RGB u 2.materiálu");
                }
                else {
                        TestPocet = Integer.parseInt(txtRed2.getText());

                        if (TestPocet > 255) {
                                Chyba = String.join("\n"
                                        , "Stanoveniá hodnota červené barvy v RGB pro
2.materiál jet větší než 255");
                                }
                }
                if (txtGreen2.getText() == null || txtGreen2.getText().trim().isEmpty()) {
                        Chyba = String.join("\n"
                                , "Chybí stanoveni zelené barvy v RGB u 2.materiálu");
                }
                else {
                        TestPocet = Integer.parseInt(txtGreen2.getText());

                        if (TestPocet > 255) {
                                Chyba = String.join("\n"
                                        , "Stanoveniá hodnota zelené barvy v RGB pro 2.materiál
jet větší než 255");
                                }
                }
                if (txtBlue2.getText() == null || txtBlue2.getText().trim().isEmpty()) {
                        Chyba = String.join("\n"
                                , "Chybí stanoveni modré barvy v RGB u 2.materiálu");
                }
                else {
                        TestPocet = Integer.parseInt(txtBlue2.getText());

                        if (TestPocet > 255) {
                                Chyba = String.join("\n"
                                , "Stanoveniá hodnota modré barvy v RGB pro 2.materiál jet
větší než 255");
                                }
                }
                if (txtMaterial3.getText() == null || txtMaterial3.getText().trim().isEmpty()) {
                        Chyba = String.join("\n"
                                , "Chybí zadání názvu 3.materálu");
                }
                if (txtRed3.getText() == null || txtRed3.getText().trim().isEmpty()) {
                        Chyba = String.join("\n"
                                , "Chybí stanoveni červené barvy v RGB u 3.materiálu");
                }
                else {
```

```
                        TestPocet = Integer.parseInt(txtRed3.getText());

                        if (TestPocet > 255) {
                                Chyba = String.join("\n"
                                        , "Stanoveniá hodnota červené barvy v RGB pro
3.materiál jet větší než 255");
                                }
                }
                if (txtGreen3.getText() == null || txtGreen3.getText().trim().isEmpty()) {
                        Chyba = String.join("\n"
                                 , "Chybí stanoveni zelené barvy v RGB u 3.materiálu");
                        }
                else {
                        TestPocet = Integer.parseInt(txtGreen3.getText());

                        if (TestPocet > 255) {
                                Chyba = String.join("\n"
                                        , "Stanoveniá hodnota zelené barvy v RGB pro 3.materiál
jet větší než 255");
                                }
                }
                if (txtBlue3.getText() == null || txtBlue3.getText().trim().isEmpty()) {
                        Chyba = String.join("\n"
                                 , "Chybí stanoveni modré barvy v RGB u 3.materiálu");
                        }
                else {
                        TestPocet = Integer.parseInt(txtBlue3.getText());

                        if (TestPocet > 255) {
                                Chyba = String.join("\n"
                                 , "Stanoveniá hodnota modré barvy v RGB pro 3.materiál jet
větší než 255");
                                }
                }
                if (txtMaterial4.getText() == null || txtMaterial4.getText().trim().isEmpty()) {
                        Chyba = String.join("\n"
                                 , "Chybí zadání názvu 4.materálu");
                        }
                if (txtRed4.getText() == null || txtRed4.getText().trim().isEmpty()) {
                        Chyba = String.join("\n"
                                 , "Chybí stanoveni červené barvy v RGB u 4.materiálu");
                        }
                else {
                        TestPocet = Integer.parseInt(txtRed4.getText());

                        if (TestPocet > 255) {
                                Chyba = String.join("\n"
                                        , "Stanoveniá hodnota červené barvy v RGB pro
4.materiál jet větší než 255");
                                }
                }
                if (txtGreen4.getText() == null || txtGreen4.getText().trim().isEmpty()) {
                        Chyba = String.join("\n"
                                 , "Chybí stanoveni zelené barvy v RGB u 4.materiálu");
                        }
                else {
                        TestPocet = Integer.parseInt(txtGreen4.getText());

                        if (TestPocet > 255) {
                                Chyba = String.join("\n"
                                        , "Stanoveniá hodnota zelené barvy v RGB pro 4.materiál
jet větší než 255");
                                }
                }
                if (txtBlue4.getText() == null || txtBlue4.getText().trim().isEmpty()) {
                        Chyba = String.join("\n"
                                 , "Chybí stanoveni modré barvy v RGB u 4.materiálu");
                        }
                else {
                        TestPocet = Integer.parseInt(txtBlue4.getText());

                        if (TestPocet > 255) {
                                Chyba = String.join("\n"
```

```
                                , "Stanoveniá hodnota modré barvy v RGB pro Č.materiál jet
větší než 255");
                        }
                }
                if (txtMaterial5.getText() == null || txtMaterial5.getText().trim().isEmpty()) {
                        Chyba = String.join("\n"
                                , "Chybí zadání názvu 5.materálu");
                        }
                if (txtRed5.getText() == null || txtRed5.getText().trim().isEmpty()) {
                        Chyba = String.join("\n"
                                , "Chybí stanoveni červené barvy v RGB u 5.materiálu");
                        }
                else {
                        TestPocet = Integer.parseInt(txtRed5.getText());

                        if (TestPocet > 255) {
                                Chyba = String.join("\n"
                                        , "Stanoveniá hodnota červené barvy v RGB pro
5.materiál jet větší než 255");
                                }
                }
                if (txtGreen5.getText() == null || txtGreen5.getText().trim().isEmpty()) {
                        Chyba = String.join("\n"
                                , "Chybí stanoveni zelené barvy v RGB u 5.materiálu");
                        }
                else {
                        TestPocet = Integer.parseInt(txtGreen5.getText());

                        if (TestPocet > 255) {
                                Chyba = String.join("\n"
                                        , "Stanoveniá hodnota zelené barvy v RGB pro 5.materiál
jet větší než 255");
                                }
                }
                if (txtBlue5.getText() == null || txtBlue5.getText().trim().isEmpty()) {
                        Chyba = String.join("\n"
                                , "Chybí stanoveni modré barvy v RGB u 5.materiálu");
                        }
                else {
                        TestPocet = Integer.parseInt(txtBlue5.getText());

                        if (TestPocet > 255) {
                                Chyba = String.join("\n"
                                , "Stanoveniá hodnota modré barvy v RGB pro 5.materiál jet
větší než 255");
                                }
                }
                if (txtMaterial6.getText() == null || txtMaterial6.getText().trim().isEmpty()) {
                        Chyba = String.join("\n"
                                , "Chybí zadání názvu 6.materálu");
                        }
                if (txtRed6.getText() == null || txtRed6.getText().trim().isEmpty()) {
                        Chyba = String.join("\n"
                                , "Chybí stanoveni červené barvy v RGB u 6.materiálu");
                        }
                else {
                        TestPocet = Integer.parseInt(txtRed6.getText());

                        if (TestPocet > 255) {
                                Chyba = String.join("\n"
                                        , "Stanoveniá hodnota červené barvy v RGB pro
6.materiál jet větší než 255");
                                }
                }
                if (txtGreen6.getText() == null || txtGreen6.getText().trim().isEmpty()) {
                        Chyba = String.join("\n"
                                , "Chybí stanoveni zelené barvy v RGB u 6.materiálu");
                        }
                else {
                        TestPocet = Integer.parseInt(txtGreen6.getText());

                        if (TestPocet > 255) {
                                Chyba = String.join("\n"
```

```
                                        , "Stanoveniá hodnota zelené barvy v RGB pro 6.materiál
jet větší než 255");
                                }
                }
                if (txtBlue6.getText() == null || txtBlue6.getText().trim().isEmpty()) {
                        Chyba = String.join("\n"
                                 , "Chybí stanoveni modré barvy v RGB u 6.materiálu");
                        }
                else {
                        TestPocet = Integer.parseInt(txtBlue6.getText());

                        if (TestPocet > 255) {
                                Chyba = String.join("\n"
                                 , "Stanoveniá hodnota modré barvy v RGB pro 6.materiál jet
větší než 255");
                                }
                }
                if (txtMaterial7.getText() == null || txtMaterial7.getText().trim().isEmpty()) {
                        Chyba = String.join("\n"
                                 , "Chybí zadání názvu 7.materálu");
                        }
                if (txtRed7.getText() == null || txtRed7.getText().trim().isEmpty()) {
                        Chyba = String.join("\n"
                                 , "Chybí stanoveni červené barvy v RGB 7.materiálu\"");
                        }
                else {
                        TestPocet = Integer.parseInt(txtRed7.getText());

                if (TestPocet > 255) {
                        Chyba = String.join("\n"
                                 , "Stanoveniá hodnota červené barvy v RGB pro 7.materiál jet
větší než 255");
                                }
                }
                if (txtGreen7.getText() == null || txtGreen7.getText().trim().isEmpty()) {
                        Chyba = String.join("\n"
                                 , "Chybí stanoveni zelené barvy v RGB 7.materiálu\"");
                        }
                else {
                        TestPocet = Integer.parseInt(txtGreen7.getText());

                if (TestPocet > 255) {
                        Chyba = String.join("\n"
                                 , "Stanoveniá hodnota zelené barvy v RGB pro 7.materiál jet
větší než 255");
                                }
                }
                if (txtBlue7.getText() == null || txtBlue7.getText().trim().isEmpty()) {
                        Chyba = String.join("\n"
                                 , "Chybí stanoveni modré barvy v RGB 7.materiálu");
                        }
                else {
                        TestPocet = Integer.parseInt(txtBlue7.getText());

                if (TestPocet > 255) {
                        Chyba = String.join("\n"
                                 , "Stanoveniá hodnota modré barvy v RGB pro 7.materiál jet
větší než 255");
                                }
                                }
        break;
        }

                if (Chyba == ("Objevily se tihle problémy:")) {

                        DirectoryChooser directoryChooser = new DirectoryChooser();
                        directoryChooser.setInitialDirectory(new File("src"));
                    File selectedDirectory = directoryChooser.showDialog(primaryStage);
                    System.out.println(selectedDirectory.getAbsolutePath());
                   String Cesta = selectedDirectory.getAbsolutePath();

                            try {
                                File myObj = new File(Cesta+"\\"+CreateTxt);
                                if (myObj.createNewFile()) {
```

```
                System.out.println("File created: " + myObj.getName());
              } else {
                System.out.println("File already exists.");
              }
            } catch (IOException e) {
              System.out.println("An error occurred.");
              e.printStackTrace();
            }
          try {
              File myObj = new File(Cesta+"\\"+SetTxt);
              if (myObj.createNewFile()) {
                System.out.println("File created: " + myObj.getName());
              } else {
                System.out.println("File already exists.");
              }
            } catch (IOException e) {
              System.out.println("An error occurred.");
              e.printStackTrace();
            }
          try {
              File myObj = new File(Cesta+"\\"+ColorTxt);
              if (myObj.createNewFile()) {
                System.out.println("File created: " + myObj.getName());
              } else {
                System.out.println("File already exists.");
              }
            } catch (IOException e) {
              System.out.println("An error occurred.");
              e.printStackTrace();
            }
          try {
              File myObj = new File(Cesta+"\\"+IdTxt);
              if (myObj.createNewFile()) {
                System.out.println("File created: " + myObj.getName());
              } else {
                System.out.println("File already exists.");
              }
            } catch (IOException e) {
              System.out.println("An error occurred.");
              e.printStackTrace();
            }
try {
    FileWriter myWriter1 = new FileWriter(Cesta+"\\"+CreateTxt);
    myWriter1.write("var i : integer");
    myWriter1.write(System.getProperty("line.separator"));
    myWriter1.write("var line : object\"");
    myWriter1.write(System.getProperty("line.separator"));
    myWriter1.write("line := track");
    myWriter1.write(System.getProperty("line.separator"));
    myWriter1.write("for i := 1 to Line.numsensors");
    myWriter1.write(System.getProperty("line.separator"));
    myWriter1.write("Line.deletesensor(i)");
    myWriter1.write(System.getProperty("line.separator"));
    myWriter1.write("next");
    myWriter1.write(System.getProperty("line.separator"));
    myWriter1.write("for i := 1 Tab_Sensors.ydim");
    myWriter1.write(System.getProperty("line.separator"));
    myWriter1.write("Line.createsensor(Tab_Sensors[1,i])");
    myWriter1.write(System.getProperty("line.separator"));
    myWriter1.write("Line.sensorID(i.ctrl:=&Met_Sensor_Identification)");
    myWriter1.write(System.getProperty("line.separator"));
    myWriter1.write("next");
    myWriter1.write(System.getProperty("line.separator"));
    myWriter1.close();

    FileWriter myWriter2 = new FileWriter(Cesta+"\\"+SetTxt);
    myWriter2.write("@.createAttr(\"Storage\",\"object\")");
    myWriter2.write(System.getProperty("line.separator"));
    myWriter2.write("@.createAttr(\"sensor\",\"integer\")");
    myWriter2.write(System.getProperty("line.separator"));
    myWriter2.close();

    FileWriter myWriter3 = new FileWriter(Cesta+"\\"+ColorTxt);
```

```
                        switch (PocetNumber) {
                        case 1:
                                myWriter3.write("if @.Name = \""+txtMaterial1+"\"");
                                myWriter3.write(System.getProperty("line.separator"));
                                myWriter3.write("@.vectorgraphicscolo :=
makeRGBValue("+txtRed1+","+txtGreen1+","+txtBlue1+")");
                                myWriter3.write(System.getProperty("line.separator"));
                            break;
                        case 2:
                                myWriter3.write("if @.Name = \""+txtMaterial1+"\"");
                                myWriter3.write(System.getProperty("line.separator"));
                                myWriter3.write("@.vectorgraphicscolo :=
makeRGBValue("+txtRed1+","+txtGreen1+","+txtBlue1+")");
                                myWriter3.write(System.getProperty("line.separator"));
                                myWriter3.write("elseif @.Name = \""+txtMaterial2+"\"");
                                myWriter3.write(System.getProperty("line.separator"));
                                myWriter3.write("@.vectorgraphicscolo :=
makeRGBValue("+txtRed2+","+txtGreen2+","+txtBlue2+")");
                                myWriter3.write(System.getProperty("line.separator"));
                            break;
                        case 3:
                                myWriter3.write("if @.Name = \""+txtMaterial1+"\"");
                                myWriter3.write(System.getProperty("line.separator"));
                                myWriter3.write("@.vectorgraphicscolo :=
makeRGBValue("+txtRed1+","+txtGreen1+","+txtBlue1+")");
                                myWriter3.write(System.getProperty("line.separator"));
                                myWriter3.write("elseif @.Name = \""+txtMaterial2+"\"");
                                myWriter3.write(System.getProperty("line.separator"));
                                myWriter3.write("@.vectorgraphicscolo :=
makeRGBValue("+txtRed2+","+txtGreen2+","+txtBlue2+")");
                                myWriter3.write(System.getProperty("line.separator"));
                                myWriter3.write("elseif @.Name = \""+txtMaterial3+"\"");
                                myWriter3.write(System.getProperty("line.separator"));
                                myWriter3.write("@.vectorgraphicscolo :=
makeRGBValue("+txtRed3+","+txtGreen3+","+txtBlue3+")");
                                myWriter3.write(System.getProperty("line.separator"));
                            break;
                        case 4:
                                myWriter3.write("if @.Name = \""+txtMaterial1+"\"");
                                myWriter3.write(System.getProperty("line.separator"));
                                myWriter3.write("@.vectorgraphicscolo :=
makeRGBValue("+txtRed1+","+txtGreen1+","+txtBlue1+")");
                                myWriter3.write(System.getProperty("line.separator"));
                                myWriter3.write("elseif @.Name = \""+txtMaterial2+"\"");
                                myWriter3.write(System.getProperty("line.separator"));
                                myWriter3.write("@.vectorgraphicscolo :=
makeRGBValue("+txtRed2+","+txtGreen2+","+txtBlue2+")");
                                myWriter3.write(System.getProperty("line.separator"));
                                myWriter3.write("elseif @.Name = \""+txtMaterial3+"\"");
                                myWriter3.write(System.getProperty("line.separator"));
                                myWriter3.write("@.vectorgraphicscolo :=
makeRGBValue("+txtRed3+","+txtGreen3+","+txtBlue3+")");
                                myWriter3.write(System.getProperty("line.separator"));
                                myWriter3.write("elseif @.Name = \""+txtMaterial4+"\"");
                                myWriter3.write(System.getProperty("line.separator"));
                                myWriter3.write("@.vectorgraphicscolo :=
makeRGBValue("+txtRed4+","+txtGreen4+","+txtBlue4+")");
                                myWriter3.write(System.getProperty("line.separator"));
                            break;
                        case 5:
                                myWriter3.write("if @.Name = \""+txtMaterial1+"\"");
                                myWriter3.write(System.getProperty("line.separator"));
                                myWriter3.write("@.vectorgraphicscolo :=
makeRGBValue("+txtRed1+","+txtGreen1+","+txtBlue1+")");
                                myWriter3.write(System.getProperty("line.separator"));
                                myWriter3.write("elseif @.Name = \""+txtMaterial2+"\"");
                                myWriter3.write(System.getProperty("line.separator"));
                                myWriter3.write("@.vectorgraphicscolo :=
makeRGBValue("+txtRed2+","+txtGreen2+","+txtBlue2+")");
                                myWriter3.write(System.getProperty("line.separator"));
                                myWriter3.write("elseif @.Name = \""+txtMaterial3+"\"");
                                myWriter3.write(System.getProperty("line.separator"));
                                myWriter3.write("@.vectorgraphicscolo :=
makeRGBValue("+txtRed3+","+txtGreen3+","+txtBlue3+")");
```

```
                                myWriter3.write(System.getProperty("line.separator"));
                                myWriter3.write("elseif @.Name = \""+txtMaterial4+"\"");
                                myWriter3.write(System.getProperty("line.separator"));
                                myWriter3.write("@.vectorgraphicscolo :=
makeRGBValue("+txtRed4+","+txtGreen4+","+txtBlue4+")");
                                myWriter3.write(System.getProperty("line.separator"));
                                myWriter3.write("elseif @.Name = \""+txtMaterial5+"\"");
                                myWriter3.write(System.getProperty("line.separator"));
                                myWriter3.write("@.vectorgraphicscolo :=
makeRGBValue("+txtRed5+","+txtGreen5+","+txtBlue5+")");
                                myWriter3.write(System.getProperty("line.separator"));
                            break;
                        case 6:
                                myWriter3.write("if @.Name = \""+txtMaterial1+"\"");
                                myWriter3.write(System.getProperty("line.separator"));
                                myWriter3.write("@.vectorgraphicscolo :=
makeRGBValue("+txtRed1+","+txtGreen1+","+txtBlue1+")");
                                myWriter3.write(System.getProperty("line.separator"));
                                myWriter3.write("elseif @.Name = \""+txtMaterial2+"\"");
                                myWriter3.write(System.getProperty("line.separator"));
                                myWriter3.write("@.vectorgraphicscolo :=
makeRGBValue("+txtRed2+","+txtGreen2+","+txtBlue2+")");
                                myWriter3.write(System.getProperty("line.separator"));
                                myWriter3.write("elseif @.Name = \""+txtMaterial3+"\"");
                                myWriter3.write(System.getProperty("line.separator"));
                                myWriter3.write("@.vectorgraphicscolo :=
makeRGBValue("+txtRed3+","+txtGreen3+","+txtBlue3+")");
                                myWriter3.write(System.getProperty("line.separator"));
                                myWriter3.write("elseif @.Name = \""+txtMaterial4+"\"");
                                myWriter3.write(System.getProperty("line.separator"));
                                myWriter3.write("@.vectorgraphicscolo :=
makeRGBValue("+txtRed4+","+txtGreen4+","+txtBlue4+")");
                                myWriter3.write(System.getProperty("line.separator"));
                                myWriter3.write("elseif @.Name = \""+txtMaterial5+"\"");
                                myWriter3.write(System.getProperty("line.separator"));
                                myWriter3.write("@.vectorgraphicscolo :=
makeRGBValue("+txtRed5+","+txtGreen5+","+txtBlue5+")");
                                myWriter3.write(System.getProperty("line.separator"));
                                myWriter3.write("elseif @.Name = \""+txtMaterial6+"\"");
                                myWriter3.write(System.getProperty("line.separator"));
                                myWriter3.write("@.vectorgraphicscolo :=
makeRGBValue("+txtRed6+","+txtGreen6+","+txtBlue6+")");
                                myWriter3.write(System.getProperty("line.separator"));
                            break;
                        case 7:
                                myWriter3.write("if @.Name = \""+txtMaterial1+"\"");
                                myWriter3.write(System.getProperty("line.separator"));
                                myWriter3.write("@.vectorgraphicscolo :=
makeRGBValue("+txtRed1+","+txtGreen1+","+txtBlue1+")");
                                myWriter3.write(System.getProperty("line.separator"));
                                myWriter3.write("elseif @.Name = \""+txtMaterial2+"\"");
                                myWriter3.write(System.getProperty("line.separator"));
                                myWriter3.write("@.vectorgraphicscolo :=
makeRGBValue("+txtRed2+","+txtGreen2+","+txtBlue2+")");
                                myWriter3.write(System.getProperty("line.separator"));
                                myWriter3.write("elseif @.Name = \""+txtMaterial3+"\"");
                                myWriter3.write(System.getProperty("line.separator"));
                                myWriter3.write("@.vectorgraphicscolo :=
makeRGBValue("+txtRed3+","+txtGreen3+","+txtBlue3+")");
                                myWriter3.write(System.getProperty("line.separator"));
                                myWriter3.write("elseif @.Name = \""+txtMaterial4+"\"");
                                myWriter3.write(System.getProperty("line.separator"));
                                myWriter3.write("@.vectorgraphicscolo :=
makeRGBValue("+txtRed4+","+txtGreen4+","+txtBlue4+")");
                                myWriter3.write(System.getProperty("line.separator"));
                                myWriter3.write("elseif @.Name = \""+txtMaterial5+"\"");
                                myWriter3.write(System.getProperty("line.separator"));
                                myWriter3.write("@.vectorgraphicscolo :=
makeRGBValue("+txtRed5+","+txtGreen5+","+txtBlue5+")");
                                myWriter3.write(System.getProperty("line.separator"));
                                myWriter3.write("elseif @.Name = \""+txtMaterial6+"\"");
                                myWriter3.write(System.getProperty("line.separator"));
                                myWriter3.write("@.vectorgraphicscolo :=
makeRGBValue("+txtRed6+","+txtGreen6+","+txtBlue6+")");
```

```
                                myWriter3.write(System.getProperty("line.separator"));
                                myWriter3.write("elseif @.Name = \""+txtMaterial7+"\"");
                                myWriter3.write(System.getProperty("line.separator"));
                                myWriter3.write("@.vectorgraphicscolo :=
makeRGBValue("+txtRed7+","+txtGreen7+","+txtBlue7+")");
                                myWriter3.write(System.getProperty("line.separator"));
                            break;
                         }
                        myWriter3.write("end");
                        myWriter3.write(System.getProperty("line.separator"));
                        myWriter3.close();

                        FileWriter myWriter4 = new FileWriter(Cesta+"\\"+IdTxt);
                        myWriter4.write("param sensorID: integer, Front: boolean, BookPos:
boolean");
                        myWriter4.write(System.getProperty("line.separator"));
                        myWriter4.write("if @.Sensor = SensorID");
                        myWriter4.write(System.getProperty("line.separator"));
                        myWriter4.write("@.stopped := true");
                        myWriter4.write(System.getProperty("line.separator"));
                        myWriter4.write("wait "+WaitBox);
                        myWriter4.write(System.getProperty("line.separator"));
                        myWriter4.write("@.const.move(@.Storage)");
                        myWriter4.write(System.getProperty("line.separator"));
                        myWriter4.write("@.Storage := \"\"");
                        myWriter4.write(System.getProperty("line.separator"));
                        myWriter4.write("@.Sensor := 0");
                        myWriter4.write(System.getProperty("line.separator"));
                        myWriter4.write("@.stopped := false");
                        myWriter4.write(System.getProperty("line.separator"));
                        myWriter4.write("end");
                        myWriter4.write(System.getProperty("line.separator"));
                        myWriter4.close();

                    } catch (IOException e) {
                        System.out.println("An error occurred.");
                        e.printStackTrace();
                    }

                                }
                                else  btn2.fire();

         });

                                btn2.setOnAction(new EventHandler<ActionEvent>() {

                                @Override
                                public void handle(ActionEvent event) {
                                        showAlertWithHeaderText();
                                }
                    });

        Scene scene=new Scene(grid,600,400);
        primaryStage.setTitle("BakalarAplcation4");
        primaryStage.setScene(scene);
        primaryStage.show();
    }
    public static void main (String[] args)
    {
        launch(args);
    }
}
```

Autor/ka BP	**Filip Dolák**
Název BP	**Zefektivnění činnosti tvorby simulačního modelu v programu Tecnomatix Plant Simulation**
Studijní program	**IPL**
Rok obhajoby BP	**2022**
Počet stran	48
Počet příloh	7
Vedoucí BP	**prof. Ing. Gabriel Fedorko, Ph.D.**
Anotace	Cílem bakalářské práce je vytvořit, popsat a vytvořit aplikaci sloužící jako překladač pro Simulační program, který by dokázal výrazně zlehčit vytvoření potřebné simulace. Také následně rozebrat a plně vysvětli, jak daná aplikace funguje, včetně jejich jednotlivých části a objasnění důvodu proč bylo konkrétní řešení použito. Na závěr je aplikace zde demonstrována na několika příkladech, které předvedou funkčnost aplikace a napomohou objasnění fungování aplikace krok po kroku.
Klíčová slova	Logistika, logistická simulace, počítačová simulace, překladač, programovací jazyk, simulační program, schéma
Místo uložení	ITC (knihovna) Vysoké školy logistiky v Přerově
Signatura	