Pedagogická fakulta
Faculty of Education

Jihočeská univerzita v Českých Budějovicích
University of South Bohemia in České Budějovice

Jihočeská univerzita v Českých Budějovicích
Pedagogická fakulta
Katedra informatiky

Bakalářská práce

Tvorba grafického uživatelského rozhraní v BlueJ

Vypracoval: Tomáš Svatek
Vedoucí práce: RNDr. Hana Havelková

České Budějovice 2013

Prohlášení

Prohlašuji, že svoji bakalářskou práci jsem vypracoval samostatně pouze s použitím pramenů a literatury uvedených v seznamu citované literatury.

Prohlašuji, že v souladu s § 47b zákona č. 111/1998 Sb. v platném znění souhlasím se zveřejněním své bakalářské práce, a to v nezkrácené podobě v úpravě vzniklé vypuštěním vyznačených částí archivovaných fakultou elektronickou cestou ve veřejně přístupné části databáze STAG provozované Jihočeskou univerzitou v Českých Budějovicích na jejích internetových stránkách, a to se zachováním mého autorského práva k odevzdanému textu této kvalifikační práce. Souhlasím dále s tím, aby toutéž elektronickou cestou byly v souladu s uvedeným ustanovením zákona č. 111/1998 Sb. zveřejněny posudky školitele a oponentů práce i záznam o průběhu a výsledku obhajoby kvalifikační práce. Rovněž souhlasím s porovnáním textu mé kvalifikační práce s databází kvalifikačních prací Theses.cz provozovanou Národním registrem vysokoškolských kvalifikačních prací a systémem na odhalování plagiátů.

V ..... dne ..... Podpis autora

JIHOČESKÁ UNIVERZITA V ČESKÝCH BUDĚJOVICÍCH
Fakulta pedagogická
Akademický rok: **2011/2012**

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	**Tomáš SVATEK**
Osobní číslo:	**P10372**
Studijní program:	**B7507 Specializace v pedagogice**
Studijní obor:	**Informační technologie ve vzdělávání**
Název tématu:	**Tvorba GUI v BlueJ**
Zadávající katedra:	**Katedra informatiky**

Zásady pro vypracování:

BlueJ je integrované vývojové prostředí pro práci v Javě specielně určené k výuce programování nabízející řadu nástrojů usnadňujících práci hlavně začínajícím programátorům. Většina aplikací se snaží poskytnout uživateli příjemné grafické prostředí, jehož vytváření není úplně triviální a v prostředí BlueJ vyloženě pracné a náročné.
Cílem práce je tedy vytvoření modulu pro tvorbu GUI (Graphical User Interface) pro výukové prostředí BlueJ - tzv. GUI extension, které poskytne programátorovi pracujícímu v BlueJ řadu nástrojů usnadňujících tvorbu GUI. K dispozici bude např. paleta standardních komponent, seznam vlastností a základních událostí.
V teoretické části práce autor stručně nastíní tvorbu GUI v Javě - práci s okny a jejich komponentami (AWT a swing), problematiku obsluhy událostí.
Modul by měl být k dispozici všem uživatelům prostředí BlueJ - měl by být zveřejněn na stránkách http://bluej.org
Součástí práce by měla být i ukázková aplikace dokumentující funkčnost vytvořeného rozšíření BlueJ.

Rozsah grafických prací: **CD ROM**

Rozsah pracovní zprávy: **40**

Forma zpracování bakalářské práce: **tištěná**

Seznam odborné literatury:

1. **HEROUT, P. Java - grafické uživatelské prostředí a čeština. České Budějovice: Kopp, 2007.**
2. **ECKEL, B. Thinking in Java. Prentice Hall PTR, 2006.**
3. **BARNES, D., J., KÖLLING, M. Objects First with Java A Practical Introduction using BlueJ, 5th edition. Prentice Hall / Pearson Education, 2012. ISBN 978-013-249266-9.**
4. **http://www.bluej.org/doc/documentation.html**
5. **http://www.bluej.org/extensions/extensions.html**

Vedoucí bakalářské práce: **RNDr. Hana Havelková**
Katedra informatiky

Datum zadání bakalářské práce: **19. dubna 2012**

Termín odevzdání bakalářské práce: **26. dubna 2013**

Mgr. Michal Vančura, Ph.D.
děkan

L.S.

doc. PaedDr. Jiří Vaníček, Ph.D.
vedoucí katedry

V Českých Budějovicích dne 12. dubna 2012

Abstrakt

Práce z oblasti programování pojednává v teoretické části o problematice grafiky a implementaci grafického rozhraní v jazyce Java (AWT a Swing) a přiřazování událostí ovládacím prvkům. Práce bude zaměřena na prostředí BlueJ, které, na rozdíl od prostředí NetBeans, neobsahuje grafický designér. Hlavním výstupem a cílem této práce bude tedy modul grafického uživatelského rozhraní do programovacího prostředí BlueJ, usnadňující návrh grafického prostředí. Tento doplněk bude jednoduchý a intuitivní na ovládání a bude dostačovat na ozkoušení či naučení základů tvorby grafického rozhraní v jazyce Java. Součástí práce bude i instalační manuál a dokumentace kódu. Výsledný modul bude prezentován na webových stránkách vývojového prostředí.

Klíčová slova:

programování, Java, BlueJ, AWT, SWING, grafické uživatelské rozhraní, BlueJ doplněk

Abstrakt

My work from the field of programming discusses the issue of graphics and implementation of graphical user interface in the Java programming language (AWT and Swing) and assigning events to the control elements. The work will be focused on the BlueJ environment, which, unlike the NetBeans environment, is not supplemented with a graphical user interface designer. The main output and goal of this work is going to be the graphical user interface module into the BlueJ programming environment, which makes the design of a graphical application easier. Concerning the usage, this extensions will be simple and intuitive, and it will be sufficient for testing and learning the basics of a creation of the graphical user interface in the Java language. Part of the work will also be an installation manual and the code documentation. The resulting module will be presented on a web page of the development environment.

Keywords:

Programming, Java, BlueJ, AWT, SWING, Graphical User Interface, BlueJ extensions

Poděkování

Rád bych tímto poděkoval své vedoucí práce paní RNDr. Haně Havelkové za rady a čas, který mi věnovala při řešení dané problematiky.

Dále také děkuji kamarádům, kteří svými nápady přispěli a v budoucnu ještě určitě přispějí k lepší funkčnosti naprogramované aplikace.

Děkuji rovněž své rodině, jež se mnou měla trpělivost a ve studiu mne vždy podporovala.

Obsah

1 Úvod **8**

1.1 Cíle práce 8

1.2 Motivace 9

2 Java a tvorba grafického rozhraní **10**

2.1 AWT 10

2.2 SWING 10

2.3 Porovnání 11

2.4 SWT 12

3 JComponent a práce s ní **12**

3.1 JComponents 12

3.2 Kontejner 14

3.3 Layouty 16

3.3.1 Nulový Layout 16

3.3.2 FlowLayout 17

3.3.3 BorderLayout 17

3.3.4 GridLayout 18

3.3.5 Další layouty 19

3.4 Eventy 19

3.4.1 ActionListener 21

3.4.2 KeyListener 21

3.4.3 MouseListener 22

3.4.4 MouseMotionListener 23

3.4.5 MouseWheelListener 24

3.5 Tvorba jednoduchého GUI 25

4 GUI pro BlueJ **27**

4.1 BlueJ API 27

4.2 Vývoj modulu 30

4.2.1 Grafické rozhraní doplňku 31

4.3 Dokumentace 32

4.3.1 Instalace . . . 32
4.3.2 Spuštění aplikace . . . 33
4.3.3 Vkládání a mazání komponent . . . 33
4.3.4 Nastavení vlastností komponent . . . 35
4.3.5 Nastavení eventů . . . 36
4.3.6 Tvorba menu . . . 38
4.3.7 Generování kódu . . . 39
4.3.8 Uložení kódu . . . 40
4.4 Popis stěžejních tříd doplňku . . . 40

5 Závěr **48**

6 Reference **49**

7 Přílohy **52**
7.1 Obrázky . . . 52
7.2 Ukázka zdrojového kódu vygenerovaného aplikací . . . 55
7.3 Class diagram hlavních tříd . . . 61
7.4 Příloha CD . . . 62

1 Úvod

Java dnes nepředstavuje jen programovací jazyk, ale celou platformu, na níž je postaven software domácích spotřebičů, mobilních telefonů, bankomatů, platebních terminálů..., a samozřejmě nejrůznějších jednoduchých i značně složitých desktopových i webových aplikací Její přenositelnost mezi operačními systémy je její obrovskou výhodou. Existuje několik verzí Javy, jako:

- Java ME (Micro Edition) – pro vývoj mobilních aplikací
- Java EE (Enterprise Edition) - určená pro vývoj a provoz podnikových aplikací a informačních systémů
- Java SE (Standard Edition) - Java, tak jak byla vyvíjena od první verze a postupně rozšiřována.
- Java FX – relativně nová verze jazyka Java navržená pro rychlý a snadný vývoj desktopových aplikací, webových aplikací i pro mobilní oblast.

S narůstajícími nároky na funkčnost aplikací se jazyk Java stále vyvíjí a pro pohodlí uživatelů by se bez návrhu grafického uživatelského rozhraní jen těžce obešla.

Díky jejímu rozšíření vznikly i nové nástroje pro vývoj aplikací jako je právě prostředí BlueJ, jež je určeno především začínajícím programátorů. Jedná se o jednoduché prostředí, které je možné dále rozšiřovat vlastními moduly tzv. extensions a zpříjemnit si tak vývoj aplikace ještě více.

1.1 Cíle práce

BlueJ je integrované vývojové prostředí pro práci v Javě speciálně určené k výuce programování nabízející řadu nástrojů usnadňujících práci hlavně začínajícím programátorům.

Většina aplikací se snaží poskytnout uživateli příjemné grafické prostředí, jehož vytváření není úplně triviální a v prostředí BlueJ vyloženě pracné a náročné . Proč tedy zůstávat u toho prostředí? Protože žádné z existujících prostředí nepředstavuje ideální výukový nástroj. Zejména, chceme-li prostředí, které klade důraz na třídy a objekty jako základní jednotky interakce. Jedná se o prostředí sofistikované, ale zároveň jednoduché na ovládání. K vizualizaci používá strukturu tříd. Koneckonců, polovinu času při výuce strávíme vysvětlováním a ukázkami jak na sebe objekty a třídy navazují.

Přesto mnozí učitelé zjistili, že studenti mají potíže právě s touto představou, pokud jde o třídy a objekty. To není překvapující, pokud vše, co kdy vidíte na obrazovce, jsou řádky kódu, nebo rozhraní, tlačítek a menu[1].

Cílem práce je tedy vytvoření modulu pro tvorbu GUI (Graphical User Interface) pro výukové prostředí BlueJ - tzv. GUI extensions, které poskytne programátorovi pracujícímu v BlueJ řadu nástrojů usnadňujících tvorbu GUI. K dispozici bude např. paleta standardních komponent, seznam vlastností a základních událostí.

V teoretické části práce stručně nastíním tvorbu GUI v Javě práci s okny a jejich komponentami (AWT a SWING), problematiku obsluhy událostí.

Modul by měl být k dispozici všem uživatelům prostředí BlueJ - měl by být zveřejněn na stránkách http://bluej.org. Součástí práce by měla být i ukázková aplikace dokumentující funkčnost vytvořeného rozšíření BlueJ.

1.2 Motivace

GUI je v dnešní době již nedílnou součástí většiny aplikací. Poskytuje uživateli jednoduché ovládání programu, rychlost a pohodlnost ovládání jeho běhu. Bohužel někdy je vytvoření uživatelského rozhraní značně pracné, hlavně pro začínající programátory a nepřítomnost jeho návrháře v BlueJ je značně omezující.

Naskytuje se možnost použití jiného vývojového prostředí, ale používání více IDE pro tvorbu aplikace, není rozumné a přechod z jednoho do druhého zdržuje. Navíc si zkušený programátor napíše uživatelské rozhraní sám . Proč tedy nemít vše pod jedním vývojovým prostředím?

Existuje spousta doplňků do onoho prostředí a to od práce s 3D grafikou až po UML a grafické vizualizace kódu, ale neexistuje doplněk právě pro tvorbu grafického uživatelského rozhraní.

Proto jsem se rozhodl naprogramovat vlastní rozšíření, které se za pomoci tzv. BlueJ API (Application Programming Interface), nechají v integrovat do prostředí BlueJ. Toto rozšíření bude poskytovat jednoduché a intuitivní ovládání založené na zkušenostech z jiných prostředí, ale bude jednodušší a vygenerovaný kód bude přehledně okomentován.

2 Java a tvorba grafického rozhraní

V dnešní době se najde jen málo aplikací, které by byly bez grafického uživatelské rozhraní. Toto rozhraní tvoří většinou základ celého programu, jenž uživateli dovoluje pracovat s daty pohodlným a rychlým způsobem.

Programovací jazyk Java nabízí standardně dva balíčky pro práci s GUI a to balíček AWT, který je starší, a Java Foundation Class (JFC) SWING.

2.1 AWT

Takzvaný Abstract Windows Toolkit je původní grafický balíček pro Javu, jejíž první verze byla vydána v roce 1995. Jeho výhodou je, že je obsažen v každé verzi Javy včetně implementací ve starých prohlížečích a je velmi stabilní. To znamená, že se můžete spolehnout na to, že bude fungovat všude, kde je Java Runtime Environment (minimální prostředí pro běh programů napsaných v Javě) a bude mít vlastnosti, které očekáváte [2]. Nevýhodou je, že má vzhled stejný jako operační systém, na kterém je spuštěn.

AWT je velmi jednoduchý nástroj, sada komponent s omezenými GUI prvky. V důsledku toho, jsou bohužel některé běžně používané komponenty, jako jsou tabulky, stromy, progres bary a další, nedostupné [3].

Prostřednictvím Graphics2D (metody pro práci s 2D grafikou) objektů a Java3D služby vzniká mnoho silných nástrojů jako je kreslení, transformace, a v kombinaci s JavaSound (nástroje pro ovládání hudby) mohou být vytvářeny konkurence schopné interaktivní hry [5]. Vývoj AWT skončil v roce 1997.

2.2 SWING

Grafický balíček pro tvorbu grafického rozhraní přišel na svět s Javou JDK 1.2 (Java Development Kit). Šlo o pokus, který měl vyřešit spoustu nedostatků, kterými disponuje AWT. To se firmě Sun podařilo a navrhlo velice silný a flexibilní grafický nástroj. Syntaxe ve Swing a AWT jsou většinou podobné, neboť Swing vychází z AWT a přebírá od něj většinu ovládacích prvků pro události a samotné grafické objekty [5].

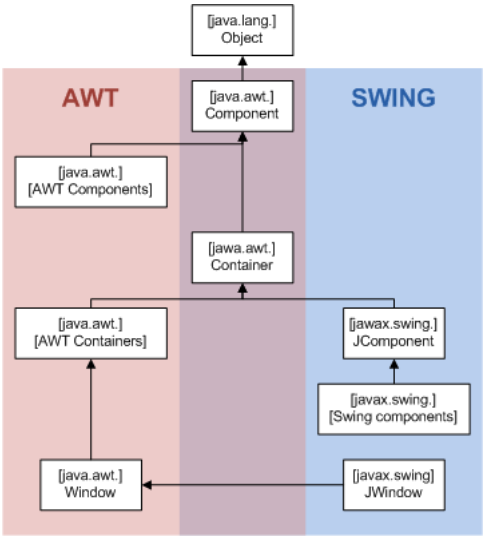
Obrovskou výhodou Swingu ale je, že na rozdíl od AWT, obsahuje mnohem více ovládacích prvků a mnohem více možností jejich nastavení. Programátor sám má možnost libovolně upravovat vzhled komponenty.

Swing poskytuje možnost implementace přídavných technologií pro asistenci tělesně postižených lidí při práci s počítačem. Některé z nich jsou implementovány přímo ve Swingu, jako např. možnost přiřazení klávesových zkratek tlačítkům, nebo přesouvání mezi prvky formuláře pomocí tabulátoru [5]

Nedoporučuje se míchání AWT a Swing prvků. Mohlo by dojít k problémům, popřípadě až k pádu programu.

2.3 Porovnání

V dnešní době již nenarazíte na aplikaci, která by byla napsána čistě v AWT. Díky komplexním možnostem upravovat si ovládací prvky tak, jak chce programátor, a jejich stejnému zobrazení na různých operačních systémech, se stal swing dominantním balíčkem, nicméně je stále doprovázen AWT, ale ne ovládacími prvky jako takovými, ale přiřazováním různých událostí a vykreslování 2D grafiky (viz obr. 1).



Obrázek 1: Hierarchie dědičnosti SWING a AWT [21].

2.4 SWT

Standard Widged Toolkit je knihovna grafických prvků, která k vykreslování používá nativní knihovny operačního systému. Tento balíček je distribuován třetí stranou Eclipse Foundation a není tudíž standardní součástí JDK.

3 JComponent a práce s ní

Základem každého GUI jsou tedy komponenty, které ovládají program nebo zobrazují data. Jedná se o různá textová pole, tlačítka, tabulky a podobné ovládací prvky, které známe z operačního systému. Dále se setkáme s pojmy, jako je kontejner nebo layout.

3.1 JComponents

S výjimkou kontejnerů nejvyšší úrovně (top-level) všechny Swing komponenty, jejichž názvy začínají na "J"vycházejí z třídy JComponent. Například: JPanel, JButton, JRadioButton, JTable a všechny dědí od třídy JComponent [18].

Třída JComponent dědí od třídy Container, která sama dědí od třídy Component (viz Obr. 1). Třída Component obsahuje vše od poskytování layoutů přes podporu malování až k nastavení událostí.

Třída JComponent poskytuje svým potomkům následující funkcionality [18]:

- Tool tips
- Kreslení a rámečky
- Možnost připojení vzhledu Look and feel
- Podporu layoutů
- Podporu přístupu pro postižené [8]
- Podpora drag and drop technologie
- Double bufferingu
- Přiřazování klávesových zkratek

Tool tips

Za pomoci textu jako parametru, můžeme pomocí metody setToolTipText zobrazit textovou nápovědu nad komponentou nad kterou se zastaví myš.

Kreslení a rámečky

Metoda `setBounds()` dovoluje vytváření rámečků okolo komponent několika různými kombinacemi a jejich vnořováním do sebe.

Look and feel

Možnost psaní vlastních vzhledů pro vykreslování komponent, nebo nastavení vzhledu, který je dostupný v JDK(Java Development Kit)

Podpora přístupu pro postižené

Třída JComponent disponuje API se základní podporou usnadnění práce pro postižené.

```
aJComponent.getAccessibleContext().setAccessibleDescription(
        "Kliknutí na tuto komponentu spustí akci XYZ.");
```

Při vytváření komponent většinou používáme již konkrétní typy komponent, a to z důvodu, že některé metody nejsou pro nekonkrétní komponentu dostupné.

```
JComponent component = new JButton();
```

Toto je povolený zápis, nicméně i přesto, že víme, že se bude jednat o tlačítko, stále se jedná o instanci třídy JComponent, a nebude tedy dostupná třeba metoda `getText()`, neboť některé JComponenty tuto metodu nemají. Na druhé straně při vytváření konkrétní komponenty již budou přístupné všechny metody nejen z JComponenty, ale i přidané pro danou komponentu.

```
JButton component = new JButton();
```

Komponenty v Java Swing jsou snadno modifikovatelné a jednoduše použitelné, ale i když je zde Swing již poměrně dlouho, stále se u určitých komponent objevují chyby, jako například neukončení editace buňky tabulky, když se klikne mimo apod.

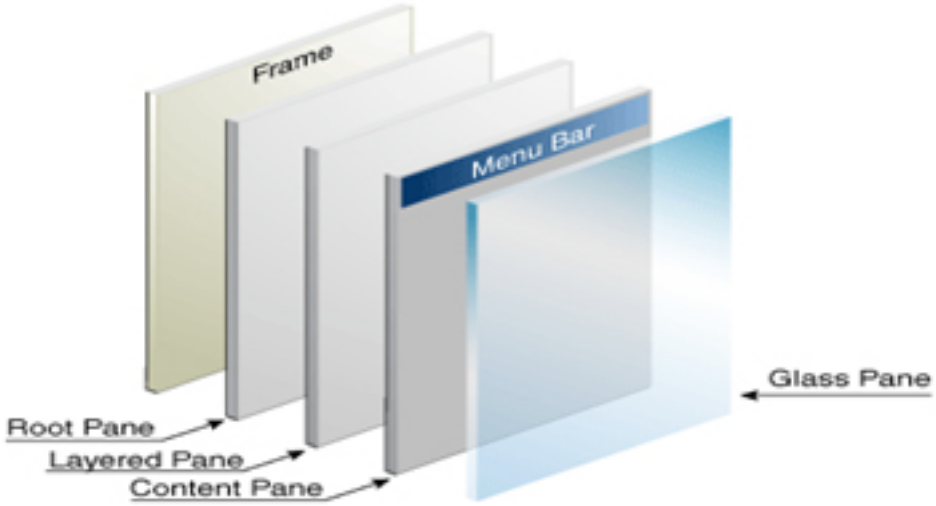
I když těchto chyb není mnoho, trvá spoustu času dohledání řešení, jak takovou chybu odstranit nebo obejít.

3.2 Kontejner

Kontejner je datový objekt, obsahující "v sobě"nějaké další objekty, v našem případě komponenty grafického rozhraní, ke kterým lze různým způsobem přistupovat a pracovat s nimi. Podle způsobů uložení dat a práce s nimi rozlišujeme různé druhy kontejnerů; při jejich použití si vždy vybereme ten, který se pro daný případ nejlépe hodí (viz obr. 2).

V našem případě se jedná o JPanel, JLayeredPane, GlassPane a dále některé top-level kontejnery jako JFrame, JDialog a JApplet.

Každý program, který používá Swing komponenty, má alespoň jeden top-level kontejner, který slouží jako základní obal pro ostatní komponenty.



Obrázek 2: Hierarchie kontejnerů [20].

Při tvorbě GUI se nejčastěji setkáváme s JFrame (okno) a JDialog jako se zástupci top-level kontejnerů a pracujeme s nimi v určité hierarchii. Víme, že JFrame je kořenový kontejner, do kterého se vkládají další komponenty například JPanel, který funguje také jako kontejner, jenž v sobě obsahuje další komponenty.

```
//Vytvoří panel a přidá do něj komponenty
JPanel contentPane = new JPanel(new BorderLayout());
contentPane.setBorder(someBorder);
contentPane.add(someComponent, BorderLayout.CENTER);
contentPane.add(anotherComponent, BorderLayout.PAGE_END);
```

Po vytvoření nějakého obsahu okna, v tomto případě jednoduchého panelu s nějakými komponentami, se tento panel musí přidat do top-level kontejneru, jinak se vytvořený panel nezobrazí.

```
//přidá vytvořený panel do okna
topLevelContainer.setContentPane(contentPane);
```

Pokud bychom udělali to samé ještě jednou, tentokrát s jiným panelem, zobrazil by se v okně pouze obsah, který je dán v `contentPane_2`, ale již neuvidíme první panel a museli bychom udělat následující:

```
contentPane_2.add(contentPane);
topLevelContainer.setContentPane(contentPane_2);
```

Co se panelů týče, máme k dispozici:

- JPanel – jednoduchý panel
- JTabbedPane – panel se záložkami
- JScrollPane – panel, jenž zajistí rolování
- JLayeredPane – panel, na který je možné dávat komponenty ve vrstvách
- JSplitPane – panel, který můžeme rozdělit vertikálně nebo horizontálně a jeho části dále naplňovat

Každému panelu můžeme nastavit tzv. layout (rozložení prvků) [19].

3.3 Layouty

Pod pojmem layout si představujeme nějaké rozložení prvků. Swing disponuje hned několika druhy layoutů a práce s nimi je považována za nejtěžší věc při tvorbě GUI. Nastíním zde práci s pár vybranými layouty.

Layout můžeme nastavit panelu buď při vytváření:

```
JPanel panel = new JPanel(new BorderLayout());
```

Nebo později pomocí metody `setLayout()` změnit:

```
panel.setLayout(new FlowLayout());
```

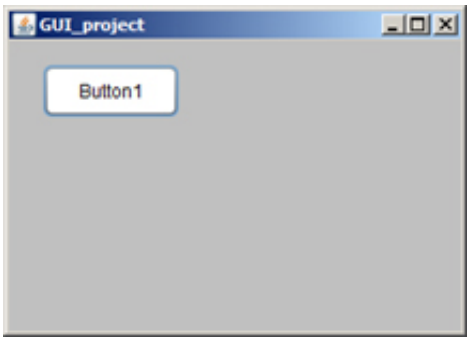
3.3.1 Nulový Layout

Nulový layout neboli panel bez layoutu, se používá tehdy, chceme-li komponenty pozicovat absolutně, to znamená nastavení souřadnic, kde [0,0] je levý horní roh okna aplikace.

Komponenta se poté umisťuje pomocí metody setBounds(), jejíž parametry jsou souřadnice x, y a velikost (šířka, výška). Tato metoda nahrazuje metody `setLocation()` a `setSize()`. Nevýhodou je stálé umístění komponent při změně velikosti okna a neustále dopočítávání souřadnic.

```
JPanel panel = new JPanel(null);
button.setBounds(22,16,90,35);
panel.add(component);
```

Výsledek můžeme vidět na obrázku níže.



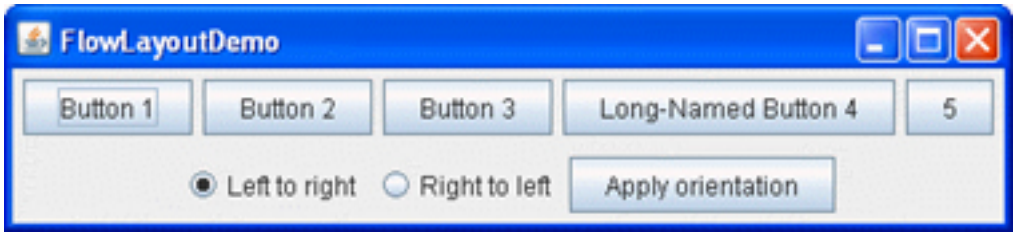
Obrázek 3: Absolutní pozicování

3.3.2 FlowLayout

Jednoduchý layout, který je používán jako výchozí v JPanel komponentě. Třída FlowLayout vkládá komponenty za sebe do řádky o preferované velikosti, bez nutnosti nastavovat lokaci umístění.

Komponenty se při změně velikosti okna posouvají tak, jak jim to dovoluje délka řádky. Nechají se nastavit i vertikální a horizontální mezery mezi řádky.

```
//Základní layout nastaven na FlowLayout
JPanel panel = new JPanel();
//popřípadě možno nastavit
JPanel panel = new JPanel(new FlowLayout());
```



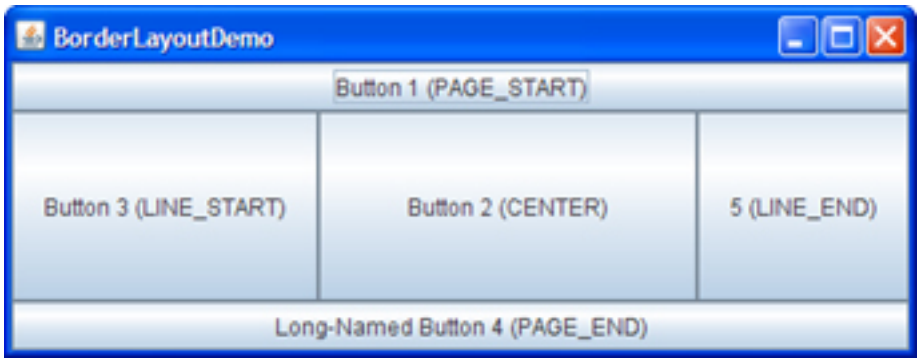
Obrázek 4: Flow Layout [19].

3.3.3 BorderLayout

Toto rozložení dovoluje pozicovat komponenty do pěti pozic.

- PAGE_START
- PAGE_END
- LINE_START
- LINE_END
- CENTER

```
JPanel panel = new JPanel(new BorderLayout());
JButton button = new JButton("Button 1 (PAGE_START)");
//vložení komponenty na pozici v layoutu
panel.add(button, BorderLayout.PAGE_START);
```



Obrázek 5: Border Layout [19].

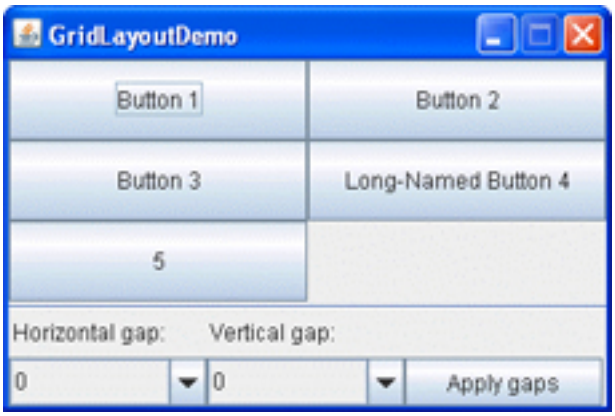
3.3.4 GridLayout

GridLayout vkládá komponenty do mřížky za sebou. Každá mřížka má stejnou velikost. Parametry pro tvorbu tohoto layoutu jsou počet řádek a počet sloupců, popřípadě odsazení komponent mezi sebou.

```
JPanel controls = new JPanel(new GridLayout(2,3));
```

V další ukázce vytvoří konstruktor dva sloupce a tolik řádek, kolik bude potřeba.

```
GridLayout experimentLayout = new GridLayout(0,2);
final JPanel compsToExperiment = new JPanel(experimentLayout);

compsToExperiment.add(new JButton("Button 1"));
compsToExperiment.add(new JButton("Button 2"));
compsToExperiment.add(new JButton("Button 3"));
compsToExperiment.add(new JButton("Long-Named Button 4"));
compsToExperiment.add(new JButton("5"));
```



Obrázek 6: Grid Layout [19].

3.3.5 Další layouty

Existuje spousta dalších možných rozvržení. Některé jsou přímou součástí javax.swing jako:

- CardLayout
- GridBagLayout
- GroupLayout
- SpringLayout

Další layouty jsou dostupné od třetích stran např:

- MiGLayout[1]
- JGoodies Form Layout[2]

3.4 Eventy

Eventy neboli události zajišťují funkčnost ovládacích prvků, reagování na akci, jako kliknutí myší, najetí myší, otočení kolečkem, zmáčknutí tlačítka, zmáčknutí klávesy atp. Dále existují posluchače událostí například při změně výběru v Combo Boxu či v tabulce nebo při označení uzlu stromu.

[1] http://www.miglayout.com/
[2] http://www.jgoodies.com/

Dále máme události na konkrétní komponenty, jako například u záložkového panelu (TabbedPane), kde se nechá nastavit akce na změnu panelu nebo u JComboBoxu kde musí být samozřejmě registrovaná událost na změnu výběru nebo události na zaškrtnutí či odškrtnutí JCheckBoxu apod.

Zmíním zde jen pár vybraných akcí na myš, klávesnici a události tlačítek. Pro to, abychom mohli přiřadit komponentě akci, musíme si vysvětlit pojmy Listener a Adapter [22].

- Listener – jedná se o rozhraní u kterého musíme implementovat všechny metody
- Adapter – konkrétní třída, kde můžeme použít operátor new a přepisujeme jenom konkrétní metodu

```
public class CloseListener implements WindowListener {
   // tento event nepotřebuji, ale i tak ho musím přepsat
    @Override
   public void windowOpened(WindowEvent e) {}
   // tento event nepotřebuji, ale i tak ho musím přepsat
    @Override
   public void windowClosing(WindowEvent e) {}
    @Override
     public void windowClosed(WindowEvent e) {
       System.exit(0);
    }
    // tento event nepotřebuji, ale i tak ho musím přepsat
   @Override
    public void windowIconified(WindowEvent e) {}
   // tento event nepotřebuji, ale i tak ho musím přepsat
    @Override
   public void windowDeiconified(WindowEvent e) {}
}
```

Jak vidíme, je to spousta kódu na rozdíl od programování pouze adaptéru.

```
addWindowListener(new WindowAdapter() {
   @Override
    public void windowClosed(WindowEvent e) {
       System.exit(0);
    }});
```

Pokud potřebujeme použít stejné akce na více komponentách, je výhodnější napsat si vedle třídu, která bude implementovat daný listener, nebo dědit od adaptéru, než psát ke každé komponentě anonymní posluchače.

3.4.1 ActionListener

Událost tohoto typu se přiřazuje nejčastěji tlačítkům, ale používá se i JComboBoxů a dalších komponent.

```
Button1 = new JButton("Vlož");
Button1.addActionListener(new ActionListener() {
        public void actionPerformed(ActionEvent evt) {
            //akce které se mají spustit při stisknutí tlačítka
 }});
```

`ActionListener` má pouze jednu metodu `actionPerformed()`.

3.4.2 KeyListener

KeyListener je událost, která registruje, jak se komponenta zachová při stisknutí klávesy.

```
Button1.addKeyListener(new KeyAdapter() {
   //stisknutí klávesy
   public void keyPressed(KeyEvent evt) {
      //TODO
   }
   //uvolnění klávesy
   public void keyReleased(KeyEvent evt){
      //TODO
   }
```

```
        //při po stisknutí a uvolnění
        public void keyTyped(KeyEvent evt){
           //TODO
        }});
```

KeyListener obsahuje tři metody a vzhledem ke zde použitému `new KeyAdapter`, by nemusely být implementovány všechny tři.

Jaká klávesa byla stisknuta, zjistíme jednoduše přes `evt.getKeyCode()`, který vrací klávesu zastoupenou číselnou hodnotou. Poté stačí porovnat, zdali se klávesa shoduje.

```
int key = e.getKeyCode();
//VK_RIGHT (šipka do prava)
if(key == KeyEvent.VK_RIGHT){
   //TODO
}
```

KeyListener bohužel automaticky nefunguje u komponent, jako je JPanel nebo JLabel. Zde si, třeba při kliknutí myši, musíme vyžádat focus pomocí metody `requestFocus()`.

3.4.3 MouseListener

`MouseListener` implementuje pět povinných metod:

- `mouseClicked` - co se stane při kliknutí tlačítka myši na komponentě
- `mousePressed` - co se stane pouze při stisknutí tlačítka myši na komponentě
- `mouseReleased` - co se stane pouze při uvolnění tlačítka myši na komponentě
- `mouseEntered` - co se stane při najetí myši na komponentu
- `mouseExited` - co se stane, když myš komponentu opustí

```
Button1.addMouseListener(new MouseAdapter() {
          @Override
             public void mouseClicked(MouseEvent e) {
                 if(SwingUtilities.isLeftMouseButton(e)){
                    //TODO
```

```
            }
        }
    });
```

Metoda se volá na jakékoliv tlačítko na myši. Pokud chceme omezit průběh metody na určité tlačítko, musíme do těla metody zahrnout podmínku, jež určí, které tlačítko myši je použité, a to jedním z následujících způsobů:

- `SwingUtilities.isLeftMouseButton(e)`
- `SwingUtilities.isRightMouseButton(e)`
- `SwingUtilities.isMiddleMouseButton(e)`

Opět můžeme nadefinovat celou novou třídu, která bude implementovat MouseListener se všemi metodami nebo použít MouseAdapter, pokud nepotřebujeme většinu metod.

Další možností je zjištění akce z eventu, která vrátí tlačítko reprezentované číselnou hodnotou, kde:

- 1 = levé tlačítko
- 2 = kolečko myši
- 3 = pravé tlačítko

```
int action = e.getButton();
```

3.4.4 MouseMotionListener

Tento listener implementuje dvě metody:

- `mouseDragged` - metoda, která se volá při stisknutí a držení tlačítka myši a jejím pohybu
- `mouseMoved` - metoda, která se volá při pohybu myši nad komponentou
- `mouseReleased` - co se stane pouze při uvolnění tlačítka myši na komponentě
- `mouseEntered` - co se stane při najetí myši na komponentu

- `mouseExited` - co se stane, když myš komponentu opustí

Opět platí, že metoda je volána pro jakékoliv tlačítko myši, pokud neurčíme v podmínce jinak.

```
Button1.addMouseListener(new MouseAdapter() {
          @Override
             public void mouseClicked(MouseEvent e) {
                 if(SwingUtilities.isLeftMouseButton(e)){
                    //TODO
                 }
              }
  });
```

3.4.5 MouseWheelListener

Posluchač, který implementuje jedinou metodu a tomouseWheelMoved, která se volá při otočení kolečka myši.

```
Button1.addMouseWheelListener(new MouseWheelListener() {
    @Override
   public void mouseWheelMoved(MouseWheelEvent evt) {
        //TODO
   }});
```

3.5 Tvorba jednoduchého GUI

V této kapitole bych rád nastínil tvorbu jednoduchého programu, který bude obsahovat tlačítko, textové pole a textovou oblast. Funkčností bude, že po stisknutí tlačítka text z textového pole nahradí text v textové oblasti. Pracujeme s absolutním pozicováním.

Obrázek 7: Výstup tutoriálu

Prvním krokem je vytvořit třídu, která dědí od JFrame, a připravit si prvky, které použijeme.

```
import javax.swing.*;
import java.awt.event.ActionEvent;

public class GUI_project extends JFrame {
   private JButton button1;
   private JTextArea textArea1;
   private JTextField textField1;
   private JPanel panel1;
```

Panel bude sloužit jako kontejner pro ostatní komponenty a stane se výplní JFramu.

Dále přichází na řadu konstruktor a inicializace komponent. K základnímu nastavení JFrame postačí velikost a viditelnost.

```
   //konstruktor
   public GUI_project(){
      setTitle("GUI_project"); //Název okna
      setSize(197,151); //velikost JFrame (okna)
        panel = new JPanel(null);
         panel.setPreferredSize(new Dimension(197,151));
```

```
button1 = new JButton();
button1.setBounds(5,5,90,35);
//nastavení eventu tlačítka, které změní text
   Button1.addActionListener(new ActionListener() {
public void actionPerformed(ActionEvent evt) {
      textArea1.setText(textField1.getText());
   }
});
   textArea1 = new JTextArea();
textArea1.setBounds(5,43,180,97);
   textField1 = new JTextField();
textField1.setBounds(97,5,90,35);
```

Dalším krokem je přidání komponent na panel a samotný panel poté JFramu.

```
//konstruktor
public GUI_project(){
   setTitle("GUI_project"); //Název okna
   setSize(197,151); //velikost JFrame (okna)
     panel = new JPanel(null);
      panel.setPreferredSize(new Dimension(197,151));

   button1 = new JButton();
   button1.setBounds(5,5,90,35);
   //nastavení eventu tlačítka, které změní text
      Button1.addActionListener(new ActionListener() {
   public void actionPerformed(ActionEvent evt) {
         textArea1.setText(textField1.getText());
      }
   });
      textArea1 = new JTextArea();
   textArea1.setBounds(5,43,180,97);
      textField1 = new JTextField();
   textField1.setBounds(97,5,90,35);
```

Teď již stačí jenom přidat metodu `main()` s novou instancí třídy `GUI_project`.

Důležitou věcí při tvorbě je pořadí, ve kterém se prvky inicializují a přidávají, stejně tak by mělo být zviditelnění okna jakožto poslední operace v konstruktoru.

4 GUI pro BlueJ

BlueJ je vývojové prostředí především pro začínající programátory jako součást projektu pro výuku objektově orientovaného programování v Javě. Disponuje jednoduchým a přehledným prostředím, kde byl kladen důraz na vizualizaci a interakci technik k vytvoření vysoce interaktivního prostředí, které podněcuje experimentování a zkoumání.

I přes jeho jednoduchost, disponuje vším potřebným od zabudovaného editoru, kompilátoru až po testovací třídy.

Jeho momentální nevýhodou je občas nefunkční našeptávač kódu, což při vytváření grafického uživatelského rozhraní zdržuje a znesnadňuje práci. Proto bych chtěl touto cestou pomoci začínajícím programátorům tak, aby si jednoduše navrhli aplikaci, ozkoušeli různá nastavení a na přehledně vygenerovaném kódu se naučili základní vlastnosti, které je možné komponentám nastavovat.

I přesto, že se jedná o poměrně nové vývojové prostředí, existuje již řada doplňků od lepší UML[3] vizualizace přes Lego NXT až po práci s 3D grafikou. Bohužel, neexistuje doplněk, který by se zabýval problematikou tvorby GUI.

4.1 BlueJ API

BlueJ disponuje BlueJ Extensions API. Jedná se o rozhraní, které umožňuje připojení vlastního doplňku přes tzv. proxy objekty[4] , se kterými manipulujeme za pomoci několika obalových tříd[5].

Pro funkčnost musíme do projektu importovat BlueJ balíček, který je dostupný ve složce /BlueJ/lib/bluejext.jar. Po připojení tohoto balíčku do projektu, musíme provést import do tříd, kde budeme pracovat s BlueJ objekty v mém případě přímo s celým prostředím BlueJ.

[3]Unified Modeling Language

[4]Jedná se o obalovací objekt, který řídí přístupy k metodám třídy, jež obaluje

[5]http://bluej.org/doc/extensionsAPI

```
import javax.swing.*;
import bluej.extensions.*;
```

BlueJ při spuštění inicializuje nainstalovaná rozšíření tak, že se podívá do instalační složky a připojí je pomocí speciálně vytvořené třídy. Tato třída neobsahuje metodu `main()` a musí být uvedena jako Main-class[6] v manifestu při kompilaci projektu, což se nastavuje obvykle ve vlastnostech projektu v daném IDE, nebo při kompilaci z příkazové řádky.

```
public class GUIExtension extends Extension{
   //Pokud je tato metoda zavolána, může se pracovat s doplňkem
   @Override
    public void startup (BlueJ bluej) {
       bluej.setMenuGenerator(new MenuBuilder());
   }
```

Jak můžeme vidět, naše hlavní třída musí dědit od třídy Extension a přepisuje metodu `startup()`.

Existují i další metody, které se mohou přepsat jako například:

- `isCompatible`
- `getVersion`
- `getName`
- `getDescription`
- `mouseExited`

Jedná se o metody, které vrací pouze informace o doplňku jako verzi, název a popis. Tyto informace jsou poté dostupné v prostředí BlueJ v menu *„Help“* a *„Instaled Extension“*, kde se po otevření informačního okna klikne na znak otazníku (viz obr. 15), který zobrazí informační okno (viz obr. 16).

Dalším krokem je dostání doplňku do menu prostředí. Extension API dovoluje přidat doplněk tam, kam potřebujeme. Pro naše účely je to menu „Nástroje“ (Tools) a to pomocí menu generátoru.

[6]Hlavní třída, která většinou obsahuje metodu main

V našem případě, jsme vytvořili vlastní třídu, která dědí od třídy `MenuGenerator`, která je součástí BlueJ API a její hlavní metoda `getToolsMenuItem` vrací komponentu typu `JMenuItem` s nadefinovanou akcí a zajišťuje, že se tlačítko zobrazí právě v menu Tools.

```
class MenuBuilder extends MenuGenerator {
    private ToolsAction aToolsAction;

    MenuBuilder() {
  //nová akce, která bude přiřazena tlačítku v menu
    //a ponese zde zadaný název
        aToolsAction = new ToolsAction("Simple GUI Extension");
    }

    public JMenuItem getToolsMenuItem(BPackage aPackage) {
        return new JMenuItem(aToolsAction);
    }

    class ToolsAction extends AbstractAction {
        public ToolsAction(String menuName) {
            putValue(AbstractAction.NAME, menuName);
        }
  //akce po stisknutí tlačítka v menu
        public void actionPerformed(ActionEvent anEvent) {
            SwingUtilities.invokeLater(new Runnable() {
                public void run() {
                 new GuiExtensionWindow(bluej);

                }
            });
        }
    }
}
```

Tímto jsme dostali doplněk do menu prostředí a zároveň si předali objekt BlueJ pro další práci s ním, jako získání cesty k balíčku, či přenačtení projektu při vytvoření nové třídy.

```
String path;
//cesta k balíčku
path = bluej.getCurrentPackage().getProject().getDir().getPath();
//přenačtení
bluej.getCurrentPackage().reload();
```

Toto API určitě stojí za podrobnější prozkoumání a zjištění plných možností, nicméně v této práci jsem si vystačil s opravdovým málem.

Dále jsou k dispozici balíčky:

- `bluej.extension.editor`
- `bluej.extension.event`

Ty umožňují práci s vestavěným editorem a sledování určitých událostí, které v BlueJ probíhají.

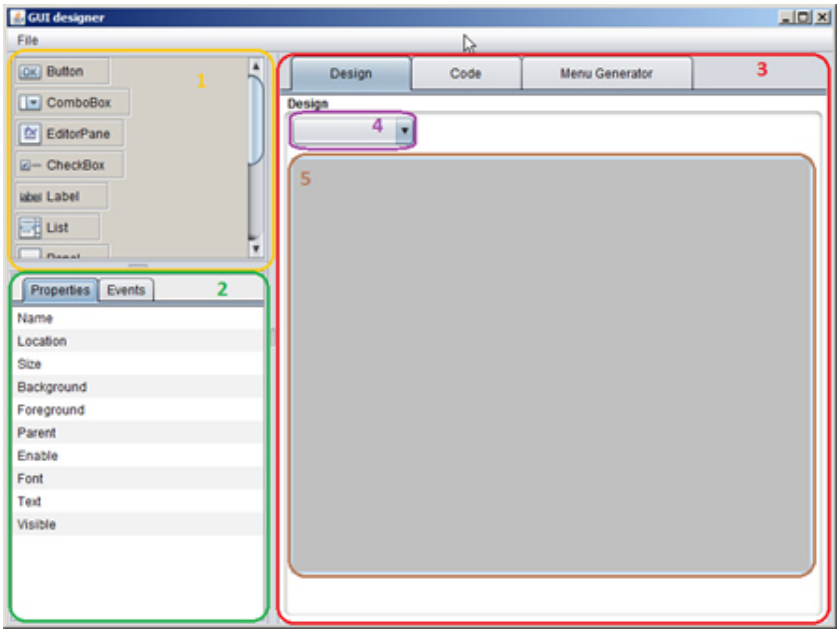
4.2 Vývoj modulu

Vývoj modulu byl rozdělen do několika kroků:

- Výběr komponent a událostí, které se použijí
- Návrh rozložení ovládacích prvků (designu aplikace)
- Prozkoumání BlueJ Extension API
- Programování dílčích částí aplikace
- Sestavení aplikace
- Testování

Během vývoje jsem narazil na několik problémů, které jsem nebyl dopředu schopen odhadnout, a tak se část návrhu měnila za chodu.

4.2.1 Grafické rozhraní doplňku



Obrázek 8: Grafické rozhraní aplikace

1. Dostupné komponenty
2. Tabulka pro editaci vlastností a eventů
3. Panel umožňující přepínání mezi designem, náhledem kódu a návrhem menu
4. Combo Box pro výběr vložené komponenty
5. Plátno pro vkládání komponent

4.3 Dokumentace

Pro správnou funkčnost je zapotřebí mít nainstalováno minimálně JDK[7] 1.7 a prostředí BlueJ.

4.3.1 Instalace

Celá aplikace je distribuována jako GUI_Extension.jar soubor. Vzhledem k výše zmíněnému BlueJ API je instalace jednoduchým nakopírováním do jedné z možných složek podle operačního systému[8].

4.3.1.1 Pro všechny uživatele:

- `<BLUEJ_HOME>/lib/extensions` (Unix)
- `<BLUEJ_HOME>\lib\extensions`(Windows)
- `<BLUEJ_HOME>/BlueJ.app/Contents/Resources/Java/extensions` (Mac OS)

Pro jednotlivé uživatele:

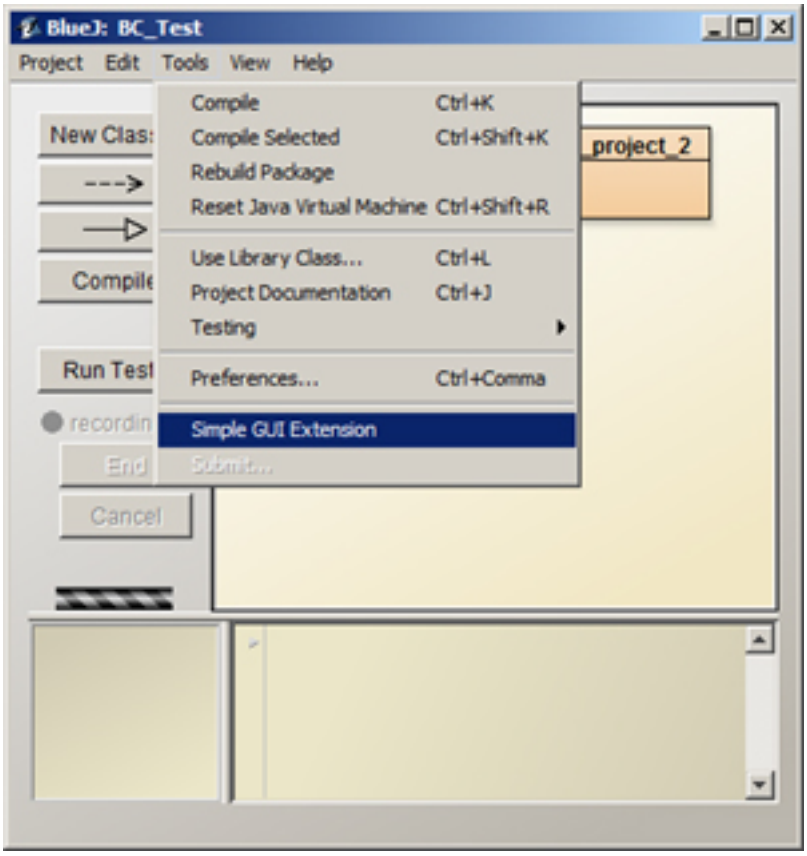
- `<USER_HOME>/.bluej/extensions`(Unix)
- `<USER_HOME>\bluej\extensions`(Windows)
- `<USER_HOME>/Library/Preferences/org.bluej/extensions`(Mac OS)

[7] Java Development Kit

[8] http://bluej.org/extensions/extensions.html

4.3.2 Spuštění aplikace

Po spuštění prostředí BlueJ je doplněk přístupný v menu Tools (Nástroje) pod názvem „Simple GUI Extension“.

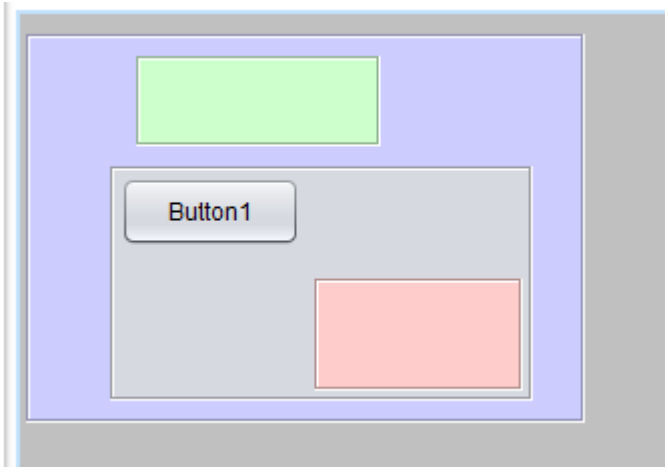


Obrázek 9: Spuštění doplňku

4.3.3 Vkládání a mazání komponent

Komponenty se vkládají pomocí drag and drop na hlavní plátno. Jedná se o JLayeredPane, tedy o vrstvený panel a každá vložená komponenta je o jednu vrstvu výše. Komponenty jsou vkládány na nulový layout a jsou absolutně pozicovány.

Aplikace dovoluje vkládat komponenty do sebe, to znamená, že pokud vložíme panel, můžeme do něj vkládat další komponenty, včetně opětovného panelu.



Obrázek 10: Vkládání doplňku

Dostupné komponenty:

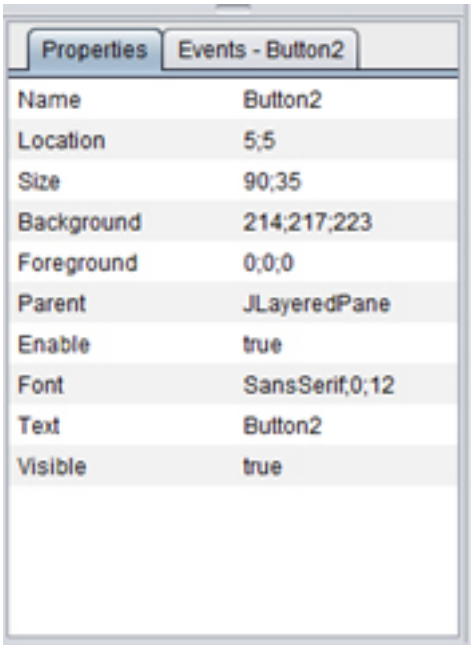
- JButton
- JComboBox
- JEditorPane
- JCheckBox
- JLabel
- JList
- JPanel
- JPasswordField
- JRadioButton
- JTextArea
- JTextField

Smazání komponenty je možno udělat dvěma způsoby:

- Zmáčknutí klávesy DELETE
- Přes pop-up menu na pravém tlačítku myši (viz obr.17)

4.3.4 Nastavení vlastností komponent

Po vložení komponenty na plátno se aktualizuje tabulka vlastností (viz obr. 11).



Obrázek 11: Tabulka vlastností

Tabulka obsahuje možnost editace základních vlastností, jako jsou:

- Jméno komponenty – dále použito při generování kódu jako název proměnné
- Umístění komponenty na plátně
- Velikost komponenty
- Barvu pozadí komponenty
- Barvu popředí komponenty
- Informaci o rodičovském prvku
- Možnost povolení nebo zakázání komponenty
- Nastavení fotu písma, jeho velikosti a řezu
- Text komponenty pokud to komponenta dovoluje
- Viditelnost komponenty

Vlastnosti velikost, font, barva pozadí a popředí se zadávají hodnotami oddělených středníkem. Font je tvořen názvem fontu, typem řezu písma a jeho velikostí.

Povolení a zakázání komponenty je umožněno kliknutím v buňce tabulky, stejně tak i zviditelnění a skrytí dané komponenty.

Pokud komponentu zneviditelníme a klikneme jinam, těžko už ji najdeme, a proto máme k dispozici Combo Box s přehledem komponent, kde danou komponentu najdeme. Při výběru komponenty se zobrazí její vlastnosti v tabulce, takže ji můžeme znovu zviditelnit.

Veškerá změna se ihned projeví aktualizací dat v tabulce vlastností.

Posun komponenty

Některé vlastnosti je možné nastavit pomocí myši, nebo kláves. Samozřejmostí je posun komponenty po plátně a to dvěma způsoby:

1. Táhnutím myší - projeví změnou kurzoru (viz obr. 18)
2. Po kliknutí na komponentu pomocí klávesových šipek

Změna velikosti komponenty

Při najetí myši do spodního pravého rohu komponenty se ukáže úchopový bod. V té chvíli je možné komponentu tahem zmenšovat nebo zvětšovat(viz obr. 19).

Změna barvy pozadí a popředí

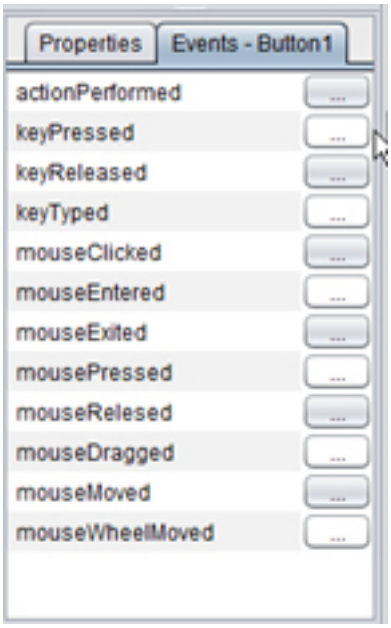
Tato funkce je dostupná přes menu na pravém tlačítku myši (viz obr. 17). Po stisknutí se zobrazí tzv. Color Chooser, ve kterém je hned několik možností výběru barvy.

4.3.5 Nastavení eventů

Eventy jsou nedílnou součástí ovládacích prvků, jak jsme si již vysvětlili (viz kapitola 3.4). V mé aplikaci jsem použil ty základní, které jsou v oné kapitole zmíněny a jež se dají použít na většinu komponent.

- ActionListener
- MouseListener
- MouseMotinListener
- MouseWheelListener
- KeyListener

Po označení komponenty stačí přepnout v tabulce vlastností na druhý panel s názvem Events a názvem komponenty, pro kterou se budou události registrovat.

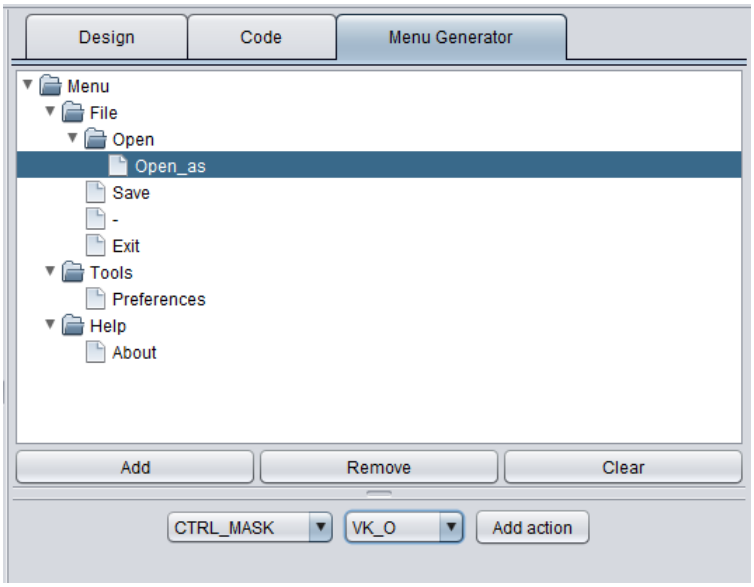


Obrázek 12: Tabulka eventu

Po vybrání události, kterou chceme komponentě přiřadit, stiskneme tlačítko a objeví se okno s dialogem, kam vložíme název metody, jež se bude provádět při dané akci. Pokud by byl název metody již použit, aplikace to oznámí.

4.3.6 Tvorba menu

Aplikace dovoluje generovat menu a to za pomoci stromové struktury.

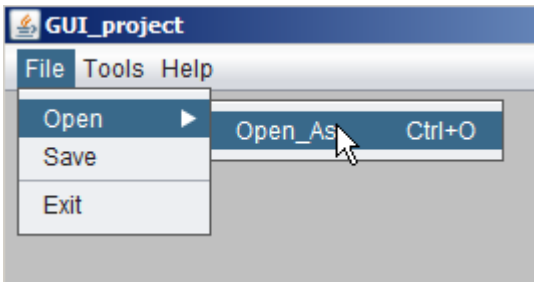


Obrázek 13: Stromová struktura tvorby menu

Tato struktura byla vybrána z důvodu přehlednosti a jednoduché obsluhy. Každý název by měl v menu existovat pouze jednou. Pokud při vložení bude zaznamenáno, že položka menu již existuje (velká a malá písmena nehrají rozdíl), bude uživatel upozorněn pomocí zprávy (viz obr. 20).

Položky menu lze přejmenovat a to buď jedním kliknutím a vyčkáním, pomocí klávesy F2, nebo přes menu na pravém tlačítku myši.

Dále stačí koncovým tlačítkům nastavit klávesové zkratky pomocí vybrání masky, jako je CTRL, ALT nebo SHIFT klávesa spolu s akční klávesou. Při změně se automaticky provede uložení.



Obrázek 14: Vygenerované menu

4.3.7 Generování kódu

Ke generování kódu dochází ve dvou případech.

1. Při ukládání kódu
2. Při změně záložky na náhled kódu

Generování kódu je jednostranná záležitost. Je celkem obtížné rozparsovat (rozdělit) kód, aby se změna projevila v designu, a proto je mnohem jednodušší upravit vzhled v designu a vygenerovat znovu kód.

Základem generování kódu je název plátna, na které se komponenty vkládají. Jeho název je názvem celého projektu, jakož i uložené třídy.

Generovaný kód (viz příloha 7.2) obsahuje všechny pravděpodobné importy, které je nutné k chodu aplikace, dále se generují proměnné, po kterých následuje konstruktor třídy, v němž se proměnné inicializují a nastavují se jim eventy. Dále se vygenerují hlavičky tříd pro eventy a po nich následuje metoda pro generování menu. Nakonec je vygenerována metoda `main()`.

Spolu s inicializací komponent v konstruktoru se nastavují veškeré vlastnosti, které jsou v tabulce vlastností dostupné, a to z důvodu ukázky zápisu kódu.

```
Button1 = new JButton();
Button1.setBounds(5,5,90,35);
Button1.setBackground(new Color(214,217,223));
Button1.setForeground(new Color(0,0,0));
Button1.setEnabled(true);
Button1.setFont(new Font("sansserif",0,12));
Button1.setText("Button1");
Button1.setVisible(true);
```

Některé vlastnosti by vůbec nemusely být použity, ale vzhledem k tomu, že aplikace má sloužit začínajícím programátorům, je na místě ukázat kompletní zápis kódu. Dále s ním mohou experimentovat dle svého uvážení.

4.3.8 Uložení kódu

Kód lze uložit přes nabídku *„File“* a *„Save“* v panelu nástrojů nebo pomoci klávesové zkratky CTRL + S. Třída se ukládá podle názvu hlavního plátna. Pokud již taková třída existuje, objeví se upozorňující hláška.

Po uložení kódu se přenačte prostředí BlueJ a uložená třída se objeví ve vizualizační části. Poté stačí třídu zkompilovat [9] a spustit.

V momentálním stavu, není aplikace schopna uložit stávající projekt jako takový.

4.4 Popis stěžejních tříd doplňku

V této části bych se rád věnoval hlavním třídám použitých v aplikaci a objasnil funkčnost určitých prvků. Přehled stěžejních tříd aplikace je dostupný v příloze jako class diagram (viz příloha 7.3).

GuiExtensionWindow

Jedná se o třídu, která se stará o uživatelské rozhraní aplikace a její ovládání. Obsahuje tedy základní komponenty, jako JLayeredPane canvas, na který se vynáší komponenty, dále JSplitPane panely, které rozdělují okno aplikace, jakož to umístění tabulek vlastností, eventů a záložkových panelů.

```
canvas = new JLayeredPane();
```

Obsahuje i nastavení pro Look and feel aplikace, které bylo nastaveno na Nimbus, pokud je dostupný, jinak na systémový vzhled. Zde jsem při používání záložkových panelů zjistil, že je možno pro editaci velikosti a okrajů záložek využívat i HTML[10] kód [16].

```
tabbedPane.addTab("<html><body leftmargin=15 topmargin=8"
    +" marginwidth=30 marginheight=5>Design</body></html>", rightMainPanel);
```

Dále třída obsahuje i kolekci TreeMap, kde je klíčem řetězec, název komponenty, a objektem je třída EventsInit, která nese informace o eventech pro danou komponentu.

```
private TreeMap<String,EventsInit> componentEvents;
```

[9] převod algoritmu zapsaného v programovacím jazyce do strojového kódu
[10] HyperText Markup Language

Další důležitou součástí je třída TransferHendler, která zajišťuje přenos mezi komponentami typu Transferable, což je většina komponent z balíčku Swing.

```
private TransferHandler tr = new PanelHandler();
```

`PanelHandler` je třída, která dědí od `TransferHandler` a zajišťuje vkládání komponenty na panel. Přepisuje dvě metody:

- `canImport()` – která rozhoduje, jestli komponenta přijme datový kopírovaný datový typ
- `importData()` - která se stará o vložení importované komponenty na plátno a nastavení událostí.

Tato instance se poté předává i panelu, který je vkládán, aby na něj bylo možno opět vkládat.

```
canvas.setTransferHandler(tr);
```

Komponentám s drag and drop akcí, musíme nastavit nastavit `TransferHandler`, který v sobě uchovává akci, jež se předává na cílový panel.

```
label.setTransferHandler(new ComponentHAndler(originalFN));
class ComponentHAndler extends TransferHandler{
          private String text = "";
          public ComponentHAndler(String text){
              this.text = text;
          }
   @Override
        protected Transferable createTransferable(JComponent c) {
           return new StringSelection(text);
           }
...
}
```

Zde je důležitá metoda `createTransferable()`, která vrací, co se přenáší v DnD[11] akci. V našem případě se jedná o textovou hodnotu.

[11] Drag and Drop

Když už máme nastavenou akci a komponentě i panelu přiřazen `TransferHandler`, musíme dále DnD komponentě říci, jak onu informaci předat.

To zajišťuje metoda `MouseHandler`, která dědí od `MouseAdapteru`.

```
class MouseHendler extends MouseAdapter{
        public void mousePressed(MouseEvent e){
          JComponent c = (JComponent)e.getSource();
          TransferHandler handle = c.getTransferHandler();
          handle.exportAsDrag(c, e, TransferHandler.COPY);
        }
}
```

Na poslední řádce vidíme, že se předává text, který obsahuje TransferHandler komponenty a prováděná akce je kopírování.

Součástí této třídy je i metoda `setDraggableComponents()`, které načte dostupné komponenty podle názvu ikony a přiřadí jim akci a `TransferHandler`. Bohužel je problém dostat se dynamicky k obrázkům v *„jar“* souboru, a tudíž musí být názvy ikon zaneseny staticky v poli. Vzhledem k nízkému počtu komponent to zde není problém.

CanvasAddComponent

Třída, která se stará o přidání konkrétní komponenty na určitý panel a to podle akce, která se vrací při puštění myši.

Akce je typu String, tedy text, který říká, jaká komponenta byla požadována a pomocí přepínače se tato komponenta vloží.

```
public JComponent addComponentOnCanvas(String component, Point point,
    TransferHandler tr){
        switch(component){
            case "Button":
                JButton button = new JButton("Button"+buttonName);
                button = (JButton) setComponents(button, point,
                    BUTTONS_DIMENSION);
                name=component+""+buttonName;
                setComponentInCanvas(button, name);
                buttonName++;
```

```
            return button;
```

Metoda `addComponentOnCanvas()` vytvoří komponentu, nastaví její jméno, které přiřadí v metodě `setComponentInCanvas()`, zvýší čítač komponenty o jedna a vrátí komponentu.

MouseActionEventClass

Jedná se o třídu, která implementuje `MouseListener`, a stará se tedy o akce na tlačítka myši, jako update tabulky vlastností po označení jiné komponenty, zvýraznění ohraničení komponenty při najetí myší na komponentu a zobrazení pop-up menu na prvé tlačítko myši.

MouseMoveEventClass

Jedná se o třídu, jež implementuje `MouseMotionListener` a stará se o změnu velikosti komponenty při tahu myší a zobrazení úchopového bodu při najetí do pozice.

```
if( e.getX() > (x-RESIZE_AREA) && e.getX() <= (p.x+RESIZE_AREA) && e.getY() >
    (y-RESIZE_AREA) && e.getY() < (p.y+RESIZE_AREA) ){
    comp.setSize(e.getX(),
   table.getTable().setValueAt(comp.getSize().width+";"+comp.getSize().height,
      2, COLUMN);
    parent.repaint();
}
```

Podmínka výše je součástí metody `mouseDragged()` a kontroluje, jestli se myš stále nachází v oblasti, kde je povelena změna velikosti při tahu myší. Pokud ano, změní velikost komponenty a aktualizuje tabulku vlastností.

EventsTable

Třída zajišťující tabulku eventů. Obsahuje akci pro tlačítko [15], jež nastavuje a ukládá název metody, která bude vygenerovaná jako event.

```
name = JOptionPane.showInputDialog("Enter method name",
       componentEvents.get(component).getActionValue(rowIndex));
```

PropertiesTable

Třída, jež se stará o tabulku vlastností. Obsahuje v sobě instanci JTableX, která má možnost jiných tzv. cell renderů, což je metoda, která vykresluje obsah buňky tak jak chceme [14].

Obyčejná tabulka má konstantní datový typ v celém sloupci a to kolikrát není žádoucí. Proto se musí psát další metody, které tuto možnost rozšiřují a dělají z tabulky efektivní nástroj.

Tabulka pracuje s tzv. modelem, který právě určuje, co je v jaké buňce. Na model se aplikuje `TableModelListener`. Ten zajišťuje aktualizaci dat při změně hodnoty buňky.

```
model.addTableModelListener(getListener());
```

Zde muselo být ošetřeno, aby se údaje nezaměňovaly, protože se tento posluchač volá při každé změně hodnoty v buňce, což znamená i v případě, že klikneme na jinou komponentu.

```
if(changingValues){
     return;
}
```

Tato podmínka zajišťuje, že se nenastaví špatné údaje.

Pokud je povolena změna údajů, pak pomocí přepínače `switch(row)`, kde, `row` je číslo řádku, se provede daná akce.

```
case 4:
   Color nFc = new Color(Integer.parseInt(value[0]),Integer.parseInt(value[1]),
           Integer.parseInt(value[2]));
   comp.setForeground(nFc);
   map.get(oldComponentName).setForeground(nFc);
   break;
```

Tento přepínač nastavuje barvu popředí komponenty, kde v poli value[0] je hodnota červené, value[1] zelené a value[2] modré barvy. Jak jsem již zmiňoval v kapitole 4.3.4, dochází k parsování textového řetězce z hodnoty v tabulce.

KeyActionEvent

Třída, která zajišťuje pohyb komponent pomocí klávesových šipek, a pomocí klávesy DELETE komponenty maže.

```
public void keyPressed(KeyEvent e) {
  currX = comp.getLocation().x;
  currY = comp.getLocation().y;
  int key = e.getKeyCode();
  switch(key){
    case KeyEvent.VK_RIGHT:
        comp.setLocation(++currX, currY);
     propertiesTable.getTable().setValueAt(comp.getLocation().x
            +";"+comp.getLocation().y, 1, COLUMN);
        break;
```

Jaká klávesa byla stisknuta, se rozhoduje pomocí přepínače a podle toho se komponenta posune.

ComponentMover

Třída, která má na starost pohyb komponenty při pohybu myší [13]. Defaultním nastavením se nedá vyjet z plátna, komponenta je omezena hranicemi rodičovského prvku, nicméně se nechají nastavit okraje, a to buď záporně, nebo kladně.

Nicméně postačí ohraničení o pět pixelů tak, aby komponenta neležela na úplném okraji okna.

```
  private Insets edgeInsets = new Insets(5, 5, 5, 5);
```

Tato instance určuje, kde jsou hranice pro uchopení komponenty.

```
  private Insets dragInsets = new Insets(0, 0, 0, 0);
```

MenuPanelGenerator

Třída zajišťující panel, na kterém bude vykreslen strom pro generování menu, spolu s Combo Boxy pro nastavení klávesových zkratek a akce.

DynamicTreeDemo

Modifikovaná třída převzatá z tutoriálu, spolu s třídou `DynamicTree` pro práci se stromovou strukturou v Javě, která zajišťuje vykreslení tlačítek pod stromem a nastavení jejich akcí. Dále provádí první naplnění stromu [23].

DynamicTree

Třída, která v sobě uchovává strom typu JTree a doplňující metody pro práci s ním. Uzly stromu jsou zastoupeny třídou `TreeMenuItem` (viz níže).

- `getCurrentNode()` – vrací zvolený uzel
- `reloadAndExpand()` – přenačte strom a rozvine ho
- `addObject(TreeMenuItem child)` – přidání prvku (tato metoda má několik modifikací
- `removeCurrentNode()` – odstranění konkrétního uzlu
- `clear()` – smazání celého stromu, kromě kořenového uzlu

Strom má nastaven `TreeSelectionListener`, který říká, co se stane při změně výběru uzlu stromu, jako nastavení Combo Boxů s klávesovými zkratkami podle dané hodnoty, či jejich znepřístupnění, pokud se nejedná o koncové tlačítko menu.

TreeMenuItem

Tato třída zastupuje uzel stromu a implementuje rozhraní `MutableTreeNode`, které v sobě obsahuje metody potřebné k tomu, aby třída mohla reprezentovat uzel stromu. Dále udržuje název menu položky, její klávesovou zkratku a akci na tlačítko.

Důležité metody:

- `setUserObject()` – tato metoda se volá po editaci uzlu ve stromě a nastavuje jeho jméno.
- `remove()` – metoda, která existuje v několika modifikacích a odstraňuje daný prvek.
- `insert()` – vložení prvku do stromu, opět několik modifikací
- `isLeaf()` – vrací informaci, zdali je uzel listem (nemá potomky)
- `getParent()` – vrací rodičovský prvek
- `getChild()` – vrací potomka
- `removeAllChildren()` – odstraní všechny potomky

UIHelper

Třída, jež pomáhá nastavit DnD komponenty v levé horní části. Vrací JLabel s ikonou [9].

Modul obsahuje daleko více tříd a jejich podrobnější popis je ve vygenerované dokumentaci jako součást přílohy na CD spolu s doplňkem.

5 Závěr

Pro připomenutí, byly vytyčeny tyto cíle:

- Stručně nastínění tvorby grafického uživatelského rozhraní v Javě
- Stručně popsat práci s komponentami Swing
- Popsat problematiku obsluhy událostí
- Vývoj doplňku (extensions) do vývojového prostředí BlueJ

Za sebe si dovoluji tvrdit, že se mi podařilo splnit vytyčené cíle, neboť práce se v první části zabývá problematikou tvorby GUI v Javě, i když ne příliš do hloubky, protože na toto téma již byly sepsány podrobnější práce, ale spíše jednoduše vysvětlit co, jak a proč.

Naprogramováním aplikace, se mi podařilo splnit hlavní cíl této práce. Aplikace sice není dokonalá, ani nemůže, vždyť jiná IDE měla celá léta na vývoj grafického designéru. Aplikace je dostačující pro začínající programátory, a to nejen díky lehkému a intuitivnímu ovládání, ale i díky přehlednému generování kódu.

Díky testování aplikace, které do této doby proběhlo, vím o mnoha nedostatcích, ale také o mnoha návrzích k vylepšení funkčnosti aplikace, ať se jedná o přidání dalších komponent, ukládání a načítání projektu, přidání práce s layouty a další věcí.

Tvorba doplňku mě osobně rozšířila znalosti z této problematiky a naučila mě nové věci, jako práci s BlueJ API a ukázala na určité nedostatky u grafických komponent.

Pro distribuci aplikace hodlám vytvořit webové stránky[12], které by měly mít zastoupení i mezi doplňky na oficiálních stránkách vývojového prostředí Bluej.org.

[12] http://home.pf.jcu.cz/~gbluej/

6 Reference

[1] BLUEJ.*BlueJ - Teaching Java* [online]. 2000 [cit. 2013-04-25]. Dostupné z: http://bluej.org/about/why.html

[2] Interval.cz:Naučte se Javu - grafické uživatelské rozhraní 1. JIŘÍ SEMECKÝ. *Interval.cz* [online]. 2003 [cit. 2012-04-30]. Dostupné z: http://interval.cz/clanky/naucte-se-javu-graficke-uzivatelske-rozhrani-1/

[3] Oracle Documentation: Javax Swing Documentation.*Oracle Documentation* [online]. 2003 [cit. 2013-04-25]. Dostupné z: http://docs.oracle.com/javase/1.4.2/docs/api/javax/swing/package-summary.html

[4] Oracle Documentation: Java AWT Documentation*Oracle Documentation* [online]. 2003 [cit. 2013-04-25]. Dostupné z: http://docs.oracle.com/javase/1.4.2/docs/api/java/awt/package-summary.html

[5] IBM developer works: SWT, Swing or AWT: Which is right for you?. BARRY FEIGENBAUM. *IBM developer works* [online]. 2006 [cit. 2012-04-30]. Dostupné z: http://www.ibm.com/developerworks/grid/library/os-swingswt/

[6] HEROUT, Pavel.*Java grafické uživatelské prostředí a čeština.* Vyd. 1. České Budějovice: KOOP, 2001, 316 s. ISBN 80-723-2150-1.

[7] BlueJ: The interactive Java environment. *BlueJ* [online]. 1999 [cit. 2012-04-30]. Dostupné z: http://www.bluej.org/extensions/extensions.html

[8] Java: How to Support Assistive Technologies. ORACLE.*Oracle - Documentation* [online]. 1995 [cit. 2012-04-30]. Dostupné z: http://docs.oracle.com/javase/tutorial/uiswing/misc/access.html

[9] GUY, Romain. Curious Creature: Drag with style in Swing. *Drag with style in Swing* [online]. 16.02. 2005 [cit. 2013-04-21]. Dostupné z: http://www.curious-creature.org/2005/02/16/drag-with-style-in-swing/

[10] ECKEL, Bruce Thinking in Java. 2nd ed. Upper Saddle River: Prentice Hall, c2000, 1127 s. ISBN 01-302-7363-5.

[11] BlueJ The interactive Java environment. [online]. 1999 [cit. 2013-01-20]. Dostupné z: http://www.bluej.org/doc/documentation.html

[12] BARNES, D., J., KÖLLING, M. Objects First with Java A Practical Introduction using BlueJ, 5th edition. Prentice Hall / Pearson Education, 2012. ISBN 978-013-249266-9.

[13] CAMICK, Rob. Java Tips Weblog: Moving windows.*Java Tips Weblog* [online]. 14.06. 2009 [cit. 2013-04-21]. Dostupné z: http://tips4java.wordpress.com/2009/06/14/moving-windows/

[14] COLSTON, Tony. Jawa World: Add multiple JTable cell editors per column.*Jawa World* [online]. 15.09. 2009 [cit. 2013-04-21]. Dostupné z: http://www.javaworld.com/javaworld/javatips/jw-javatip102.html

[15] CHARLES. Cordinc Blog: JButtons in a JTable*Corinc Blog* [online]. 30.01. 2010 [cit. 2013-04-21]. Dostupné z: http://www.cordinc.com/blog/2010/01/jbuttons-in-a-jtable.html

[16] XUAN, Yun. Java.net: Change the Tab Size of JTabbed-Pane*Java.net* [online]. 27.01. 2009 [cit. 2013-04-21]. Dostupné z: https://weblogs.java.net/blog/xuanyun/archive/2009/01/change_the_tab.html

[17] JELÍNEK, Lukáš. Linuxsoft.cz: Úvod do grafiky a GUI.*Linuxsoft.cz* [online]. 24.04. 2006 [cit. 2013-04-21]. Dostupné z: http://www.linuxsoft.cz/article.php?id_article=1184

[18] ORACLE Java Technology: The JComponent Class.*ORACLE Java Technology* [online]. 1995-2013 [cit. 2013-04-21]. Dostupné z: http://docs.oracle.com/javase/tutorial/uiswing/components/jcomponent.html

[19] ORACLE Java Technology: Using Layout Managers.*ORACLE Java Technology* [online]. 1995-2013 [cit. 2013-04-21]. Dostupné z: http://docs.oracle.com/javase/tutorial/uiswing/layout/using.html

[20] ORACLE Java Technology: Using top-level Containers. *ORACLE Java Technology* [online]. 1995-2013 [cit. 2013-04-21]. Dostupné z: http://docs.oracle.com/javase/tutorial/uiswing/components/toplevel.html

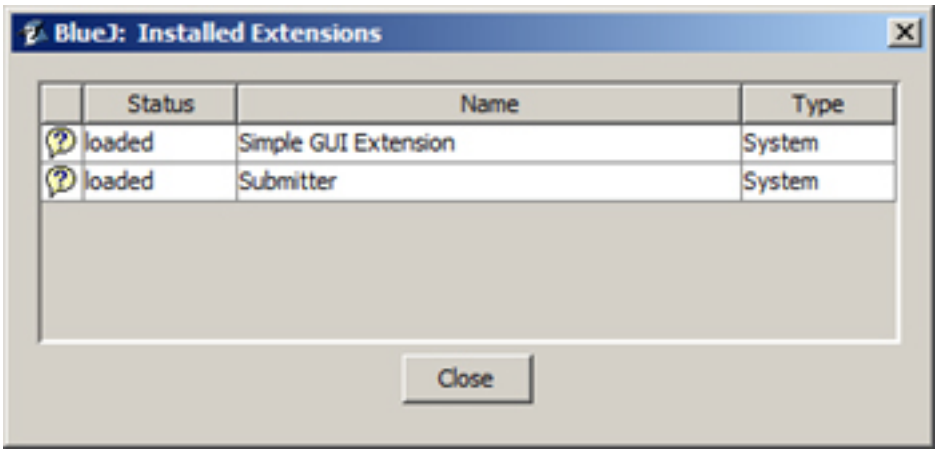
[21] ZÁVĚRKA, Jakub. Pseudo-class diagram hierarchie AWT a Swingu v Javě. *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2013-04-21]. Dostupné z: http://cs.wikipedia.org/wiki/Swing_Java

[22] Stackoverflow: What is the difference between listeners and adapters?. *Stackoverflow* [online]. 06.05. [United States: s.n., 2001 [cit. 2013-04-21]. Dostupné z: http://stackoverflow.com/questions/10470104/what-is-the-difference-between-listeners-and-adapters

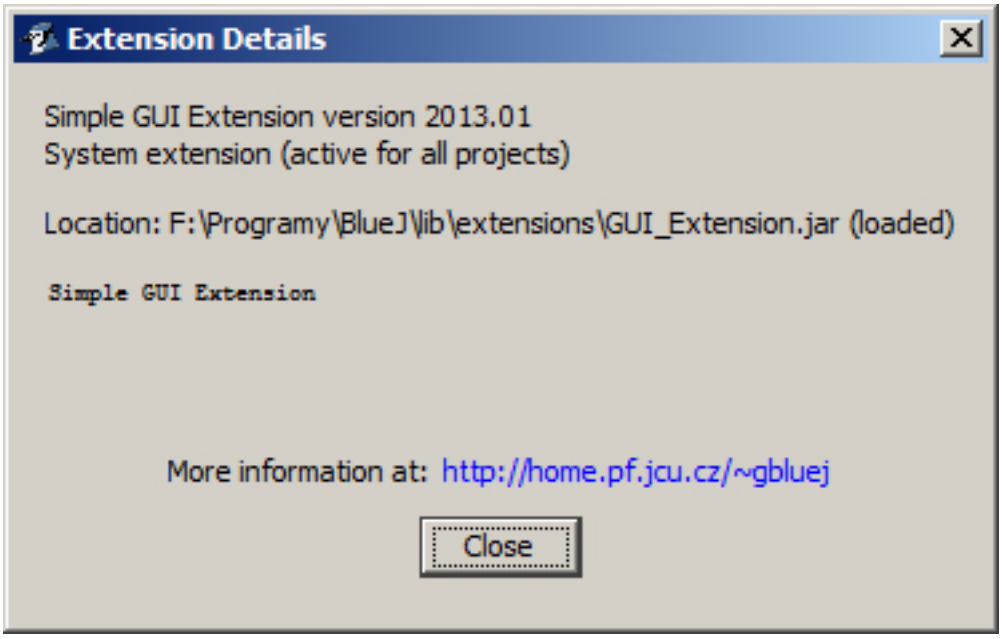
[23] ORACLE DOCUMENTATION. *How to Use Trees* [online]. 1995-2013 [cit. 2013-04-25]. Dostupné z: http://docs.oracle.com/javase/tutorial/uiswing/components/tree.html

7 Přílohy

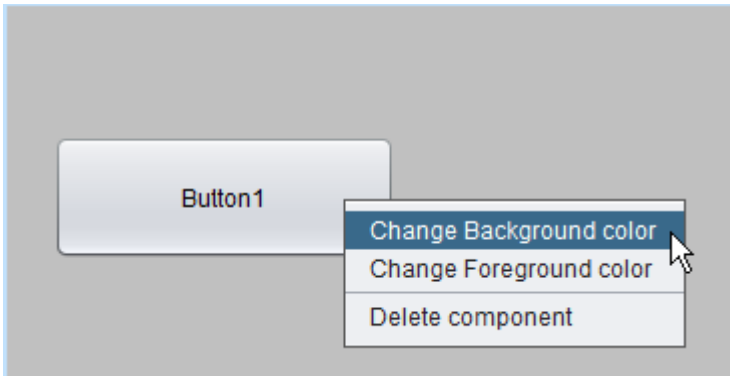
7.1 Obrázky



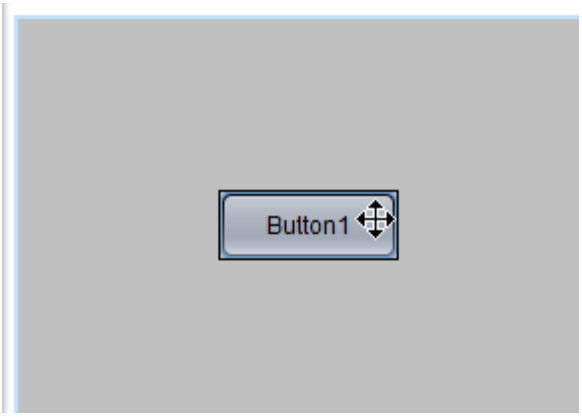
Obrázek 15: Instalovaná rozšíření



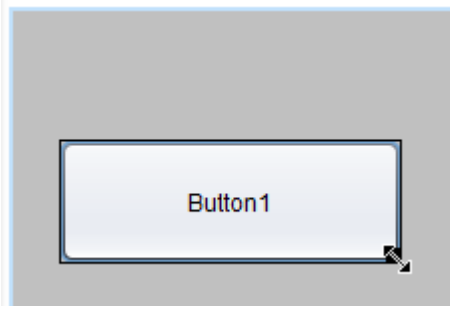
Obrázek 16: Info o doplňku



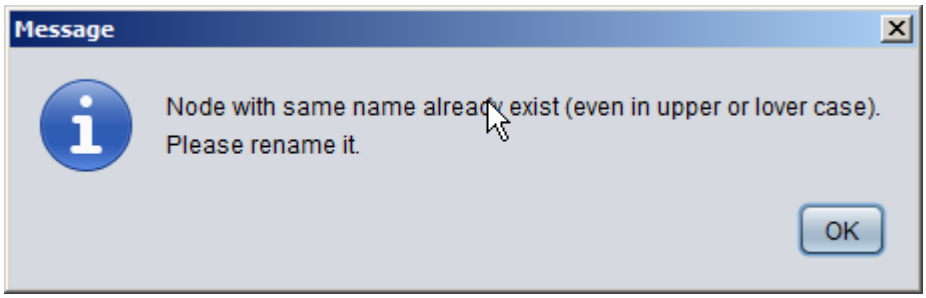
Obrázek 17: Pop-up menu



Obrázek 18: Pohyb kopmponentou

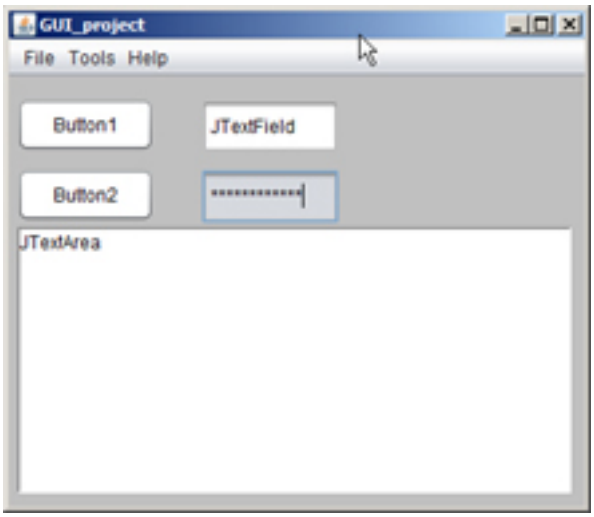


Obrázek 19: Změna velikosti



Obrázek 20: Upozornění na položku menu se stejným názvem

7.2 Ukázka zdrojového kódu vygenerovaného aplikací



Obrázek 21: Ukázková aplikace

```
     /**
*Text genereted by Simple GUI Extension for BlueJ
*/
import javax.swing.UIManager.LookAndFeelInfo;
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.event.KeyAdapter;
import java.awt.event.KeyEvent;
import java.awt.event.MouseAdapter;
import java.awt.event.MouseEvent;
import java.awt.event.MouseWheelEvent;
import java.awt.event.MouseWheelListener;
import javax.swing.border.Border;
import javax.swing.*;


public class GUI_project extends JFrame {

   private JMenuBar menuBar;
```

```
private JButton Button1;
private JButton Button2;
private JPasswordField PasswordField1;
private JTextArea TextArea1;
private JTextField TextField2;

//Constructor
public GUI_project(){

   setTitle("GUI_project");
   setSize(377,283);
   setVisible(true);

   //menu generate method
   generateMenu();
   setJMenuBar(menuBar);

   //pane with null layout
   JPanel panel = new JPanel(null);
   panel.setPreferredSize(new Dimension(377,283));
   panel.setBackground(new Color(192,192,192));


   Button1 = new JButton();
   Button1.setBounds(5,15,90,35);
   Button1.setBackground(new Color(214,217,223));
   Button1.setForeground(new Color(0,0,0));
   Button1.setEnabled(true);
   Button1.setFont(new Font("sansserif",0,12));
   Button1.setText("Vlož");
   Button1.setVisible(true);

   //Set action for button click
   //Call defined method
   Button1.addActionListener(new ActionListener() {
```

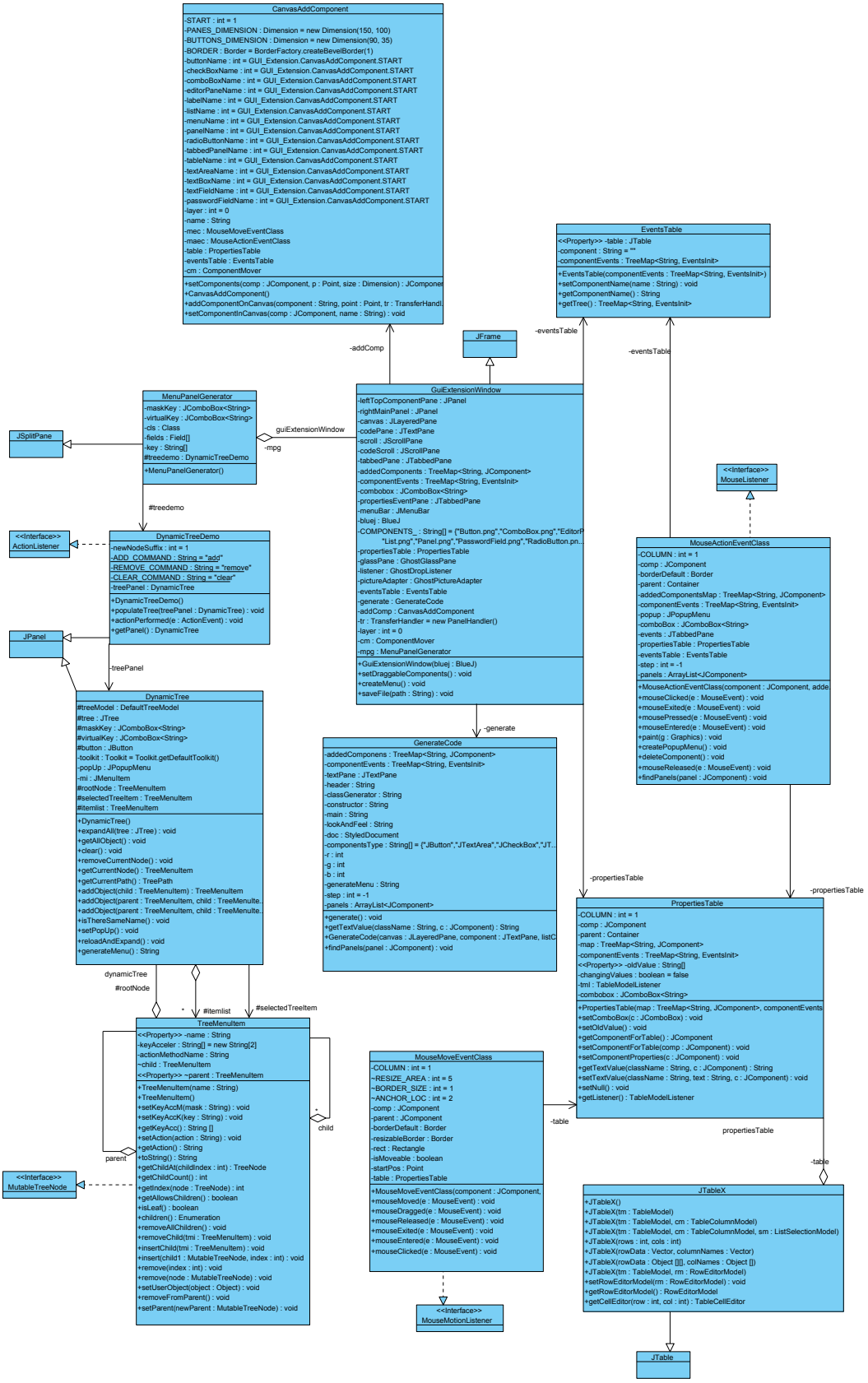
```
   public void actionPerformed(ActionEvent evt) {
      vloz(evt);
   }
});


Button2 = new JButton();
Button2.setBounds(5,61,90,35);
Button2.setBackground(new Color(214,217,223));
Button2.setForeground(new Color(0,0,0));
Button2.setEnabled(true);
Button2.setFont(new Font("sansserif",0,12));
Button2.setText("Smaž");
Button2.setVisible(true);

//Set action for button click
//Call defined method
Button2.addActionListener(new ActionListener() {
   public void actionPerformed(ActionEvent evt) {
      smaz(evt);
   }
});


PasswordField1 = new JPasswordField();
PasswordField1.setBounds(126,62,90,35);
PasswordField1.setBackground(new Color(214,217,223));
PasswordField1.setForeground(new Color(0,0,0));
PasswordField1.setEnabled(true);
PasswordField1.setFont(new Font("sansserif",0,12));
PasswordField1.setVisible(true);

TextArea1 = new JTextArea();
TextArea1.setBounds(5,99,363,175);
TextArea1.setBackground(new Color(255,255,255));
```

```
    TextArea1.setForeground(new Color(0,0,0));
    TextArea1.setEnabled(true);
    TextArea1.setFont(new Font("sansserif",0,12));
    TextArea1.setText("JTextArea");

   TextArea1.setBorder(BorderFactory.createBevelBorder(1));
    TextArea1.setVisible(true);

    TextField2 = new JTextField();
    TextField2.setBounds(126,16,90,35);
    TextField2.setBackground(new Color(255,255,255));
    TextField2.setForeground(new Color(0,0,0));
    TextField2.setEnabled(true);
    TextField2.setFont(new Font("sansserif",0,12));
    TextField2.setText("JTextField");
    TextField2.setVisible(true);

    //adding components to panel
    panel.add(Button1);
    panel.add(Button2);
    panel.add(PasswordField1);
    panel.add(TextArea1);
    panel.add(TextField2);

    //adding panel to JFrame
    getContentPane().add(panel);
    setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    pack();
}

//Method actionPerformed for Button1
private void vloz (ActionEvent evt) {
      //TODO
}
```

```
//Method actionPerformed for Button2
private void smaz (ActionEvent evt) {
      //TODO
}

//method for generate menu
public void generateMenu(){
   menuBar = new JMenuBar();

   JMenu file = new JMenu("File");
   JMenu tools = new JMenu("Tools");
   JMenu help = new JMenu("Help");

   JMenuItem open = new JMenuItem("Open ");
   JMenuItem save = new JMenuItem("Save ");
   JMenuItem exit = new JMenuItem("Exit ");
   JMenuItem preferences = new JMenuItem("Preferences ");
   JMenuItem about = new JMenuItem("About ");


   file.add(open);
   file.add(save);
   file.addSeparator();
   file.add(exit);
   tools.add(preferences);
   help.add(about);

   menuBar.add(file);
   menuBar.add(tools);
   menuBar.add(help);
}
```

```
    public static void main(String[] args){
      System.setProperty("swing.defaultlaf",
         "com.sun.java.swing.plaf.nimbus.NimbusLookAndFeel");
      javax.swing.SwingUtilities.invokeLater(new Runnable() {
         public void run() {
            new GUI_project();
         }
      });
   }

}
```

7.3 Class diagram hlavních tříd



7.4 Příloha CD

V této příloze lze nalézt:

- zdrojové kódy aplikace
- vygenerovanou dokumentaci tzv. JavaDoc
- tuto bakalářskou práci v elektornické podobě
- videonávod pro aplikaci
- BlueJ project s třídou jakož to výsledek (z první části videa) použití doplňku.