Výuka objektového programování v prostředí Imagine

Teaching of basic of Object oriented programming in Imagine

Bakalářská práce
Mgr. Miroslav Zikmund
Vedoucí Bakalářské práce: Paeddr. Jiří Vaníček, Ph.D.
Jihočeská univerzita v Českých Budějovicích
Pedagogická fakulta
Katedra informatiky
2011

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 pedagogickou 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 Českých Budějovicích dne

Anotace

Tato bakalářská práce se zabývá výukou objektového programování v Imagine Logo. V současné době existuje mnoho objektově orientovaných programovacích jazyků (Java, C++, Python, C#, …), které se hojně využívají k vývoji programů.

Odborníci během let výuky programování přišli s několika přístupy jak vyučovat objektově orientované programování (OOP). Jedním z takovýchto přístupů je „object first“, který předkládá studentům objektový přístup od samého počátku výuky. Prostředí Imagine je pro takovýto přístup k výuce vhodnou platformou.

Imagine Logo však předkládá trochu odlišný pohled na OOP než běžné objektově orientované jazyky (Java, C#, …). K takovéto výuce jsou nutné výukové materiály, které takovýto přístup umožní. Tato práce by měla poskytnout učitelům programování určitý návod, jakým je možné postupovat při výuce OOP v Imagine Logo. Předkládá řadu úloh a žákovských listů, které studenty provedou přes nastavování implicitních vlastností objektů, přes klonování, vytváření tříd, využívání objektových procedur a proměnných až k dědičnosti. Principy objektového programování jsou objasněny na příkladech vytvořených v prostředí Imagine Logo.

Klíčová slova

Logo, Imagine, OOP, objekty, programování

Anotace

This thesis focus on object oriented programming in Imagine Logo. There is a lot of object oriented programming languages (Java, C++, C#, Python…) and they are often used for developing of programs.

Experts of OOP bring up several approaches how to teach OOP. One of them is "object first". Approach says that students should start working with objects from the first lesson. Imagine Logo is good platform for this approach.

Imagine Logo set up a little different view on OOP from common OOP languages (Java, C#…). For that kind of teaching is necessarily to have some teaching materials which allow this approach. This thesis should provide the way how teach OOP in Imagine Logo. It provides some tasks and pupils sheets that leads student through the settings of implicit information of objects, cloning, building classes, using of object procedures and variables and inheritance. I will show the principles of object programming on some examples built up in Imagine Logo.

Klíčová slova

Logo, Imagine, OOP, objects, programming

Poděkování

Rád bych poděkoval PaedDr. Jiřímu Vaníčkovi, Ph. D. za jeho podněty při tvorbě této bakalářské práce.

Obsah

PROHLÁŠENÍ **2**

ANOTACE **3**

ABSTRACT..... CHYBA! ZÁLOŽKA NENÍ DEFINOVÁNA.

PODĚKOVÁNÍ **5**

1 ÚVOD **8**

2 CÍLE PRÁCE **10**

3 METODA PRÁCE **11**

4 SOUČASNÝ STAV PROBLEMATIKY **12**

5 OBJEKTOVÉ ŘEŠENÍ IMAGINE **13**

5.1 SVĚT IMAGINE 13

5.1.1 Vytváření tříd a instancí *14*

5.1.2 Dědění *14*

5.1.3 Polymorfismus *15*

5.2 PROMĚNNÉ V IMAGINE 15

5.2.1 Neobjektové proměnné *16*

5.2.2 Objektové proměnné *16*

6 METODIKA VÝUKY OBJEKTOVÉHO PROGRAMOVÁNÍ **18**

6.1 OBECNÉ POSTUPY VÝUKY OOP 18

6.1.1 Seznámení *18*

6.1.2 Zkoumání programu *18*

6.1.3 Vlastní tvorba *18*

6.2 METODIKA VÝUKY OOP V PROSTŘEDÍ IMAGINE 19

6.2.1 Práce s jednou želvou *19*

6.2.2 Základní činnosti s více želvami *20*

6.2.3 Želvy využívající privátní informace *22*

6.2.4 Klonování objektů *23*

6.2.5 Instance jako vzor pro vytváření instancí *25*

6.2.6 Vícevrstvá hierarchie *25*

6.2.7 Vytváření nové třídy *26*

7 NAVRHOVANÝ POSTUP VÝUKY 27

7.1 ZMĚNY V NAVRHOVANÉM POSTUPU 27

7.2 OBSAH JEDNOTLIVÝCH KROKŮ VÝUKY 29

7.2.1 Co všechno želva umí 29

7.2.2 Želví kamarádi 31

7.2.3 Učíme želvy novému chování 33

7.2.4 Klonujeme želvy aneb želva jako učitel 34

7.2.5 Želva jako rodič aneb želví rodokmen 36

7.2.6 Vytváříme vlastní třídu 37

8 ZÁVĚR 39

REFERENCE 41

PŘÍLOHA A - ŽÁKOVSKÉ LISTY 43

CO VŠECHNO ŽELVA UMÍ 44

ŽELVÍ KAMARÁDI 46

KLONUJEME ŽELVY ANEB ŽELVA JAKO UČITEL 50

ŽELVA JAKO RODIČ ANEB ŽELVÍ RODOKMEN 52

VYTVÁŘÍME VLASTNÍ TŘÍDU 54

PŘÍLOHA B – CD S MATERIÁLY 56

1 Úvod

V poslední době narůstá potřeba osvojení si základních dovedností práce s výpočetní technikou. Proto byla také do Rámcového vzdělávacího programu zařazena oblast Informační a komunikační technologie (ICT) jako povinná součást základního vzdělávání na 1. a 2. stupni základních škol. Snahou je umožnit všem žákům dosáhnout základní úrovně informační gramotnosti tj. naučit se základní dovednosti v ovládání výpočetní techniky a orientovat se ve světě informací a využívat je při vzdělávání i praktickém životě.[7]

Jakmile se děti naučí základům práce s počítačem, začínají se učit vyhledávat, zpracovávat a prezentovat informace a komunikovat pomocí internetu. Kromě těchto základních témat, která jsou součástí výuky na základních a středních školách, se velké oblibě těší například počítačová grafika, tvorba WWW stránek a také programování.

Právě k výuce programování nám velice dobře může posloužit prostředí Imagine Logo. Prostředí Imagine nám nabízí odlišný pohled na programování, než klasický způsob realizovaný např. pomocí jazyku Java. Usnadňuje žákům první kroky a umožňuje v krátké době vytvářet vlastní projekty. Imagine vychází z předchozích zkušeností s prostředím Comenius Logo a umožňuje metodiku výuky algoritmizace, představivosti a řešení problémů pomocí kreslení. Takzvaná želví geometrie je ve světě ověřená a svojí pěknou grafikou poskytuje dobrou motivaci pro žáky.[8]

Kromě této metodiky výuky programování však Imagine umožňuje i další přístupy jako je objektově orientované programování. Metodický postup popsaný v kapitolách 7.1 až 7.6, poskytuje právě pro takovýto způsob výuky podkladové materiály. Metodika počítá s tím, že cílem výuky programování na základních či běžných školách není výchova budoucích programátorů, ale snaha rozvíjet ve studentech

kompetence z oblasti algoritmizace a podpora tvořivosti a projektového způsobu práce[8].

2 Cíle práce

Cílem mé práce bylo vytvoření metodických materiálů pro objektově orientované programování v prostředí Imagine Logo. Tyto výukové materiály měly obsahovat doporučení, jakým způsobem postupovat v případě, že chceme pomocí Imagine Logo vyučovat objektově orientované programování. Výuka měl být rozdělena do několika kroků, ve kterých měl být čtenář postupně seznámen s objektovým řešení problému v prostředí Imagine.

Na názorných příkladech měl být předveden přechod z v podstatě neobjektového využívání Imagine až k vytváření vlastních tříd. V jednotlivých krocích tak mělo být předvedeno využívání instančních proměnných a metod, klonování, specialita Imagine – využívání instance jako prototypu pro vytváření jiných objektů a samozřejmě dědičnost.

Tyto metodické materiály měly být setříděny a doplněny o takzvané žákovské listy, které by později mohly být nápomocny při výuce programování na základní či středních školách.

3 Metoda práce

Při studiu literatury a prvních seznamovacích krocích s objektovým řešením Imagine jsem zjistil, že přestože je Imagine řešeno objektově, není při jeho výuce možné aplikovat výukové metod používané při výuce v běžných objektově orientovaných jazycích (Java, C#,…). Proto jsem musel nastudovat problematiku programování v Imagine a porovnat jí s literaturou zabývající se výukou OOP obecně.

Tyto informace jsem utřídil, doplnil o informace získané z nápovědy prostředí Imagine a sestavil z nich teoretické podklady pro tvorbu metodického postupu.

Na základě těchto materiálů jsem vytvořil základní strukturu výuky OOP v Imagine. Takto sestavenou výuku rozdělenou do kapitol jsem doplnil o stěžejní body výuky. Ke každé kapitole jsem sestavil tzv. žákovský list [A], pomocí kterého si žáci a studenti na připraveném projektu ověří své znalosti v dané problematice. Žákovský list obsahuje kromě úkolů, vedoucích k vytvoření projektu také doplňující informace, které žákům zdůrazní základní principy.

Každý takto připravený projekt se skládá ze dvou souborů. Z prvního souboru žáci vychází při tvorbě projektu a druhý soubor uložený ve složce „Hotové_projekty“ na přiloženém CD [B] předkládá výsledné řešení.

4 Současný stav problematiky

Objektový přístup k programování se ve významnější míře objevil již před desítkami let. Využívá toho, že reálný svět, ve kterém žijeme, je založen na objektech, které vzájemně komunikují. Doby, kdy programátor potřeboval výborně ovládat algoritmizaci problémů, jsou již minulostí. Dnes je většina algoritmických problémů již vyřešena a řešení je dostupné z běžných knihoven. Základní dovedností se tak stává především správná aplikace dříve vymyšlených postupů. Zkušenosti ukazují, že největší důraz by v současnosti měl být kladen na znuvupoužitelnost a snadnou modifikovatelnost dříve vytvořených programů. Takovéto možnosti poskytuje právě objektové programování, a proto se v současnosti využívá při vývoji prakticky všech velkých projektů.[9]

Co se týče výuky objektového programování, je na našem trhu dostatečné množství knih [3][4][5]. Jde však vesměs o knihy psané především pro budoucí programátory profesionály. Kromě těchto knih je na internetu k dispozici mnoho materiálů, které radí jak postupovat při výuce objektového programování u začátečníků i pokročilých programátorů, kteří se rozhodli rozšířit své dovednosti o objektové programování. Tyto postupy nejsou aplikovatelné na prostředí Imagine, ale mohou nastínit některé možnosti.

Na druhou stranu můžeme najít publikace, které jsou primárně určené právě pro prostředí Imagine [1][2][8]. Ty sice předkládají mnoho příkladů od „želví grafiky“ až po práci s více objekty, ale nesměřují cíleně k pochopení zásad objektového programování.

Jediný materiálem předkládající možný postup při výuce OOP v Imagine, který se mi podařilo objevit je tak příspěvek pánů Kalaše a Blaha na konferenci Eurologo 2001 nazvaný Object Metaphore Helps Create Simple Logo Projects [6].

5 Objektové řešení Imagine

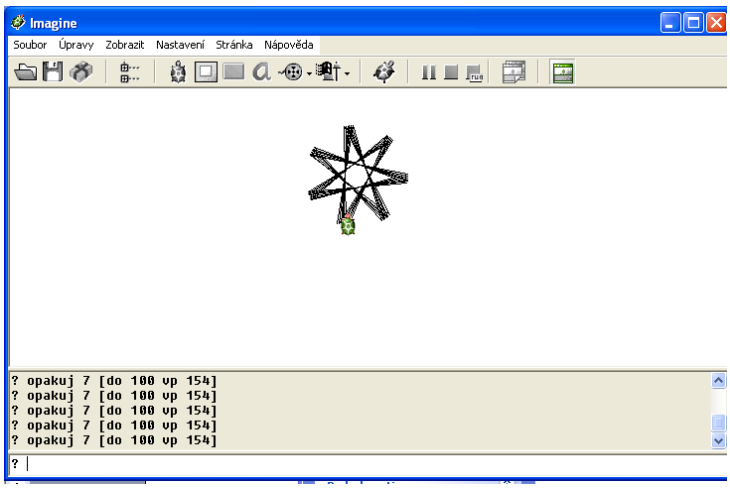
Abychom mohli verzi Loga považovat za objektově orientovanou, musí podle Kalaše a Blaha [6] obsahovat následující objektově orientované rysy:

- umožňuje vytvořit hierarchii objektů umožňující dědičnost metod
- umožňuje vytvářet objekty s jejich vlastními atributy, privátními proměnnými a metodami
- posílání zpráv, které specifikují procedury
- implementuje polymorfismus – umožňuje, aby různé objekty interpretovaly stejnou metodu různým způsobem

Imagine Logo splňuje všechny tyto požadavky.

5.1 Svět Imagine

Imagine Logo vznikl v roce 2001 jako kompletně objektově orientovaný jazyk řízený událostmi. Je navrženo tak, aby si uživatel objektové struktury vůbec nemusel všimnout a mohl využívat klasickou metodu výuky programování v Logu. Na druhou stranu poskytuje takové možnosti, které umožňují plnohodnotný objektový přístup k programování.



Obrázek 1 - Prostředí Imagine

Svět Imagine se skládá z objektů a zpráv, které můžeme objektům posílat. Všechny třídy (želva, stránka, papír, tlačítko, …) jsou podtřídami třídy Objekt.

Při vytvoření nového objektu je jeho chování určeno sloučením chování všech jeho nadtříd (neboli děděním) a chováním uloženým přímo v objektu. Objekty si zároveň vytváří vztahy s dalšími objekty. Jedním vztahem je propojení mezi objektem a jeho rodičem, dalším je propojení mezi objektem a místem, kde objekt žije (stránka, papír) a dále si objekt může propůjčit chování od nějakých dalších objektů.

5.1.1 Vytváření tříd a instancí

Imagine obsahuje několik předdefinovaných základních tříd (želva, stránka, papír, tlačítko, …). Od těchto tříd můžeme odvozovat vlastní třídy. K tomu slouží příkaz **nováTřída,** kde určíme, od jaké třídy budeme odvozovat, název nové třídy a seznam nastavení nové třídy (to je možné později upravit).

Pokud chceme vytvořit nějaký objekt (instanci) máme v Imagine dvě možnosti. Buď vytvoříme instanci nějaké třídy příkazem **? nová "názevTřídy seznamNastavení** (některé instance základních tříd můžeme vytvářet pomocí tlačítka na hlavním panelu – želva, tlačítko atd.), nebo vytvoříme instanci z instance **? nová "názevInstance seznamNastavení.** Tato možnost nám umožňuje vytvořit si vzorek, nastavit mu všechny potřebné vlastnosti a potom od něj vytvářet další objekty.

5.1.2 Dědění

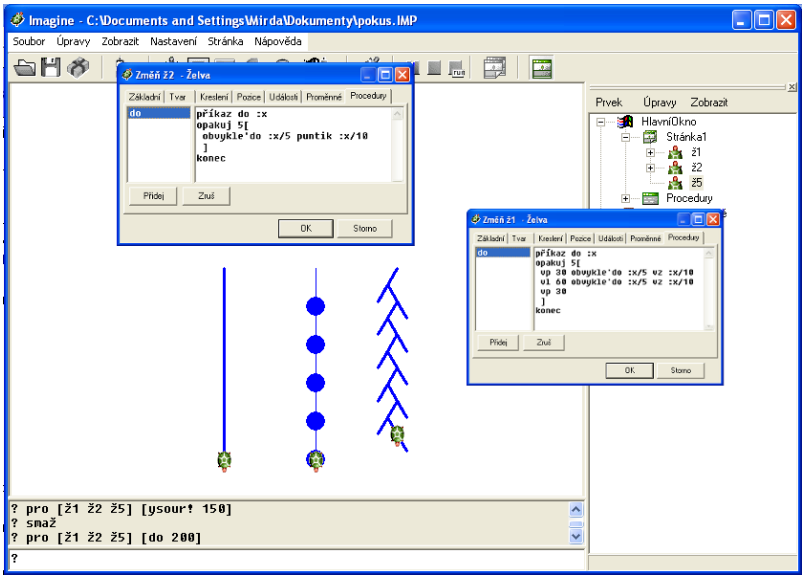
Když pošleme objektu instrukci, snaží se jí pochopit. Pokud instrukci nezná, pokouší se jí zdědit od svého předka, stejně tak to platí i pro proměnné.

Při vyhledávání procedur Imagine nejprve začne prohledávat vlastní objektové procedury, pokud proceduru nenajde, začne prohledávat jeho předky, pokud ani tam nenajde, co hledá, prohledává místo, kde objekt žije

(stránka, papír,…) a jeho předky. Pokud nenajde, co hledá, informuje o tom uživatele.

5.1.3 Polymorfismus

Imagine umožňuje překrývat procedury základních tříd. Pokud budeme například měnit způsob, jakým želva vykonává pohyb vpřed, můžeme jí definovat proceduru **dopředu** nebo zkráceně **do**. Na obrázku níže (Obrázek 2) vidíte 3 způsoby vykonání příkazu **do 200**. Pokud takto překryjeme nějakou proceduru, ale budeme chtít v nějaké situaci použít původní definici příkazu, použijeme slovo **obvykle**. Příkaz **obvykle'procedura** imagine pochopí tak, že definici procedury nezačne hledat u vlastníka běžící procedury, ale v jeho nejbližším předkovi.



Obrázek 2 - Polymorfismus v Imagine

5.2 Proměnné v Imagine

Prostředí Imagine používá dva základní typy proměnných:

- neobjektové proměnné
- objektové proměnné

5.2.1 Neobjektové proměnné

Rozdělujeme je na lokální a globální. Ke globálním proměnným můžeme přistupovat odkudkoliv a Imagine je zobrazí v okně Paměť ve složce Globální proměnné (Obrázek 3). Vytváří se příkazem **dosaď** nebo **pojmenuj**. Lokální proměnné existují, pouze dokud neskončí její proces a vytváří se příkazem **dosaďZde** nebo **lokálně**, nebo je vstupním parametrem v daném procesu.

Obrázek 3 - Globální proměnné v okně Paměť

5.2.2 Objektové proměnné

Rozdělují se na základní nastavení, vlastní proměnné a společné proměnné.

- Základní nastavení
  - Každá základní třída v Imagine má sadu základních proměnných, které budou mít také všechny od ní odvozené objekty a třídy. Ke každému základnímu nastavení existují dvě základní procedury **názevProměnné!** (s vykřičníkem) slouží ke změně nastavení a **názevProměnné** (bez vykřičníku) slouží ke zjištění aktuálního stavu.
  - **? ž1'tloušťkaPera! 7** slouží k nastavení tloušťky pera a
  - **? piš ž1'tloušťkaPera** slouží k vypsání aktuální tloušťky.

- Vlastní proměnné
    - Jsou podobné s proměnnými základními. Pokud nějakému objektu nebo třídě vytvoříme vlastní proměnnou, automaticky se pro ni vytvoří procedury pro nastavení její hodnoty a pro zjištění jejího stavu (stejně jako u základních proměnných).
- Společné proměnné
    - Fungují na stejném principu jako předchozí dvě. Proměnné jsou přístupné všem potomkům (mohou zjišťovat jejich stav i nastavovat hodnoty), ale potomci dědí pouze přístupová práva k této proměnné. To znamená, že hodnota proměnné je stejná pro všechny potomky na rozdíl od vlastní proměnné, kterou si každý potomek může nastavit individuálně.

6 Metodika výuky objektového programování

Jestliže jsme přijali fakt, že cílem výuky programování není vychovat budoucí programátory, měli bychom se zamyslet nad tím, jakým způsobem při výuce postupovat.

6.1 Obecné postupy výuky OOP

Vlastní vstup do světa objektového programování probíhá podle Pecinovského [10] ve třech krocích hlavních krocích:

6.1.1 Seznámení

V prvním kroku se děti seznámí s nějakým připraveným programem. Prohlédnou si jeho strukturu, seznámí se s třídami a objekty a ujasní si vzájemné vztahy jednotlivých tříd a jejich instancí.

6.1.2 Zkoumání programu

V dalším kroku si děti vytvářejí instance a zkouší s nimi pracovat. Volají jejich metody, zkoumají jejich chování a hodnoty atributů. Hlavním cílem je seznámit děti se vztahy mezi třídou a objektem.

6.1.3 Vlastní tvorba

Ve třetím kroku se začínají vytvářet vlastní programy. Nejprve děti upravují existující programy, na kterých se naučí vše potřebné a teprve poté začnou vytvářet programy vlastní, které mohou začlenit do rozsáhlejší aplikace a rozšířit tak její možnosti.

6.2 Metodika výuky OOP v prostředí Imagine

Blaho s Kalašem [6] rozdělují výuku programování v prostředí Imagine do 7 kroků, První dva kroky jsou doplněny o informace z jiné jejich práce [1]:

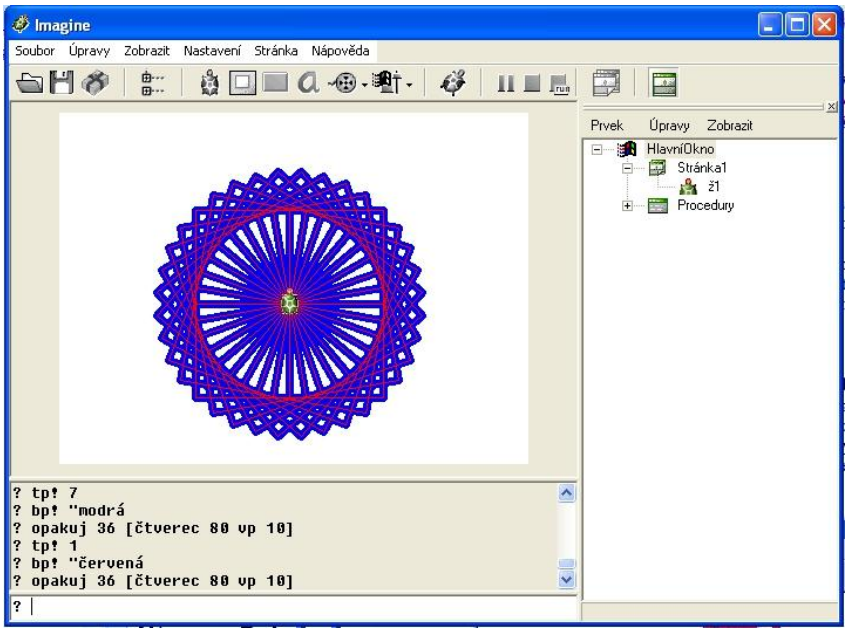
1. Práce s jednou želvou
2. Základní činnosti s více želvami
3. Želvy využívající privátní informace
4. Klonování objektů
5. Instance jako vzor pro vytváření instancí
6. Vícevrstvá hierarchie
7. Vytváření nové třídy

6.2.1 Práce s jednou želvou

V této části je výuka zaměřená převážně na tzv. želví grafiku. Tato část nevyžaduje žádné vědomosti o objektovém řešení prostředí Imagine . Děti využívají želvu ke kreslení různých tvarů a obrázků a učí se ovládat želvu pomocí příkazové řádky.

Děti zjišťují, že želva kromě toho, že vykonává jejich pokyny má také své vlastnosti, které mohou měnit (barva pera, tloušťka pera apod.).

Postupně se naučí pracovat s cykly (Obrázek 4)a vytvářet vlastní příkazy. Své příkazy postupně vylepšují a vytváří tak složitější obrázky, které se mohou spojovat a vytvářet další příkazy.



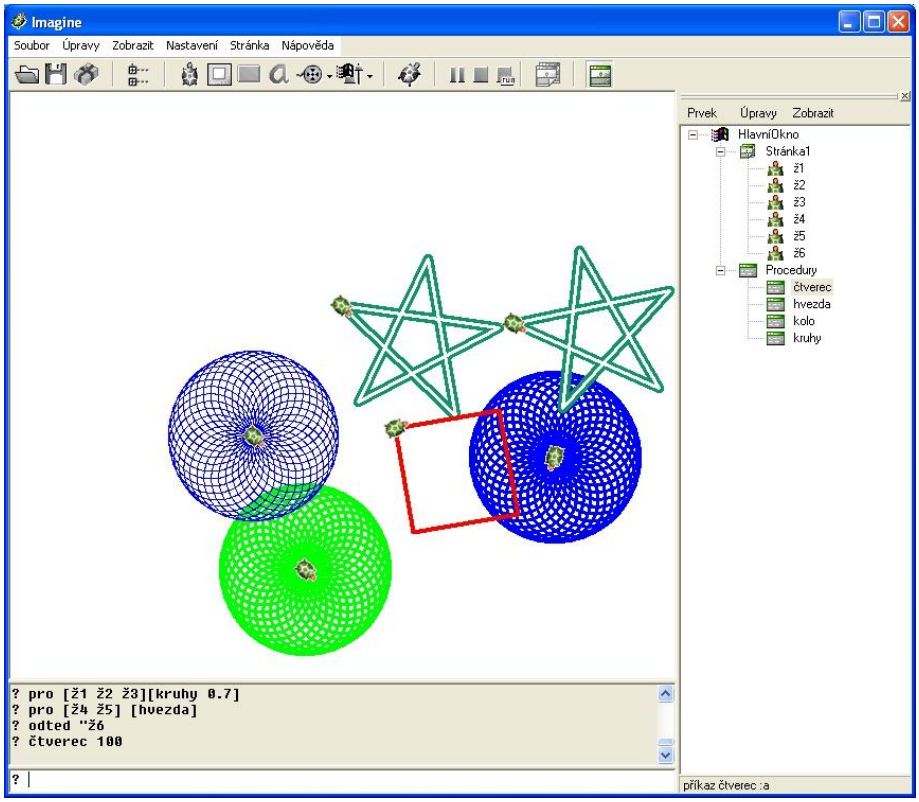
Obrázek 4 - Želví grafika a cyklus opakuj

Své příkazy dále vylepšují tím, že ve svých příkazech používají vstupní parametr, který ovlivňuje výsledné chování příkazu. Blaho a Kalaš tuto látku sice zařazují v [1] až za práci s více želvami, avšak v [6] jí využívají již prvním kroku výuky objektového programování.

6.2.2 Základní činnosti s více želvami

Tento krok je, co se týče práce s želvou, dosti podobný s krokem předchozím s tím rozdílem, že děti již pracují s více želvami. Nejprve se je učí vytvářet, přesouvat a odstraňovat podobně jako na Pecinovského [10] ukázkovém projektu, kde děti umisťují na plátno různé grafické projekty a manipulují s nimi.

Poté děti pracují s rodným listem želvy. Mění její jméno, tvar, viditelnost, domovskou pozici. Pokud využíváme více želv, tak se děti zároveň musí naučit jednotlivé želvy oslovovat (adresovat) viz. Obrázek 5.

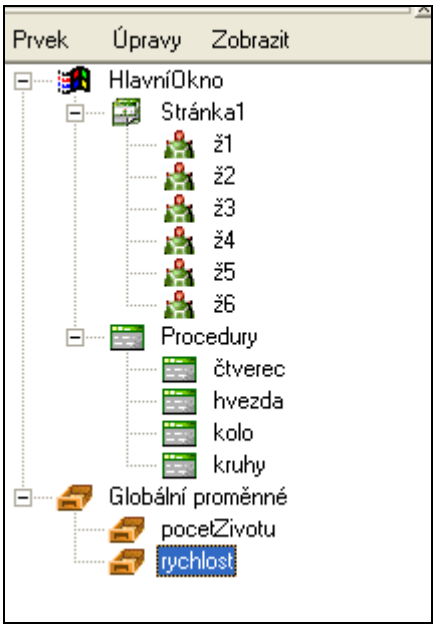


Obrázek 5 - Používání více želv

Pecinovský [10] k tomuto přistupuje v opačném pořadí, nejprve dětem vysvětlí tvorbu identifikátorů a teprve poté přistupuje k vysvětlování atributů instancí.

Když se děti naučí oslovovat jednotlivé želvy, či skupiny želv, mohou přistoupit k vytváření paralelních procesů. Děti vytvoří pro každou želvu nějaký proces, který želva vykonává neustále, nezávisle na ostatních želvách.

Dále se děti naučí vytvářet globální proměnné, které můžou později použít jako parametr při volání procedur (Obrázek 6).

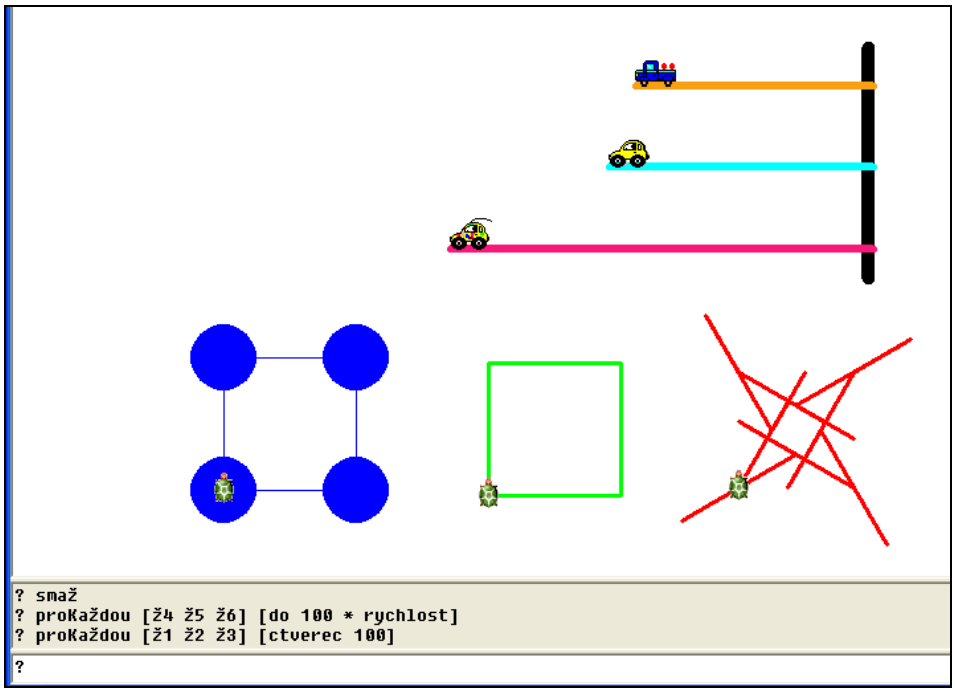


Obrázek 6 - Globální procedury a proměnné

6.2.3 Želvy využívající privátní informace

Někdy však není vhodné používat globální proměnné, proto se děti v dalším kroku seznámí s vytvářením proměnných privátních. Již teď děti vědí, že každá želva má nějaký rodný list, který si uchovává informace o tvaru, barvě pera, zvětšení, směru apod. Nyní se dozví, že každý objekt v prostředí Imagine si může uchovávat také několik privátních proměnných, což se dá velmi dobře využít při řešení různých problémů.

Kromě privátních proměnných můžeme každému objektu také vytvořit privátní procedury. Děti můžou například naučit želvy kreslit domy různého vzhledu, letadla lítat podle různých pravidel apod. Imagine nám zároveň umožňuje tzv. překrývání přednastavených metod. Děti si mohou například vytvořit u některých želv privátní proceduru dopředu a želvy se tak budou pohybovat podle jiných pravidel [6] (Obrázek 8).



Obrázek 7 - Používání privátních informací

6.2.4 Klonování objektů

Tato kapitola se zaměřuje na situace, kdy je třeba vytvořit více objektů se stejným chováním a stejnými (nebo podobnými) vlastnostmi – rybičky v akváriu, prvky stavebnice, karty pexesa apod.

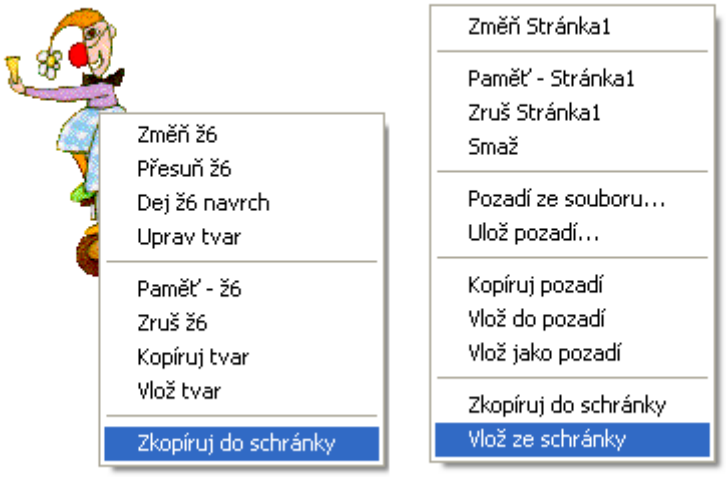
Při klonování vytváříme identickou kopii nějakého objektu. Když vytváříme klon, vytváříme vlastně nový objekt, který má stejné nastavení jako objekt původní (pouze s jiným jménem). Protože však vznikne nový samostatný objekt, už nám nic nebrání v tom, abychom klonům jejich nastavení změnili.

Proces klonování probíhá následujícím způsobem:

- nejprve si vytvoříme objekt, se kterým chceme pracovat,
- poté mu nastavíme všechny námi požadované vlastnosti, procedury a reakce na události,

- teprve pak přejdeme k samotnému klonování.

To můžeme udělat dvěma způsoby. Buď příkazem **klonuj** v příkazovém řádku, nebo kopírováním objektu do schránky a vložením ze schránky (Obrázek 9).



Obrázek 8 - Klonování objektů

Tomcsányiová [11] poukazuje na to, že pokud chceme vpravit objekty do programování dětí ve věku 10 – 14 let, tak je vhodné vysvětlit dětem, že spousta věcí, které nás obklopují, jsou v podstatě kopiemi jen s mírnými úpravami. Například sklenice. Všechny mají stejnou funkci, ale mohou vypadat odlišně.

Výhoda klonování oproti předchozímu vytváření jednotlivých objektů je nasnadě. Stačí nám nastavit jeden objekt a jednoduchým klonováním vytvoříme potřebné množství objektů se stejným chováním, které později můžeme upravit. Nevýhodou je, že pokud později zjistíme, že je nutné některou z vlastností nebo chování upravit, musíme to provést u každého objektu zvlášť.

6.2.5 Instance jako vzor pro vytváření instancí

Další možnost, kterou nám Imagine poskytuje je použití instance jako vzor pro vytváření instancí. Na první pohled se může zdát, že jde o stejné řešení jako předchozí, dokonce i postup je stejný. Vytvořit objekt > nastavit vše potřebné > podle něj vytvořit další objekty. Rozdíl je v tom, že nyní, pokud se rozhodneme přidat nějakou reakci na událost, upravit nějakou proceduru apod. nemusíme tak učinit u každého jednotlivého objektu, ale stačí nám upravit objekt původní. K vytvoření instance z instance se použije stejný příkaz jako při vytváření nové želvy, jen použijeme název instance, od které budeme odvozovat (**? Nová "želva [ ]** se změní na **? Nová "ž1 [ ]**). Takovýto postup předkládá například projekt Mořský svět, vytvářený v rámci žákovského listu nazvaného „želva jako rodič aneb želví rodokmen" [A].

6.2.6 Vícevrstvá hierarchie

Při použití klonování vytváříme pouze další instance jedné třídy (tj. např. želvy). Pokud však vytváříme objekty z nějaké instance pomocí příkazu **nová,** vytváříme tak další vrstvu, založenou na jiném objektu. Objekty tak dědí vlastností svého předka.

Můžeme si například vytvořit želvu a nazveme jí auto (**? Nová "želva [jméno auto tvar auto]**). Specifikujeme jí všechny vlastnosti a chování. Od této želvy odvodíme další auta (**? Nová "auto […]**). Tím vytváříme instance druhé vrstvy – auto – která je odvozená od želvy. Od tohoto auta si můžeme odvodit další auto, které můžeme nazvat např. nákladní upravit mu potřebné vlastnosti a pomocí něj vytvořit 3. vrstvu.

6.2.7 Vytváření nové třídy

Vytváření objektů od jiných objektů však může způsobit neočekávané chování. V programu se objevují dva typy objektů. Jedny jsou pouze instancemi (potomky) a druhé jsou vzorem (rodiči) a instancemi zároveň. To že rodiče nemohou být zrušeni, pokud nejsou zrušeni všichni jejich potomci, nebo že pokud změníme chování nějakého objektu, může mít dopad jak pouze na něj, tak na celou skupinu, pak může způsobit neočekávané chování při běhu programu.

Aby děti vytvářeli opravdu objektové řešení je potřeba, aby si definovat vlastní třídu. To způsobí, že definice třídy bude opravdu abstraktní a nebude se pohybovat společně s ostatními objekty na stránce.

Vše potřebné (události, proměnné, procedury atd.) se definuje pouze jednou a případná úprava pak má efekt na celou skupinu objektů. Všechny takto definované třídy však musí být potomky jednoho z primitivních objektů Imagine (např. Želvy, tlačítka, …) a podobně jako v případě vytvoření nové želvy i při vytvoření nové instance námi vytvořené třídy můžeme změnit její nastavení.

Pokud máme vytvořenou novou třídu, můžeme od ní odvozovat další třídu a vytvářet tak vícevrstvou hierarchii podobně jako v předchozím kroku.

7 Navrhovaný postup výuky

Metodika výuky OOP v Imagine navrhovaná Blahem a Kalašem [6] byla mírně upravena. Některé kroky bylo nutné doplnit o kompetence, které by se měl žák v daném kroku naučit, neboť práce Blaha a Kalaše předkládá hlavně praktické řešení konkrétních projektů pro danou kapitolu. Výsledkem je postup rozdělený do šesti následujících kroků:

1. Co všechno želva umí
2. Želví kamarádi
3. Učíme želvy novému chování
4. Klonujeme želvy, aneb želva jako učitel
5. Želva jako rodič, aneb želví rodokmen
6. Vytváříme vlastní třídu

Ke každé kapitole jsem navíc připravil projekt se žákovským listem, na kterém si žáci mohou ověřit, zda se v dané problematice orientují.

7.1 Změny v navrhovaném postupu

Kapitola „co všechno želva umí“ vychází z kapitoly „práce s jednou želvou“. Blaho a Kalaš [6] však k tomuto kroku přistupují pouze jako k prostředku k vytváření želví grafiky. Proto byl tento krok doplněn o používání „rodného listu“ želvy, který přiblíží dětem želvu jako objekt podle zásady předkládat studentům objekty od samého počátku výuky programování [10]. K této kapitole byl vytvořen projekt Kouzelník, ve kterém děti pomocí kouzelné hůlky „čarují“ obrázky. V projektu si vyzkouší práci s rodným listem želvy, ale také tvorbu procedur a vytváření želví grafiky.

Kapitola „želví kamarádi“ vychází z „práce s více želvami“. Obsah kapitoly neprodělal téměř žádné změny. Zaměřuje se především na komunikaci s jednotlivými želvami (tj. zasílání zpráv) a navíc upozorňuje na nutnost jedinečné identifikace každého objektu. Kapitola využívá projekt Vzdušný prostor, kde se děti učí vytvářet nové želvy, zasílat jim zprávy a vytvářet paralelní procesy.

Kapitola „učíme želvy novému chovaní“ vychází z kapitoly „želvy využívající privátní informace“ a obsah této kapitoly nedoznal žádných změn. V této kapitole děti vylepšují projekt Vzdušný prostor tak, že začínají používat privátní proměnné a procedury. V projektu se učí pracovat i s jinými objekt než je želva.

Příliš změn nedoznala ani kapitola zabývající se klonováním („klonujeme želvy aneb želva jako učitel“) vycházející z „klonování objektů“ Blaha a Kalaše [6]. V připraveném projektu děti vytváří jednoduchou stavebnici, která obsahuje dva druhy objektů, jedním jsou předlohy, které slouží pro vytváření stavebních útvarů a druhými jsou právě prvky stavebnice. Děti se zde zdokonalují v nastavování vlastností objektů při vytvoření.

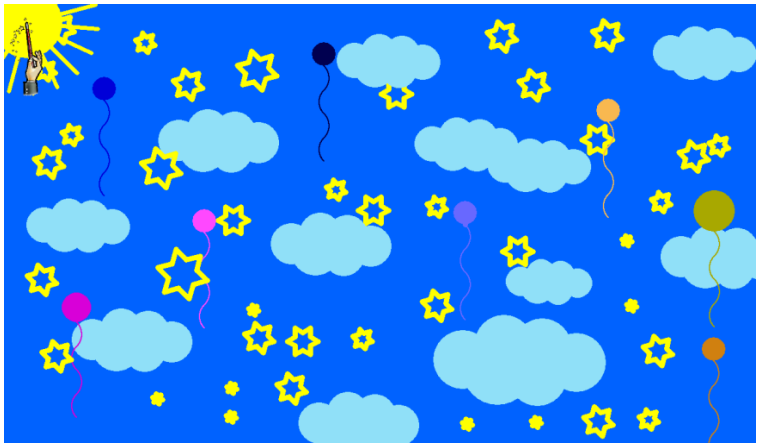
Další kapitola nazvaná „želva jako rodič aneb vytváříme želví rodokmen“ je výsledkem spojení kapitol „instance jako vzor pro vytváření instancí“ a „vícevrstvá hierarchie“. Ke spojení těchto dvou kapitol došlo z toho důvodu, že kapitola „vícevrstvá hierarchie“ je pouze prohloubením informací předchozí kapitoly a kromě toho, že si děti mohou vytvářet složitější hierarchii objektů nepřináší nic nového. Kapitolu jsem navíc doplnil o rozlišování mezi vlastními a společnými proměnnými, protože rozlišování těchto dvou typů proměnných má efekt právě až od této kapitoly. Kapitola využívá projekt Mořský svět, kde děti vytváří mořské živočichy a rozdělují je do skupin podle chování.

Poslední krok, zabývající se vytvářením nové třídy nedoznal žádných změn. K němu je využíván projekt Mořský svět, který je již od začátku řešen objektově pomocí nově vytvořených tříd. Ke zvolení stejného projektu jako při předchozím kroku jsem přistoupil proto, že se tak, jak doporučuje Tomscányiová [11], snáze pochopí rozdíl mezi oběma přístupy.

7.2 Obsah jednotlivých kroků výuky

7.2.1 Co všechno želva umí

V této kapitole se děti seznámí s prostředím Imagine. Ukážeme dětem, z kterých částí se prostředí skládá, kam se zapisují příkazy a jak se vytváří a ukládají projekty. Celý úvod doplníme o příběh o želvě z ostrova Logo, která leze v písku, ocáskem za sebou v písku nechává čáru a poslouchá naše příkazy. Proto se děti učí ovládat želvu a vytvářet tzv. želví grafiku, která je náplní také projektu *Kouzelník*.

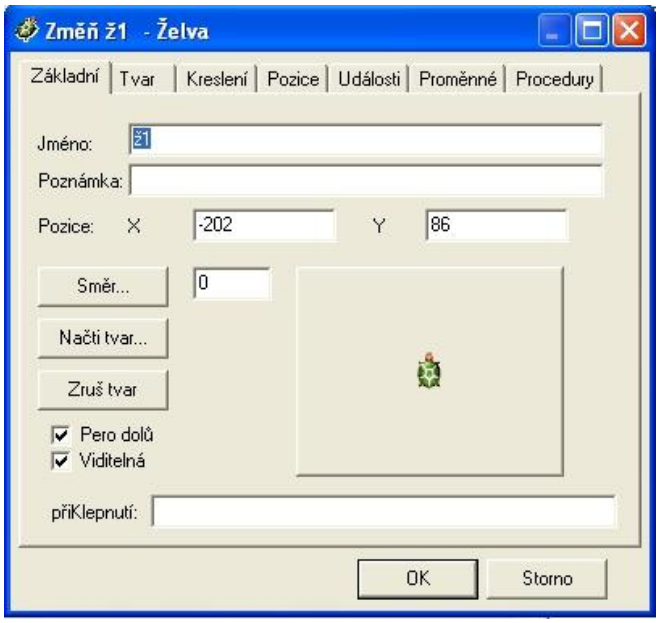


Obrázek 9 - Projekt Kouzelník

Když se děti naučí ovládat želvu pomocí příkazů, naučí se vytvářet své vlastní příkazy (zatím pouze globální procedury). Ty začnou později vylepšovat vstupními parametry.

V rodném listu také objeví záložku události, která umožňuje vytvářet reakce na nějakou událost v životě želvy.

Hlavním cílem tohoto kroku však není tzv. želví grafika, i když je zde hojně využívána. Pokud chceme dětem představit objektový přístup, měli bychom zdůrazňovat hlavně to, že želva je nějaký objekt a má své vlastnosti. K tomu je vhodné využit tzv. rodného listu želvy, který graficky znázorňuje základní nastavení želvy. Děti pak mění základní nastavení jak pomocí příkazového řádku, tak pomocí rodného listu.



Obrázek 10 - Rodný list želvy

Činnost	Příkazy	Nové informace
Základní ovládání želvy	do, vz, vpravo, vlevo, domů, smaž, puntík, opakuj	Želva poslouchá naše příkazy
Nastavování vlastností	**bp!**, **tp!**, **směr!**	Želva (objekt) má nějaký soubor vlastností.
Vytváření procedur	**uprav** (**up**) **definuj**	Můžeme vytvářet volat a spojovat vlastní procedury.
Procedury s parametrem		Výsledek procedury se dá ovlivnit vstupním parametrem.
Vytváření proměnných	**dosaď**	Můžeme vytvářet proměnné, ke kterým můžeme kdykoliv přistupovat.
Reakce na události	**přiTáhnutí** **přiKlepnutí** **přiNajetíMyši**	Želva může reagovat na nějakou naší činnost.

7.2.2 Želví kamarádi

V druhé kapitole se děti dozvídají, že ve světě Imagine může existovat více než jedna želva. Cílem je naučit děti pracovat s více objekty.

Děti postupně zjistí, že pokud chtějí vytvářet složitější programy, tak si občas s jednou želvou nevystačí, proto postupně želvě vytvoří další kamarády. Nejprve je vytváří pomocí tlačítka na hlavním panelu, později pomocí příkazu **nová** a nakonec se naučí nastavovat některé parametry již při vytvoření nové želvy.

Potom si pomocí rodného listu prohlíží jejich údaje a mění některá nastavení. Při této příležitosti zdůrazníme, že každému objektu náleží právě jeden rodný list, neboli soubor vlastností a každý objekt musí mít své jedinečné jméno.

Když mají vytvořené větší množství želv, ukážeme jim způsoby, jak dávat želvám příkazy. Nejprve dávají příkazy s přímým oslovením želv (**ž3'do 100**), později je naučíme používat příkazy **odteď** a **pro** a na příkladu jim ukážeme rozdíl mezi těmito dvěma příkazy. Oba tyto příkazy doplníme o proměnnou **kdo**, která uchovává seznam aktivních objektů.

Pokud jsme tak neudělali dříve, tak děti naučíme také fungování příkazu **každých** pro vytváření paralelních procesů, který využijeme v situacích, kdy chceme, aby želva neustále vykonávala nějakou činnost. Toho využívá například projekt *Vzdušný prostor*.

Obrázek 11 - projekt Vzdušný prostor

Zde samozřejmě můžeme využít i jiných objektů než jsou želvy, můžeme je naučit pracovat se stránkou (změnit pozadí, přidat reakce na události apod.) či tlačítky (např. pro smazání stránky).

K tomu samozřejmě opět využijeme rodný list a také představíme dětem okno Paměť, kde vidí všechny vytvořené objekty.

Činnost	Příkazy	Nové informace
Vytváření dalších želv	**nová**	I když objekty vycházejí z jedné třídy, mohou mít rozdílné vlastnosti
Zasílání zpráv objektům, aktivní objekty	**kdo, odteď, pro, proKaždou**	Pokud chceme nějakému objektu zaslat zprávu, musíme Imagine informovat, komu má zprávu předat.
Vytváření paralelních procesů	**každých**	Želva může vykonávat nějakou činnost neustále a nezávisle na dalších činnostech.

7.2.3 Učíme želvy novému chování

Hlavním cílem dalšího kroku je naučit děti, že každý objekt v Imagine si může uchovávat nějaké soukromé (privátní) informace. S tím, že každý objekt má své základní nastavení, se seznámili v předchozích krocích. Zatím by měly vědět, že jde o nějaké proměnné, jejichž nastavení si každý objekt sám uchovává (má je ve svém rodném listu).

Nyní dětem ukážeme možnost přidávat objektům další proměnné. Já využívám projekt z předchozí lekce (*Vzdušný prostor*), kde se pohybují objekty různou rychlostí po obloze. V tomto projektu je nutné každému objektu říkat, jakou rychlostí má létat. My děti naučíme, že pokud nějaká vlastnost charakterizuje objekt (podobně jako tvar, směr, domovská pozice, nebo právě rychlost pohybu), je vhodné, aby si jej pamatoval sám. Zatím používáme pouze vlastní

proměnné, protože rozdíl mezi vlastními a společnými proměnnými je patrný až při od bodu pátého kroku.

Do této fáze děti věděly, že každá želva si může uchovávat pouze základní nastavení a události, ale příkazy a procedury byly vždy přístupné všem. Nyní se děti naučí vytvářet privátní procedury. Děti si u každé želvy vytvoří stejný příkaz s rozdílným seznamem instrukcí a poté nechají vykonat příkaz všechny takto upravené objekty, čímž se naučí využívat principů polymorfismu.

Když používáme privátní procedury a proměnné, musíme dětem vysvětlit, jak k nim želva přistupuje. Vysvětlíme jim, že když dáme želvě nějaký příkaz, želva se nejprve podívá, jestli umí vykonat příkaz sama a teprve poté začne hledat jinde. Tak jim představíme možnost překrývat základní procedury a používání příkazu obvykle.

Činnost	Příkazy	Nové informace
Vytváření objektových proměnných	**ať JeVlastní**	Každý objekt může mít kromě základního nastavení také vlastní proměnné definované uživatelem.
Vytváření objektových procedur	**uprav** **obvykle**	Dva objekty mohou na jednu zprávu reagovat různě.

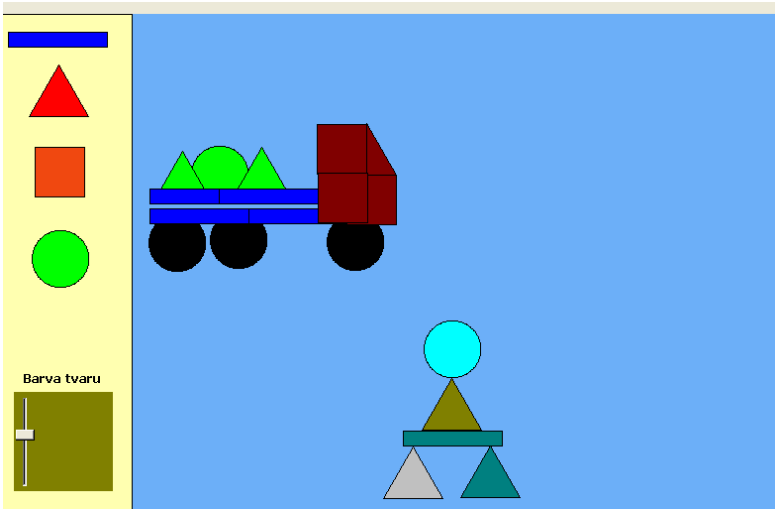
7.2.4 Klonujeme želvy aneb želva jako učitel

V předchozích krocích se děti postupně naučily jak upravovat objekty tak, aby se chovaly podle jejich představ. Stále však byly nucené učit každý objekt novým dovednostem zvlášť. Další krok, jim značně usnadní práci v situacích, kdy potřebují více objektů se stejným nebo podobným chováním. V tomto kroku proniknou do tajů klonování.

Děti se naučí, že předtím, než začnou vytvářet nějaké větší množství objektů, je vhodné si nejprve promyslet, co budou od těchto objektů očekávat. Pokud si totiž vytvoří nejprve objekty a teprve poté se je rozhodnou upravovat, budou mít před sebou zdlouhavou práci. Pokud si však vše předem rozmyslí, budou si moci klonováním značně usnadnit práci.

Naučíme je, že si nejprve musí vytvořit nějaký prototyp, kterému nastaví všechny potřebné vlastnosti. Upraví základní nastavení (tj. stav pera, tvar apod.), poté vytvoří všechny potřebné vlastní proměnné a pak vytvoří všechny objektové procedury a nastaví reakce na události.

Když mají hotový prototyp, můžou přistoupit ke klonování. Nejprve jim ukážeme klonování pomocí kopírování objektu do schránky a poté pomocí příkazu **klonuj**, který umožňuje upravit některé vlastnosti již při vytváření klonu, který využívá projekt *Stavebnice*.



Obrázek 12 - Projekt Stavebnice

Společné proměnné fungují při klonování stejně jako vlastní proměnné tj. nedědí se přístup, ale s novým objektem se vytvoří nová společná proměnná, kterou můžeme nastavit nezávisle na původní proměnné, takže je zbytečné jí zde využívat.

Činnost	Příkazy	Nové informace
klonování	**klonuj**	Pokud vytváříme větší množství objektů s podobným chováním, není vhodné nastavovat každý zvlášť.

7.2.5 Želva jako rodič aneb želví rodokmen

V tomto kroku dětem představíme možnost vytvářet instance z instance. Hlavní myšlenka navazuje na předchozí krok a to vytvořit nejprve prototyp se všemi vlastnostmi a pomocí něj vytvářet další objekty.

Při tvorbě klonů se dodatečná změna chování musí provádět u všech takto vytvořených objektů. Proto dětem nabídneme jinou možnost a to právě vytváření instance z instance, kdy se k nějaké instanci vytvářejí takzvaní potomci. To jim umožní měnit chování objektů za chodu programu, a všechny objekty se od objektu, od něhož jsou odvozeny, vše okamžitě naučí.

V tomto kroku zároveň můžeme odlišovat vlastní a společné proměnné. Zatímco barvaPera, stavPera apod. jsou vlastní proměnné, tudíž je můžeme měnit, aniž bychom ovlivnili nastavení celé skupiny objektů, změna společné proměnné ovlivňuje celou skupinu.

Tím, že si děti vytvoří instance z instance, to však nekončí. Když už mají vytvořené nějaké odvozené objekty a rozhodnou se, že část z nich chtějí něčím obohatit, nemusejí si vytvářet novou želvu,

měnit jí tvar a nastavovat vše znovu. Stačí si vytvořit potomka původního objektu, který už zdědil procedury a všechny další nastavení a následně potomka upravit.

Děti se tak naučí vytvářet si určitou hierarchii objektů a seskupovat je do určitých logických celků se společnými vlastnostmi a chováním.

Pro lepší znázornění hierarchie můžeme dětem dát za úkol sestavit rodokmen jejich programu. Při tom je dobré zdůraznit, že není vhodné vytvářet novou vrstvu, pokud nehodlají nějakým způsobem bohatit chování objektů.

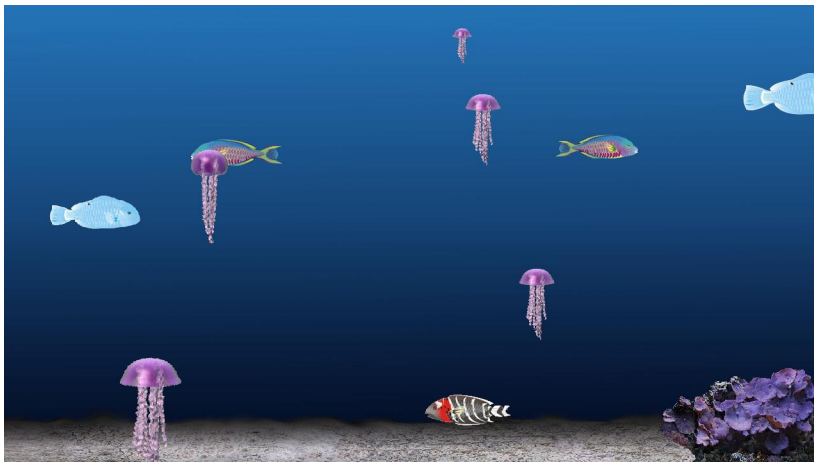
Činnost	Příkazy	Nové informace
Vytváření instance z instance	**nový** / **nová** / **nové**	Pokud od nějakého objektu odvozujeme (nekopírujeme jej), pak můžeme změnou jeho nastavení ovlivnit objekty od něj odvozené.
Vytváření hierarchie		Pokud vytváříme více objektů se společným nastavením, je vhodné je spojovat do skupin.

7.2.6 Vytváříme vlastní třídu

Předchozí řešení má však jeden malý háček. Jak poznat, který z objektů je vzorem? Pokud totiž budeme chtít smazat vzorový prototyp, od kterého jsou ostatní objekty odvozeny, Imagine nám to nedovolí, protože objekt má potomky, kteří jsou na něm závislí. Jsou však situace, kdy je však nutné se všemi objekty pracovat stejně, jako je tomu například v projektu *Mořský_svět-třídy*, kde se rybky, které přijdou o své životy, odstraňují.

Proto si s dětmi začneme vytvářet vlastní třídu. Děti si prohloubí své abstraktní myšlení. Nejprve vytvoří základní nastavení

třídy, aniž by viděly nějaký výsledek, definují všechny procedury, potřebné proměnné, reakce na události a nastaví základní vlastnosti tak, jak chtějí, aby byly nastaveny při vytvoření nového objektu. Teprve pak začnou vytvářet vlastní objekty.



Obrázek 13 - Projekt Mořský svět - třídy

Podobně jako v předchozím kroku si mohou vytvářet hierarchii tříd. Díky tomu pak mohou měnit chování objektu kdykoliv potřebují a změna bude mít dopad na všechny vytvořené instance.

Činnost	Příkazy	Nové informace
Vytváření tříd	**nováTřída**	Pokud budeme vytvářet více objektů podobného chování a předpokládáme, že je při chodu programu budeme rušit, je nutné definovat vlastní třídu.

8 Závěr

Přestože je prostředí Imagine pro objektový přístup připraveno, i sami autoři na stránkách Imagine vyzdvihují možnost využívat Imagine zcela neobjektově. Tím možná nevědomky nabádají k tomu, aby se uživatelé při tvorbě projektů a výuce programování v Imagine objektovému přístupu vyhýbali.

Přesto, anebo právě proto, je jediný ucelenější materiál zabývající se objektovým přístupem dílem právě autorů tohoto prostředí [6]. Problémem je zřejmě to, že prostředí Imagine nahlíží na objekty a OOP poněkud odlišněji, než je tomu u jiných objektově orientovaných jazyků.

Z tohoto důvodu bylo vytváření metodických materiálů velmi složité a některé kroky postupu se opíraly z velké části o osobní zkušenosti.

Při tvorbě praktických projektů jsem někdy narážel na problém, kterým byla nepřesná nápověda, ale metodou pokus omyl se problém podařilo vždy zdárně vyřešit. Výraznějším problémem, na který jsem narazil bylo to, že při pohybu většího množství objektů na stránce, docházelo ke značnému zpomalení činnosti programu. Pokud jsem se v ten moment pokusil zobrazit rodný list, musel jsem minimalizovat okno Imagine a teprve při maximalizaci se rodný list zobrazil.

Tvorba objektů v Imagine sice neprobíhá stejně jako u běžných programovacích jazyků, ale základní princip je zde stejný, objekty komunikující pomocí zasílání zpráv, a proto přestože je přístup Imagine odlišný, může značně pomoci i budoucím profesionálním programátorům.

Osobně si myslím, že výhodou výuky OOP v Imagine je právě to, jakým způsobem vytváří a interpretuje třídy a objekty, protože tím

překlene často nezáživné začátky objektového programování v běžných programovacích jazycích. Imagine odstraňuje i značnou složitost při vytváření i jednoduchých objektových programů a tak mohou objektově programovat i děti, bez nadání pro programování.

Reference

[1] BLAHO, Andrej; KALAŠ, Ivan. *Imagine Logo : Učebnice programování pro děti*. Brno : Computer Press, 2006. 48 s. ISBN 80-251-1015-X.

[2] VANÍČEK, Jiří; MIKEŠ, Radovan. *Informatika pro základní školy a víceletá gymnázia 3*. Brno : Computer press, 2006. 96 s. ISBN 80-251-1082-6.

[3] KEOGH, Jim. *OOP bez předchozích znalostí : průvodce pro samouky*. Brno : Computer press, 2006. 222 s. ISBN 80-251-0973-9.

[4] ČADA, Ondřej. *Objektové programování : naučte se pravidla objektového myšlení* . Praha : Grada, 2009. 200 s. ISBN 978-80-247-2745-5.

[5] PECINOVSKÝ, Rudolf. *Myslíme objektově v jazyku Java.* Praha : Grada, 2008. 576 s. ISBN 978-80-247-2653-3.

[6] BLAHO, Andrej; KALAŠ, Ivan . Object Metaphore Helps Create Simple Logo Projects. In *Eurologo 2001* [online]. Bratislava : Department of Informatics Education, Comenius University, 2001 [cit. 2011-02-15]. Dostupné z WWW: <http://www.ocg.at/activities/books/volumes/band%20156/K12blaho%20kalas1.doc>.

[7] Rámcový vzdělávací program pro základní vzdělávání. Praha : VÚP, 2007. 126 s. Dostupné z WWW: <http://www.vuppraha.cz/wp-content/uploads/2009/12/RVPZV_2007-071.pdf>.

[8] VANÍČEK, Jiří. *Imagine Logo aneb moderní výuka programování do českých škol*. Portál Česká škola [online]. 2006,

[cit. 2011-04-20]. Dostupný z WWW: <http://www.pf.jcu.cz/imagine/doc/Cpress_clanek_Imagine.pdf>. ISSN 1213-6018.

[9] PECINOVSKÝ, Rudolf. Současné trendy v metodice výuky programování. In *Počítač ve škole 2006* [online]. Nové město na Moravě : Gymnázium Vincence Makovského, 2006 [cit. 2011-04-21]. Dostupné z WWW: <http://www.gynome.nmnm.cz/konference/files/2006/sbornik/pecinovsky.pdf>.

[10] PECINOVSKÝ, Rudolf. Proč a jak učit OOP žáky základních a středních škol. In *Žilinská didaktická konferencia 2004* [online]. Žilina : Žilinská univerzita v Žiline, 2004 [cit. 2011-04-22]. Dostupné z WWW: <http://vyuka.pecinovsky.cz/prispevky/Proc_a_jak_ucit_OOP_na_ZS_a_SS.pdf>.

[11] TOMCSÁNYIOVÁ, Monika. Cloning in Logo programming. In *EuroLogo 2001* [online]. Wien : Österreichische Computer Gesellschaft, 2001 [cit. 2011-04-22]. Dostupné z WWW: <http://edi.fmph.uniba.sk/~tomcsanyiova/Clanky/Eurologo/2001/Klonovanie.htm>. ISBN 3-85403-156-4.

Příloha A - Žákovské listy

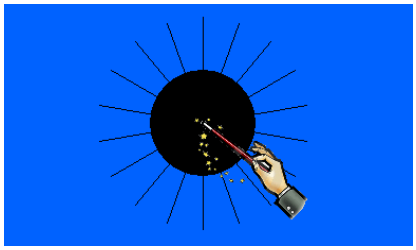
Co všechno želva umí

Tato kapitola má za úkol procvičit základní ovládání želvy, vytváření nových procedur a práci s rodným listem. Budeme si hrát na kouzelníka, který se rozhodl dát na dráhu malíře, a pomocí kouzelné hůlky budeme čarovat obrázky na obloze.

Úkol: Otevři si soubor *kouzelník.IMP*, změň tvar želvy na kouzelnou hůlku, nastav jí automatické táhnutí a změň jí jméno na **kouzelná_hůlka.**

Pamatuj! Rodný list je soubor vlastností želvy a umožňuje ti měnit její základní nastavení. Pokud jej chceš vyvolat z příkazového řádku, použij příkaz **jménoŽelvy'změňMě**

Úkol: Vytvoř proceduru **slunce**, která nakreslí slunce tak, jak je vidět na následujícím obrázku.



Pamatuj! Příkaz **opakuj** se využívá v situacích, kdy má želva vykonat stejné instrukce několikrát za sebou. Pokud chceš naučit želvu novou proceduru, použij příkaz **uprav.**

Úkol: Uprav proceduru **slunce** tak, aby se slunce vždy kreslilo žlutou barvou a paprsky byly alespoň 5 bodů široké.

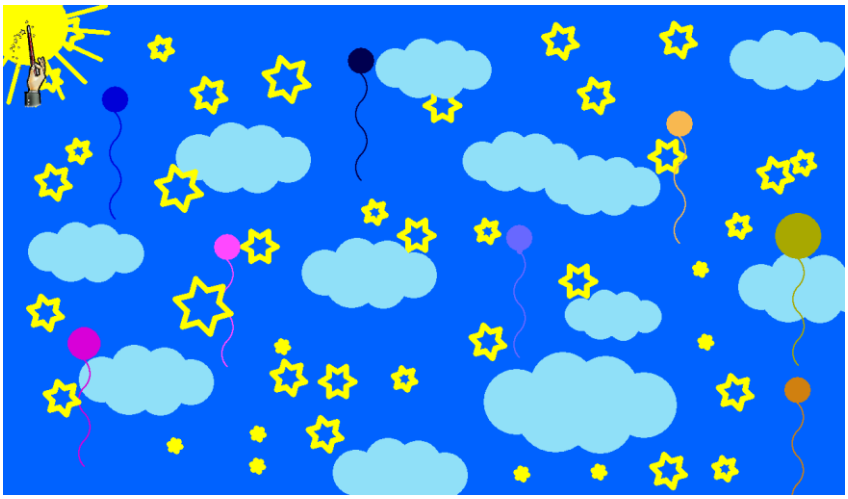
Úkol: Vytvoř proceduru **hvězda** a alespoň 2 další procedury na kreslení obrázků. Uprav hůlku tak, aby při poklepání nakreslila hvězdu.

Pamatuj! Pokud chceš, aby se při přetahování hůlky nekreslila čára, musíš vždy po dokreslení obrázku zvednout pero nahoru.

Úkol: Změň všechny procedury tak, aby se pomocí vstupního parametru dala určit velikost obrázku.

Pamatuj! Vstupní parametr se používá k ovlivnění výsledku dané procedury.

Úkol: Pomocí vytvořených procedur vyčaruj obrázky různých tvarů a různých velikostí.



Želví kamarádi

V této kapitole si procvičíme, jak se pracuje s více želvami najednou. V minulém projektu jsme se učili kreslit slunce, hvězdy a podobné obrázky. Dnes ještě zůstaneme v oblacích a vytvoříme si živý obrázek oblohy, na které se budou pohybovat letadla mraky a další objekty.

Úkol: Otevři si soubor *Vzdušný_prostor.IMP*, změň tvar želvy na mrak a nastav jí vše potřebné. Potom vytvoř proceduru **let** s parametrem **:rychlost**, která bude každých 30 milisekund posouvat želvu o parametr **:rychlost** dopředu.

Pamatuj! Příkaz **každých** vytvoří nový proces, který se bude vykonávat neustále, dokud ho nezastavíme.

Úkol: Přidej další objekty na svou oblohu a vytvoř proceduru **start**, při jejímž zavolání se každý objekt začne pohybovat jinou rychlostí vpřed.

Pamatuj! Každá želva má své jedinečné jméno, které slouží k její identifikaci.

Úkol: Uprav pohyb některých objektů tak, aby lítaly jiným způsobem (např. balon se točil v kruhu, vrtulník klesal a stoupal…)

Úkol: Nastav stránku tak, aby se při poklepání vytvořil nový objekt s automatickým táhnutím a perem nahoru.

Pamatuj! Želva není jediným objektem v Imagine. Želvy se pohybují po **stránce**, dále můžeme využít **tlačítko**, nebo **posuvník** atd. Všechny tyto objekty mají svůj rodný list.

Úkol: Nastav každému jednotlivým typům objektů (letadlům, mrakům, vrtulníkům) pohybujících se po tvé obloze jinou reakci na událost.

Učíme želvy novému chování

V předchozím projektu jsme na obloze vytvářeli létající objekty. Chtěli jsme, aby se některé pohybovaly jinak než jiné a měly různou rychlost. Dnes si náš projekt trochu vylepšíme a naučíme objekty, aby si samy pamatovaly jak rychle a jakým způsobem létat.

Úkol: Otevři soubor *Lepší_vzdušný_prostor.IMP*, který jsi vytvořil v minulé lekci a umísti na stránku přepínač Start! / Stop!, který při zmáčknutí spustí proceduru **start** a v druhé poloze zastaví všechny procesy. Podle toho také bude měnit svůj popis.

Úkol: Vytvoř každé želvě proměnnou **rychlost** a uprav procedury **let** a **start** tak, aby tuto proměnnou využívaly.

Pamatuj! Každému objektu v Imagine můžeme vytvářet nové proměnné, které budou uloženy v rodném listě daného objektu a budeme k nim moci přistupovat stejně jako k základnímu nastavení každého objektu.

Úkol: Nastav stránku tak, že se při poklepání vytvoří na onom místě nová želva a nastaví se jí při vytvoření proměnná rychlost v rozmezí 0,5 – 2,5.

Pamatuj! Privátní proměnné se mohou vytvářet společně s vytvořením objektů.

Úkol: Některým želvám vytvoř privátní proceduru **let**, podle které budou létat jiným způsobem než ostatní.

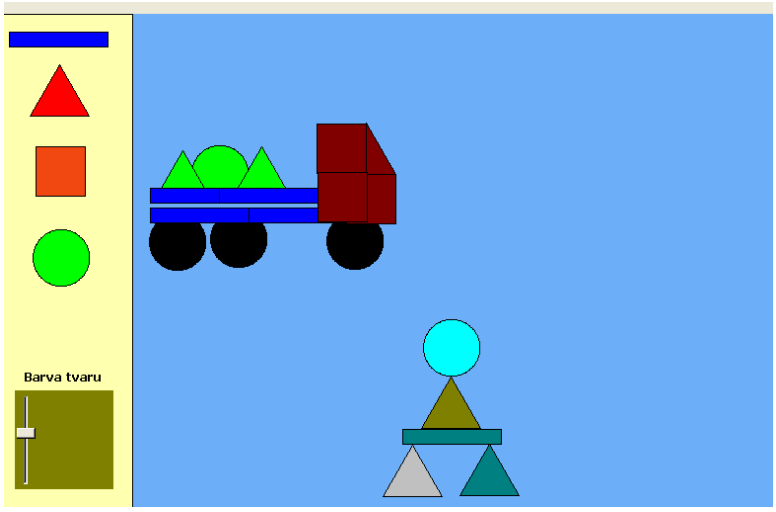
Pamatuj! Když se želva snaží pochopit nějaký příkaz, nejprve zjistí, jestli mu sama rozumí (má jej mezi privátními procedurami) a teprve pokud jej nenajde, hledá někde jinde.

Úkol: Helikoptérám vytvoř reakci na poklepání, při kterém helikoptéra klesne o nějakou výšku. Pro tyto účely jim vytvoř proceduru **klesej**.

Úkol: Vytvoř stránce proměnnou, která bude uchovávat obrázek ze souboru *obloha.jpg* a nastav stránku tak, aby se při smazání načetla tato proměnná do pozadí stránky.

Klonujeme želvy aneb želva jako učitel

Vždy, když jsme vytvářeli nový objekt v nějakém projektu, museli jsme mu změnit základní vlastnosti, jako byl stav pera, automatické táhnutí, reakce na událost, rychlost nebo nějaké procedury. Tentokrát si vytvoříme projekt stavebnice, na kterém si vyzkoušíme základy klonování, které nám vytváření nových objektů značně usnadní.



Úkol: Otevři si soubor *Stavebnice.IMP* a nastav posuvník **BarvaTvaru** tak, aby se při změně měnila jeho barva.

Úkol: Vytvoř želvě jménem **obdélník** novou proceduru nazvanou **vytvořKlon**, která vytvoří uprostřed stránky kopii této želvy. Zároveň vytvoř reakci na poklepání, která bude tuto proceduru volat.

Pamatuj! Když použijeme konkrétní název (“ž1, “obdélník apod.) všechny kopie budou odkazovat na stejný objekt. Pokud použijeme proměnnou **mojeJméno**, každý objekt bude odkazovat sám na sebe.

Úkol: Uprav vytvořenou proceduru tak, aby byla vytvořená kopie zamčená a měla nastavené automatické táhnutí.

Úkol: Uprav proceduru tak, aby se barva vytvořeného objektu nastavila podle hodnoty posuvníku.

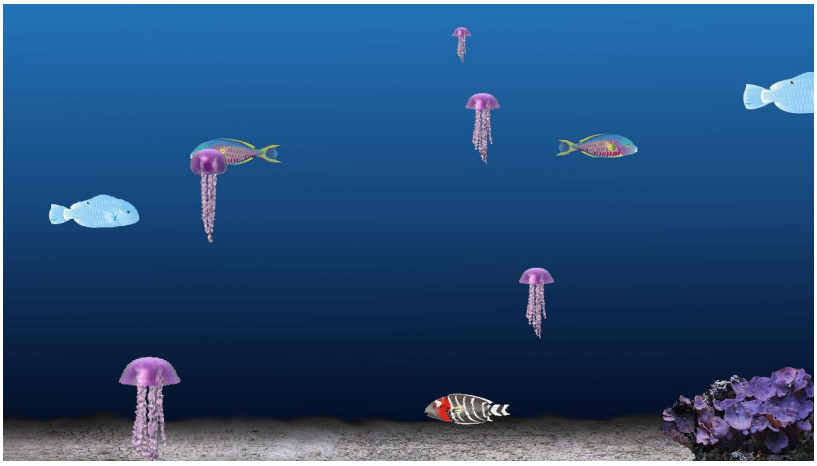
Pamatuj! Pokud je nutné použít nějakou operaci (výpočet, zjištění hodnoty nějaké proměnné apod.) tam, kde je očekávána nějaká hodnota (např. nastavování hodnot při vytváření objektu), Imagine nám bude hlásit chybu. Většinou stačí operaci pouze uzavřít do kulatých závorek.

Úkol: Uprav proceduru tak, aby se při nějaké události vytvořený objekt smazal a při jiné události se posunul navrch.

Úkol: Doplň stavebnici o další tvary.

Želva jako rodič aneb želví rodokmen

Dnes si vytvoříme malý mořský svět, na kterém si vyzkoušíme vytváření nových objektů pomocí nějakého dříve nastaveného. Zjistíme, že je vhodné seskupovat objekty do skupin podle chování a vytvoříme si objektovou hierarchii.



Otevři soubor *Mořský_svět.IMP*. Když si prohlídneš okno Paměť, zjistíš, že stránka při smazání obnovuje obrázek na pozadí a želva nazvaná **plavec** se při klepnutí otočí o 180 stupňů.

Úkol: Vytvoř želvě privátní proměnnou a proceduru tak, že se želva po zavolání procedury začne pohybovat vpřed rychlostí určenou privátní proměnnou.

Úkol: Nastav želvě událost **přiPoklepání**, která na libovolné pozici vytvoří novou želvu, která bude potomkem želvy **plavec**, a několik takových potomků vytvoř.

Pamatuj! Při vytváření objektu podle nějakého jiného objektu (instance z instance) můžeš nastavit vlastnosti už na začátku podobně jako při klonování.

Úkol: Vytvoř globální proceduru, po jejímž spuštění začnou všichni potomci **plavce** plavat.

Pamatuj! Při vytváření objektu podle nějakého jiného objektu (instance z instance) můžeš nastavit vlastnosti už na začátku podobně jako při klonování.

Úkol: Uprav **plavci** proceduru plav tak, aby se spuštěný proces jmenoval podle následujícího vzoru: **mojeJméno-plav**.

```
? piš všechnyprocesy
@Řádek plavec-plav ž1-plav ž2-plav ž3-plav duhovka-plav meduza-plav ž4-plav ž5-plav ž6-plav ž7-plav
```

Pamatuj! Pokud spouštíme více procesů se stejným tělem, je vhodné jednotlivé procesy pojmenovávat. Jméno nám umožní ukončit konkrétní proces.

Úkol: Uprav jednu nově vytvořenou želvu. Změň její vzhled a nastav, aby se při poklepání vytvářel její potomek. Potom jí uprav reakci na poklepání tak, aby při poklepání plavala jednou v kružnici a poté pokračovala v plavbě rovně.

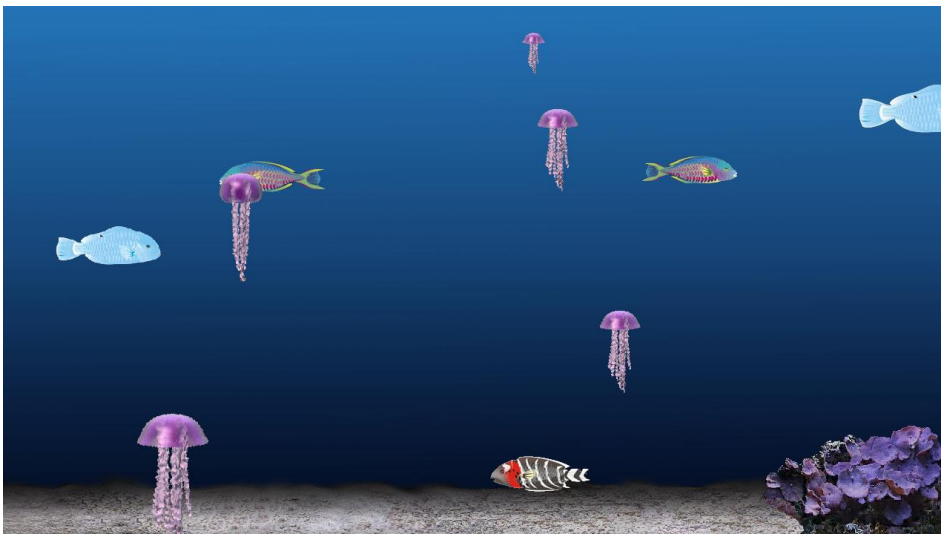
Úkol: Uprav jedné z želv vytvořených z původní rybky tvar na **jedovatáMedůza** a změň způsob jejího pohybu tak, aby se při plavání náhodně natáčela.

Pamatuj! Pokud vytváříme nějakému objektu proceduru, která se jmenuje stejně jako nějaká implicitní (součástí základního nastavení jako třeba procedura **dopředu**) a chceme použít tu implicitní, musíme použít slovo **obvykle** (**obvykle'dopředu 10**).

Úkol: Nastav medúzu tak, že při klepnutí se medúze pár vteřin změní fáze na 3 a poté se vrátí na 1.

Vytváříme vlastní třídu

V minulém projektu, kde jsme si vytvářeli mořský svět, jsme objekty seskupovali do hierarchického uspořádání. Tento projekt měl však nevýhodu, že se mezi sebou mísily děti s rodiči, což v některých situacích může dělat neplechu. Dnes si tento projekt vytvoříme od začátku, ale tentokrát pro vytváření potomků nepoužijeme instance, ale vytvoříme si vlastní třídy. Jako výchozí využijeme projekt *Mořský_svět-třídy.IMP*.



Úkol: Vytvoř novou třídu, která bude potomkem třídy želva. Nazvi jí třeba **plavec**, nastav jí nějakou předponu a tvar nastav na **rybka1.** Přidej proměnnou **stavPera** a hodnotu nastav na **pn.**

Pamatuj! Každá vytvořená třída má stejné základní nastavení jako její předek. Toto nastavení můžeme změnit tak, že vytvoříme nějakou proměnnou, proceduru, nebo událost, se stejným názvem jako má její předek a nastavíme jí podle našeho přání.

Úkol: Nastav vytvořenou třídu podobně jako v předchozím projektu (vytvoř jí proměnnou rychlost, proceduru plav – proces opět pojmenuj jménoObjektu-plav, otočení o 180° při klepnutí).

Pamatuj! Pokud vytvoříš několik objektů a později změníš proceduru nebo událost, změna bude mít vliv na všechny objekty této třídy podobně, jako je tomu při vytváření instance z instance.

Úkol: Vytvoř globální proceduru **start**, po jejímž spuštění začnou všichni potomci **plavce** plavat.

Úkol: Vytvoř další dvě třídy, **medúza** a **duhovka**, založené na třídě plavec. Ve třídě medúza vytvoř proceduru **dopředu** (nebo **do** podle toho, kterou používáš) a uprav jí tak, aby se před posunem dopředu náhodně natočila o pár stupňů doleva nebo doprava. Duhovkám změň tvar na rybka2 a nastav, aby se při klepnutí chovaly jinak, než plavci.

Úkol: Ve třídách plavec a duhovka vytvoř vlastní proměnnou životy a nastav jí dle uvážení. Potom vytvoř vlastní proměnnou jed ve třídě medúza a opět jí nastav nějakou hodnotu.

Teď si upravíme projekt tak, aby se po klepnutí stala medúza na chvíli jedovatou a když se v tu dobu setká s rybou, tak jí sebere několik životů podle toho, kolik má medúza jedu.

Úkol: Ve třídě medúza vytvoř událost přiKlepnutí, kdy na chvíli medúza přejde do „žahavého módu“ a událost přiSrážce, která sebere rybě životy za předpokladu, že bude medúza v žahavém módu.

Pamatuj! Starostí nějakého objektu by nemělo být to, co se stane s jiným objektem (rybkou) při nějaké události (přiSrážce s medúzou). Pro něj je důležité jen to, jakou informaci má předat (množství jedu) a co se stane dál, nechá na druhém objektu.

Úkol: Vytvoř ve třídě plavci proceduru, která zastaví pohyb rybky, sebere jí nějaké množství životů, ověří, jestli nějaké životy ještě rybce zbývají a pokud ano, opět spustí proceduru **plav**. Pokud rybce žádné životy nezbudou, rybku zničí.

Příloha B – CD s materiály

Obsahem tohoto CD jsou projekty připravené pro plnění úkolů ze žákovských listů [A], dále výsledné projekty vypracované na základě úkolů ze žákovských v adresáři Hotové_projekty a obrázky pro vytváření projektů ve složce Obrázky.

Popis souborů:

Název souboru	Popis činnosti	Žákovský list
Kouzelník.IMP	Vytváření želví grafiky	Co všechno želva umí
Vzdušný_prostor.IMP	Úvod do práce s více želvami	Želví kamarádi
Lepší_vzdušný_prostor.IMP	Vytváření privátních proměnných a procedur	Učíme želvy novému chování
Stavebnice.IMP	Vytváření objektů pomocí klonování	Klonujeme želvy aneb želva jako učitel
Mořský_svět.IMP	Vytváření instance z instance a vytváření hierarchie objektů.	Želva jako rodič aneb vytváříme želví rodokmen
Mořský_svět-třídy.IMP	Definování nových tříd v Imagine	Vytváříme vlastní třídu