INVESTICE DO ROZVOJE VZDĚLÁVÁNÍ

Výučba programovania hrou

Miroslav Biňas
Fakulta elektrotechniky a informatiky TUKE
miroslav.binas@tuke.sk

Úvod

Komenského známe „*škola hrou*“ je v našich česko-slovenských končinách stále veľmi známe. Otázkou však zostáva, do akej miery je známa aplikácia tohto slovného spojenia v praxi. Do akej miery sa naši študenti učia tým, že sa hrajú? Je vlastne možné túto metodiku použiť aj dnes?

Oblasť IT má ku hrám obzvlášť blízko, keďže významným odvetvím tejto oblasti je aj herný priemysel. Súčasné tzv. *áčkové tituly* pripravujú veľké herné štúdia a zoznam ľudí, ktorí sa akoukoľvek formou podieľali na ich vývoji častokrát pripomína záverečné titulky ktoréhokoľvek filmu. A práve počítačové hry prispeli v mnohých prípadoch k tomu, aby sa človek venoval niektorej oblasti IT aj profesionálne.

Za špecifickú oblasť je možné označiť tzv. *vzdelávanie založené na hrách* (z. ang. *Game Based Learning* (GBL), ale tiež známe aj ako *Serious Games*). Hráč hraním takejto hry získava určité schopnosti alebo získava nové vedomosti [1]. Aj keď sa tento termín používa najviac v súvislosti s počítačovými hrami, je možné do tejto oblasti zaradiť akékoľvek hry, napr. orientačný beh, počas ktorého jeho hráči lúštia rozličné logické úlohy.

Čo však v prípade, ak chceme svojich študentov naučiť programovať a napríklad aj naučiť vytvárať vlastné počítačové hry? Ako postupovať v tomto prípade, resp. aké nástroje je možné použiť na dosiahnutie tohto cieľa?

V nasledujúcom texte opíšem metodiky na výučbu *základov algoritmizácie* a *základov objektového programovania*, ktoré je možné s výhodou použiť na výučbu programovania na základných a stredných školách.

Materiál byl zpracován v rámci projektu "Systémová podpora trvalého profesního rozvoje (CPD) pedagogických pracovníků propojením pedagogické fakulty se školami na Jižní Moravě – **EDUCOLAND**"

Projekt je spolufinancován Evropským sociálním fondem a státním rozpočtem ČR.

Princípy na dosiahnutie úspešného

Pri výučbe programovania sa zvykneme či už vedome alebo nevedome dopúšťať viacerých chýb. Niektoré z nich je možné nájsť v publikáciách [2] alebo [3]. Rovnako však je možné osvojiť si niekoľko princípov, ktoré je dobré pri výučbe akéhokoľvek typu programovania aplikovať [2], [3], [4]. Medzi tie základné je možné zaradiť:

- *Čo najskôr umožniť študentom programovať*. Študent nemusí priamo na prvom stretnutí vedieť vypočítať faktoriál čísla *N*, ale navigovať robota pomocou príkazov `KROK` a `VLAVO_VBOK` naozaj dokáže.

- *Príklady musia predstavovať pre študentov výzvu*. Nie je nič horšie, ako znížiť u študentov motivačný faktor nezaujímavými úlohami. Ak je úloha naopak zaujímavá, študenti ju chcú riešiť sami alebo prídu na cvičenie už s vyriešenou úlohou. Ako veľmi vhodné úlohy sa v tomto prípade javí programovanie rozličných viac či menej zložitých hier.

- *Technológie navyše nie sú pri výučbe princípov dôležité.* Často venujeme pri výučbe programovania úsilie predstavovaním rozličných technológií, ako napr. tvorbe používateľského rozhrania (často ako sprievodná téma kurzov venovaných objektovému programovaniu). Študenti budú mať dosť času na zvládnutie týchto tém v ďalších nadväzujúcich kurzoch. Horšie však je, ak nezvládnu základy v udenej problematike.

V ďalšom texte sa k týmto princípom ešte vrátim pri hodnotení vybranej metodiky na výučbu programovania.

Základy algoritmizácie s robotom Karlom

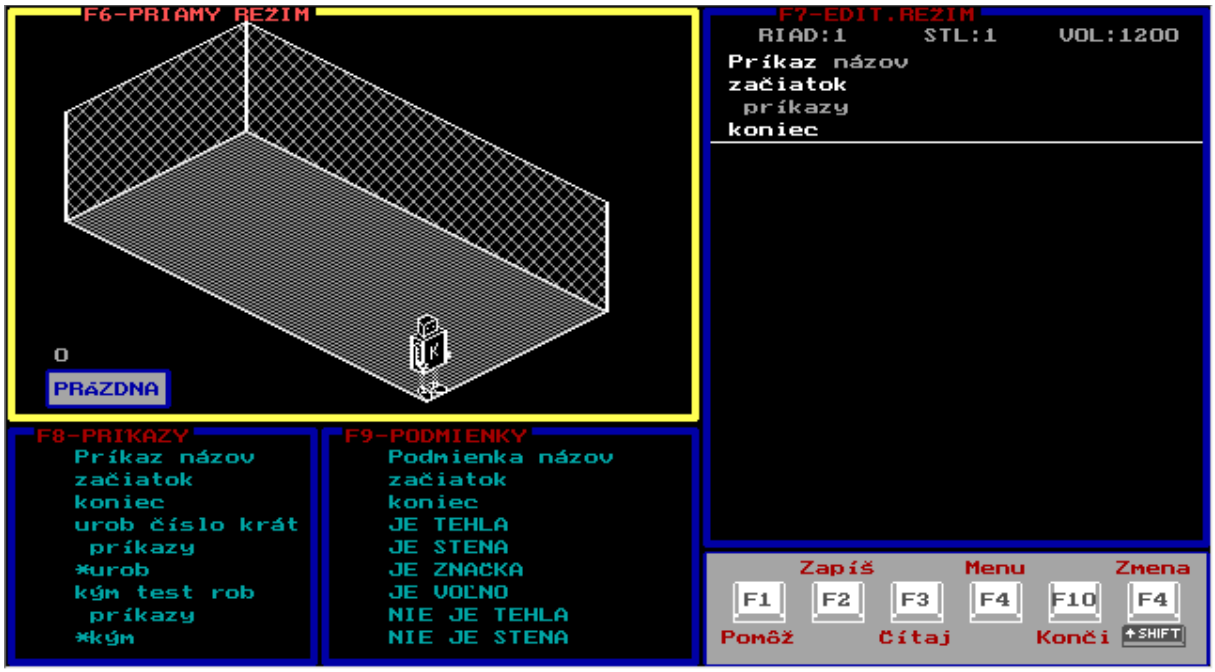
Mikrosvet robota Karla

Metodika programovania s robotom Karlom nie je v našich končinách nijako neznáma. Autorom tejto metodiky je Richard Pattis [5] a u nás sa stala veľmi populárnou na prelome 80-tych a 90-tych rokov vďaka veľmi populárnej domácej 3D verzii pre mikropočítačoch PMD (obr. 1). Veľkej popularizácii tejto metodiky pomohli aj domáce časopisy *Zenit pionierov* a *ABC*.

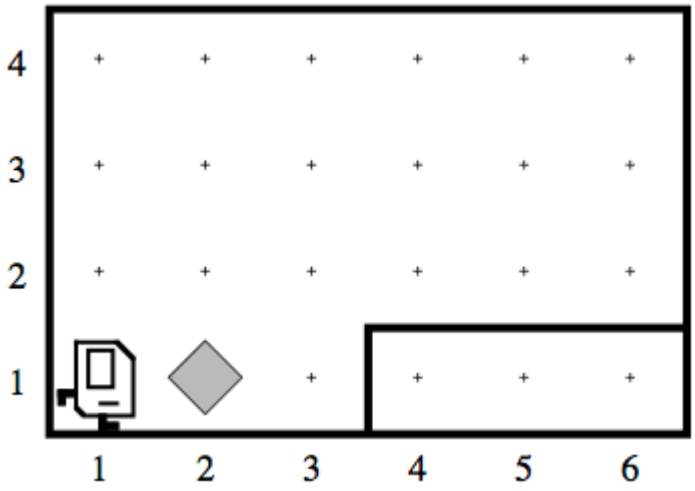
Mikrosvet robota Karla je reprezentovaný horizontálnymi (v orig. *streets*) a vertikálnymi (v orig. *avenues*) ulicami a Karel sa môže pohybovať len po ich priesečníkoch. V priamej ceste môžu Karlovi zabrániť steny, ktoré sú nepriechodné. Okrem toho môže Karel cestou zbierať a pokladať značky (v orig. *beepers*) do batohu „neobmedzenej“ veľkosti [5] [6]. Ukážka jednej situácie v svete robota Karla je na obr. 2.

Materiál byl zpracován v rámci projektu "Systémová podpora trvalého profesního rozvoje (CPD) pedagogických pracovníků propojením pedagogické fakulty se školami na Jižní Moravě – **EDUCOLAND**"

Projekt je spolufinancován Evropským sociálním fondem a státním rozpočtem ČR.



Obrázok 1: Obrazovka z PC verzie robota Karla 3D



Obrázok 2: Robot Karel a jeho svet

Zoznam príkazov, pomocou ktorých je možné robota Karla riadiť, nie je dlhý:

- krok vpred
- otočenie robota o 90 stupňov vľavo

Materiál byl zpracován v rámci projektu "Systémová podpora trvalého profesního rozvoje (CPD) pedagogických pracovníků propojením pedagogické fakulty se školami na Jižní Moravě – **EDUCOLAND**"

Projekt je spolufinancován Evropským sociálním fondem a státním rozpočtem ČR.

- položenie a zobratie značky z aktuálnej pozície
- vypnutie a zapnutie robota

Okrem týchto základných príkazov je Karel vybavený ešte niekoľkými senzormi, pomocou ktorých vie robiť rozhodnutia (napr. senzor pre zistenie, či je pred Karlom stena alebo či sa na aktuálnej pozícii nenachádza žiadna značka).

Robot Karel na základných školách

Počas svojej 33 ročnej existencie sa je možné stretnúť s mnohými implementáciami robota Karla. Či už s implementáciami v rozličných jazykoch (napr. v jazyku C [5] alebo Java[1]), s využitím rozličných programátorských paradigiem (napr. štruktúrované programovanie [5], [7] alebo objektové programovanie[2]) až po modifikácie pôvodného nápadu a špecifické prostredia s vlastným jazykom (napr. *Robot Emil*[3] alebo projekt *XKarel*[4]).

Na základných školách sa však miesto písania kódu preferuje využívanie „detských programovacích jazykov“, akými sú v našich končinách dobre známy *Baltík*[5], variácie na korytnačiu grafiku a v súčasnosti stále viac populárny *Scratch*[6]. Tým, že žiaci nie sú nútení písať kód priamo, je možné minimalizovať možnosť vzniku najrozličnejších chýb spojených so syntaxou. Miesto toho programujú „iným“ spôsobom (napr. skladajú bloky, ktoré sú reprezentované graficky, do funkčných celkov).

Robot Karel sa taktiež dočkal podobných prevedení a jedným z nich je aj demo na vizuálny prgramovací editor *blockly*[7] od spoločnosti Google. V tomto deme sa nachádza 10 úrovní (tém), v ktorých je cieľom vždy dostať svojho panáčika do cieľa. Táto ukážka vhodne využíva aj herné princípy, kde hráč (študent) dostane na začiatku len sadu istých blokov a vyriešiť problém musí za daný počet krokov. V rámci týchto desiatich úrovní sa študenti zoznámia so samostatnými témami týkajúcimi sa programovania a algormitzácie, ako sú volanie príkazov, cykly a podmienené vykonávanie. Pritom však celý čas „programujú“, pretože na to, aby prešli do ďalšej úrovne musia vytvoriť program.

1 Kurz CS106A – Programming Methodology na Stanfordskej univerzite http://cs.stanford.edu/people/eroberts/courses/cs106a/

2 Karel J Robot - http://csis.pace.edu/~bergin/KarelJava2ed/Karel++JavaEdition.html

3 http://www.emil.input.sk/

4 http://xkarel.sourceforge.net/eng/

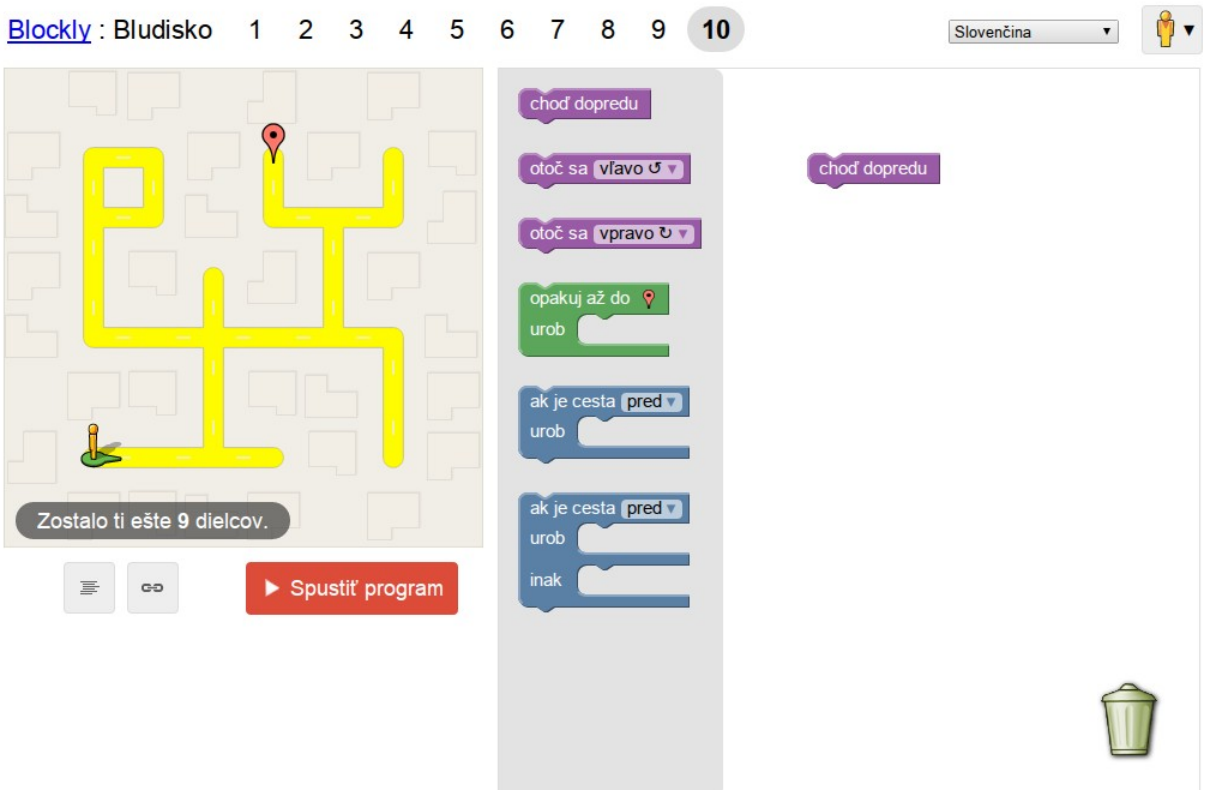
5 http://www.sgpsys.com/cz/

6 http://scratch.mit.edu/

7 http://code.google.com/p/blockly/

Materiál byl zpracován v rámci projektu "Systémová podpora trvalého profesního rozvoje (CPD) pedagogických pracovníků propojením pedagogické fakulty se školami na Jižní Moravě – **EDUCOLAND**"

Projekt je spolufinancován Evropským sociálním fondem a státním rozpočtem ČR.



Obrázok 3: Začiatok záverečnej úrovne variácie v Blockly (Maze)

Príklady úloh

Vďaka svojej 34 ročnej histórii existuje mnoho zbierok rozličných úloh pre rozličné variácie a implementácie robota Karla. Zadanie programátorskej úlohy vo všeobecnosti zvykne obsahovať aj ukážkový vstup a výstup – výstup reprezentuje údaje, ktoré vygeneruje správne vyriešený program, ak na vstupe dostal zadané údaje. V prípade robota Karla sa miesto vstupných a výstupných údajov zvykne uvádzať počiatočná (ako bude vyzerať svet vtedy, keď sa v ňom Karel ocitne) a koncová sitúcia (ako bude svet vyzerať potom, keď Karel dokončí svoju činnosť). Príklad takejto úlohy môže vyzerať nasledovne:

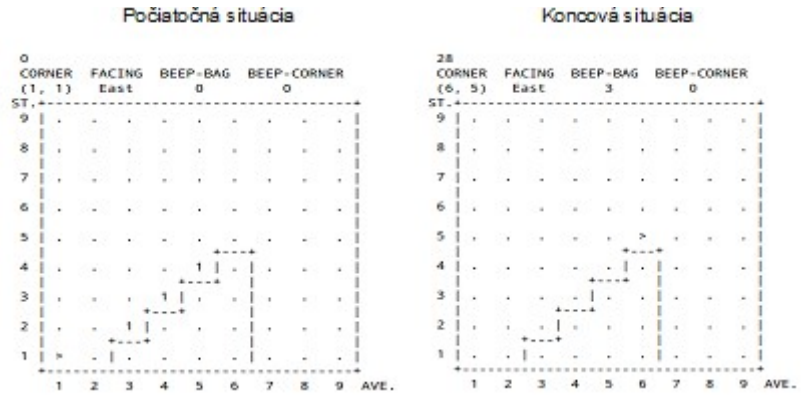
Vytvorte pre Karla program, pomocou ktorého mu pomôžete vyšplhať sa po schodoch nahor, až na najvyšší stupienok. Cestou pomôžte Karlovi pozbierať všetky značky, ktoré na schodoch nájde. Po vyšplhaní sa na najvyšší schod Karel všetky značky položí.

Hodnotenie metodiky

Hodnotiť metodiku sa pokúsim prostredníctvom možností aplikácie princípov uvedených v úvodnej časti tohto príspevku.

Materiál byl zpracován v rámci projektu "Systémová podpora trvalého profesního rozvoje (CPD) pedagogických pracovníků propojením pedagogické fakulty se školami na Jižní Moravě – **EDUCOLAND**"

Projekt je spolufinancován Evropským sociálním fondem a státním rozpočtem ČR.



Obrázok 4: Príklad počiatočnej a koncovej situácie pre robota Karla

1. Čo najskôr umožniť študentom programovať

Metodiku sa mi podarilo použiť niekoľkokrát na rozličných typoch študentov – od žiakov ZŠ až po študentov VŠ. Rovnako tak sa mi podarilo použiť aj dvojaký spôsob prístupu – programátorský aj „skladací“ (blockly). Pri čistom programátorskom štýle dokázali študenti po hodine a pol vytvárať jednoduché programy a s nimi riešiť jednoduché problémy robota Karla. Pri „skladacom“ štýle aplikovanom na *blockly* som študentom povedal iba toľko, že „prví traja, ktorí prekonajú desiatu úroveň, dostanú jednotku“. A na zvládnutie desiatej úrovne im stačilo zhruba 45 minút.

Táto metodika sa nám veľmi osvedčila na úvodných hodinách programovania aj na univerzite (Technická univerzita Košice, Slovensko), kde ju používame podobne ako v kurze *CS106A* na *Stanfordskej univerzite*. A aj napriek tomu, že vyzerá detsky, je výborne aplikovateľná aj na univerzitných študentov, ktorí ešte nikdy neprogramovali, aby sa základmi programovania zoznámili.

2. Príklady musia predstavovať pre študentov výzvu

Vďaka množstvu existujúcich príkladov je možné nájsť rozličné typy úloh – od tých menej zložitých, pri ktorých sa Karel len presúva z miesta A na miesto B až po algoritmické „hlavolamy“, pri ktorých napr. je potrebné Karla dostať von z bludiska alebo vymyslieť, ako (bez použitia premenných) zdvojnásobiť počet značiek na konkrétnej pozícii. Študenti, ktorí mali s programovaním už predchádzajúce skúsenosti, zvykli na týchto úvodných hodinách ohŕňať nosom. Ale aj oni sa zvykli zapotiť pri riešení niektorých zložitejších úloh.

3. Technológie navyše nie sú pri výučbe princípov dôležité

Pri používaní metodiky programovania s robotom Karlom nie ste závislý od žiadnych technológií navyše. Ak teda nerátame inštaláciu špecifického programového vybavenia, aby bolo možné robota Karla spustiť. Po spustení však sa už študenti venujú len riešeniu samotných problémov.

Materiál byl zpracován v rámci projektu "Systémová podpora trvalého profesního rozvoje (CPD) pedagogických pracovníků propojením pedagogické fakulty se školami na Jižní Moravě – **EDUCOLAND**"

Projekt je spolufinancován Evropským sociálním fondem a státním rozpočtem ČR.

Objektové programovanie

Existujúce metodiky a nástroje

Vďaka dostupnosti rozličných nástrojov a technológií sa objektové programovanie začalo dostávať aj na stredné školy. Pre jednoduchšie zvládnutie objektovo-orientovaných princípov je možné použiť niekoľko metodík, ako napr. *objects first* (najprv objekty), *patterns first* (najprv vzory) alebo *architecture first* (najprv architektúra). Stále sa je však možné stretnúť aj s tzv. „klasickým“ prístupom, kedy sa dôraz kladie hlavne na syntax jazyka a objektové programovanie je prezentované len ako ďalšia jazyková konštrukcia. Poprípade sa je možné s výučbou objektového programovania stretnúť vo forme tvorby aplikácií s grafickým používateľským rozhraním, kedy študenti programujú len obsluhu vzniknutých udalostí, ako napr. kliknutie na tlačidlo „OK“. Takýto prístup by sa však dal charakterizovať ako *objektové programovanie bez objektového programovania*, pretože študenti sami vlastné triedy a objekty nevytvárajú, ale všetko za nich vytvára samotné prostredie.

V našich končinách je asi najznámejším prístupom „*objects first*“ vďaka popularizátorovi tejto metodiky Rudolfovi Pecinovskému, ktorý vydal na túto tému už niekoľko publikácií (napr. publikácia [8] sa stala ľudovo známou ako „*modrá učebnica Javy*“). Pecinovský si osvojil prístup prezentovaný v publikácii [9] a nástroj *BlueJ*[8], ktorý vznikol s cieľom priblížiť objektovo orientovaný prístup novým študentom.

V súčastnosti existuje mnoho nástrojov ponúkajúcich túto metodiku prístupu, ako napr. *Greenfoot*[9] (viď obrázok 5), ktorý vznikol na základe projektu *BlueJ* a zameriava sa na výučbu objektovo orientovaných princípov pomocou tvorby jednoduchých hier. Alebo projekt *Alice*[10], ktoré poskytuje kompletné prostredie na tvorbu akýchsi „virtuálnych príbehov“ s cieľom prilákať ku programovaniu hlavne dievčatá. Tento projekt podporuje aj spoločnosť *ElectronicArts* a poskytla do neho grafiku zo svojej hernej série *The Sims*.

Samostatnú kategóriu by mohli predstavovať tzv. *inteligentné vzdelávacie systémy* (*inteligent tutoring systems*). Takýto systém poskytuje študentovi okamžitú spätnú väzbu bez akéhokoľvek zásahu učiteľa. To znamená, že systém sám dokáže viesť študenta pri jeho riešení problému správnym smerom, pričom ho dokáže vždy naviesť správnym smerom, ak z neho pri riešení problému vybočí.

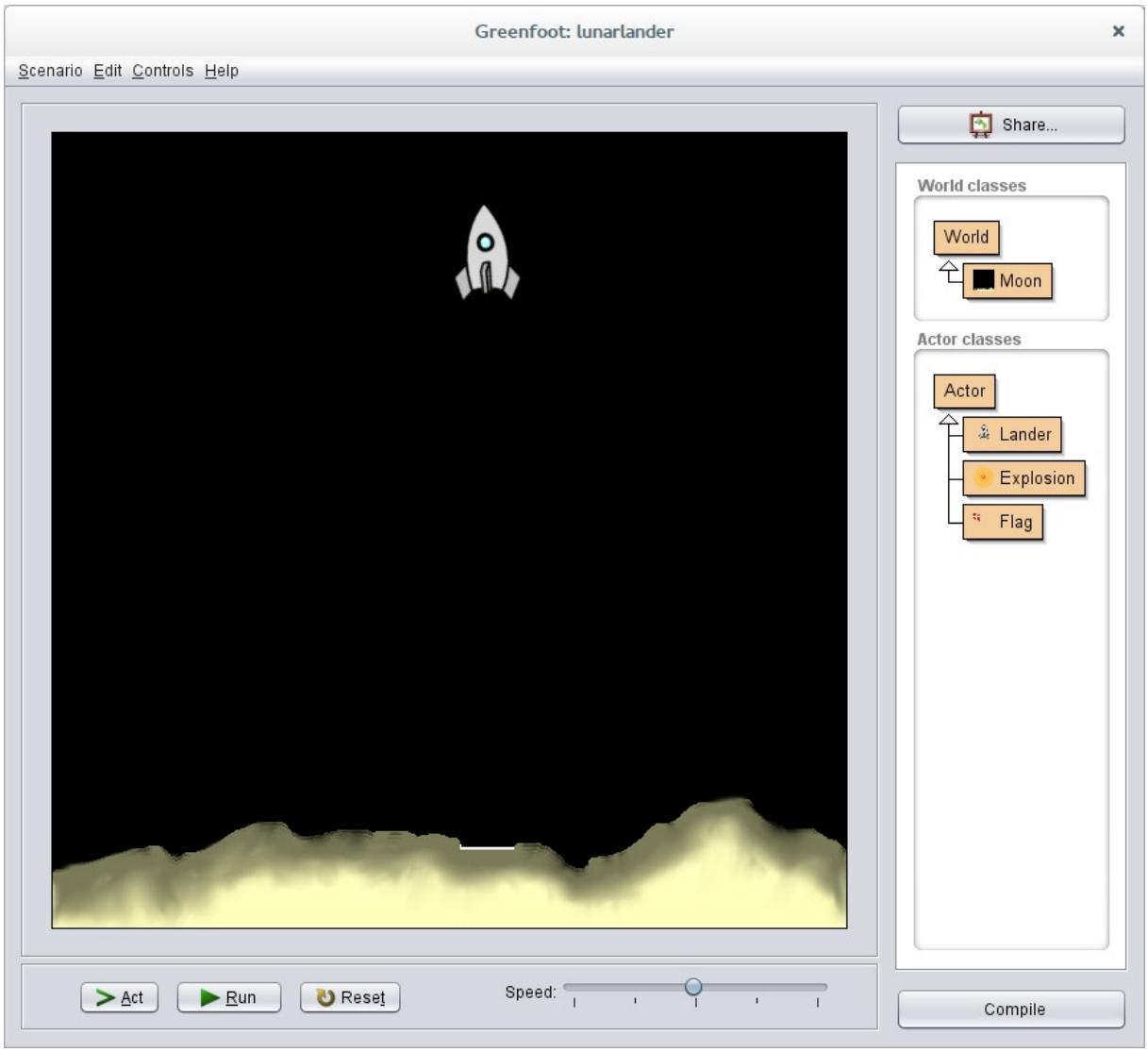
8 http://www.bluej.org

9 http://www.greenfoot.org

10 http://www.alice.org

Materiál byl zpracován v rámci projektu "Systémová podpora trvalého profesního rozvoje (CPD) pedagogických pracovníků propojením pedagogické fakulty se školami na Jižní Moravě – **EDUCOLAND**"

Projekt je spolufinancován Evropským sociálním fondem a státním rozpočtem ČR.



Obrázok 5: Obrazovka z prostredia Greenfoot s ukážkovou aplikáciou LunarLander

V rámci hľadania nových prístupov ku výučbe objektového programovania sme jeden takýto systém s názvom OAT (*Object Access Tool*) [10] na pôde Technickej univerzity v Košiciach vytvorili. Používame ho na prvom cvičení predmetu *Objektového programovanie* a cieľom tohto cvičenia je sa hrať. Na študentov čakajú spolu 3 levely grafickej akčnej hry, pričom len prvý sa dá prejsť štandardným hraním. Pri zdolávaní ostatných levelov musia študenti „podvádzať“. A aby svoje podvody zvládli, musia porozumieť základom objektového programovania, ktoré pri aplikovaní svojich podvodov použijú. Tak napríklad – keď im dôjde strelivo, majú možnosť vytvoriť nové inštancie streliva, ktoré potom už len umiestnia na podlahu.

Systém s hráčom komunikuje pomocou samostatného okna, v ktorom mu postupne predkladá úlohy, ktoré má študent riešiť (viď obrázok 6). Okrem úloh samotných mu tiež vo

Materiál byl zpracován v rámci projektu "Systémová podpora trvalého profesního rozvoje (CPD) pedagogických pracovníků propojením pedagogické fakulty se školami na Jižní Moravě – **EDUCOLAND**"

Projekt je spolufinancován Evropským sociálním fondem a státním rozpočtem ČR.

vhodných chvíľach predstavuje samotné princípy objektového programovania, ktoré mu môžu pomôcť pri riešení jednotlivých úloh.

Obrázok 6: Inteligentný systém OAT – vľavo systém učí hráča nové znalosti a zadáva mu novú úlohu a vpravo je obrazovka so samostnou hrou

Počas ďalších cvičení už študenti samozrejme vytvárajú vlastný kód pomocou metodiky *objects first*. Študenti vytvárajú jednoduché triedy a pomocou nástroja OAT z nich následne vytvárajú objekty a umiestňujú ich do herného sveta. V ňom zasa pomocou OAT-u vedia zobrazovať stav členských premenných jednotlivých objektov a volať ich metódy. Grafická forma prezentácie jednotlivých objektov a zmena ich animácií pri zmene ich stavu napomáha študentom jednoduchšie porozumieť princípom objektového programovania. Príkladom môže byť napr. trieda reprezentujúca dvere a metódy `otvorit()` a `zatvorit()`. Volanie týchto metód nad existujúcou inštanciou dverí, ktoré sa následne otvoria resp. zatvoria prehratím príslušnej animácie je podstatne názornejšie, ako trieda reprezentujúca osobu s metódami na zmenu základných vlastností takejto osoby.

Objektové programovanie na stredných školách

Skúsenosti s nasadením objektového programovania na stredných školách sú rôzne. Najčastejšie, s čím som sa mohol sám stretnúť, je používanie objektového jazyka a príbuzných technológií, s kladením dôrazu práve na technológie bez osvojovania si objektových princípov. Najviditeľnejší je tento prístup vtedy, keď sa študenti venujú tvorbe používateľského prostredia a ošetrovaniu udalostí ako kliknutie na konkrétne tlačidlo, zatvorenie okna alebo výber položky v zozname. Obľúbenými jazykmi a prostrediami sú pre tieto účely napr. *Visual Basic* alebo jazyk *C#* a prostredie *Microsoft Visual Studio*. Ukážkovým typom takejto aplikácie je napr. kalkulačka, kedy študenti nerobia žiadny objektový návrh, ale sústreďujú sa na návrh grafického používateľského rozhrania, funkcionalitu stláčania jednotlivých kláves a vyčíslenie výsledného výrazu.

Materiál byl zpracován v rámci projektu "Systémová podpora trvalého profesního rozvoje (CPD) pedagogických pracovníků propojením pedagogické fakulty se školami na Jižní Moravě – **EDUCOLAND**"

Projekt je spolufinancován Evropským sociálním fondem a státním rozpočtem ČR.

Metodika *objects first* je naopak pre študentov stredných škôl naozaj vhodná. Nie len že sa zameriava na objektový návrh od samotného začiatku, ale pri vhodnom zostavení úloh alebo výsledného projektu, povzbudzuje vlastnú kreativitu študentov.

Túto metodiku odporúčam používať v kombinácii s niektorým s horeuvedených prostredí, pretože každé jedno z nich má veľmi blízko od návrhu k implementácii. Z vlastnej skúsenosti však musím povedať, že nie je najvhodnejšie v takomto nástroji zostať príliš dlho. Keďže študenti nemajú skúsenosť s iným nástrojom, budú sa snažiť v budúcnosti implementovať aj rozsahovo väčšie projekty, pomocou nástrojov ktoré poznajú – teda pomocou nástrojov určených pre aplikovanie metodiky *objects first*.

Prostredie, ktoré sa javí ako vhodné, je prostredie *BlueJ* a to najmä kvôli Rudolfovi Pecinovskému a jeho publikáciám, ktoré môžu poslúžiť ako učebnice. Jednou z nevýhod môže byť nie príliš atraktívny výstup aplikácií, ktoré je možné týmto spôsobom dosiahnuť (väčšinou textový výstup).

Zaujímavejším prostredím sa preto javí prostredie *Greenfoot*, pomocou ktorého je možné vytvárať jednoduché grafické počítačové hry (viď príklad na obrázku 8). Nevýhodou môže byť nutná znalosť jednoduchého softvérového rámca, ktorý umožňuje jednoduchým spôsobom pracovať s obrázkami, zvukmi a ovládaním.

Na základe skúseností s týmito prostrediami môžem odporučiť aj nástroj OAT, ktorý sme vytvorili ako rozšírenie profesionálneho programátorského prostredia *Netbeans*. Nástroj ponúka výhody, ktoré ponúka toto IDE, ale poskytuje aj jednoduchý nástroj na prezentáciu návrhu projektu vo forme diagramu tried, pričom je možné pracovať aj so živými inštanciami a meniť ich vlastnosti (viď obrázok 9). Rovnako sa však vyžaduje znalosť jednoduchého softvérového rámca, pomocou ktorého je podobne ako v prípade prostredia *Greenfoot* možné vytvárať samotné objekty v hernom svete, pracovať s ich animáciami (obrázkami) a s mapou reprezentujúcou herný svet.

Príklady úloh

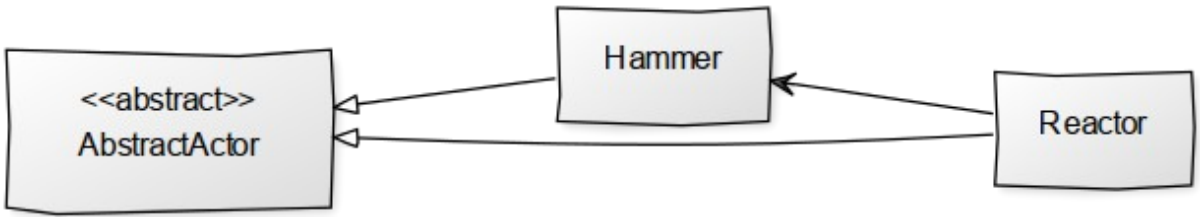
V nasledujúcom texte sa bude nachádzať jeden kompletný krok scenára cvičenia, v ktorom majú študenti vytvoriť kladivo na opravenie poškodeného reaktora. Pri vytváraní jednotlivých úloh sa snažíme, aby zadanie úlohy bolo čo najkratšie (jedna rozkazovacia veta) a v prípade potreby doplníme špecifikáciu úlohy v ďalšom texte.

Primary Objective #1: Fixing Broken Reactor Object

Ako ste si všimli, pracovať s atómovým reaktorom nie je žiadna sranda. Hlavne keď sa začne prehrievať a neexistuje spôsob, ako ho opraviť. V tomto kroku sa teda pokúsime poškodený reaktor včas opraviť vhodným nástrojom - kladivom.

Materiál byl zpracován v rámci projektu "Systémová podpora trvalého profesního rozvoje (CPD) pedagogických pracovníků propojením pedagogické fakulty se školami na Jižní Moravě – **EDUCOLAND**"

Projekt je spolufinancován Evropským sociálním fondem a státním rozpočtem ČR.



Obrázok 7: Diagram tried, ktorý vyjadruje vzťah tried Reactor a Hammer.

Úloha 1.1

Vytvorte triedu `Hammer`*, ktorá bude reprezentovať objekt kladivo potrebný na opravu reaktoru a inicializujte ju.*

Animácia reprezentujúca kladivo na mape je uložená v súbore `hammer.png`. Kladivo nebude mať žiadne špeciálne členské premenné.

Obrázok 8: Animácia hammer.png (rozmery obrázka: 16x16, trvanie prehrávania: 10)

Úloha 1.2

Vytvorte v triede `Reactor` *metódu* `repair()`*, pomocou ktorej bude možné reaktor opraviť.*

Metóda nebude nič vracať, ale bude mať jeden parameter typu `Hammer`. Jej správanie bude nasledovné:

- Metóda bude pracovať iba vtedy, ak je parameter metódy referenciou na inštanciu triedy `Hammer` (nie je `null`).
- Počas opravy sa teplota jadra schladí na hodnotu *1000* stupňov, ale len v prípade, ak už bola vyššia.
- Použitie kladiva zníži poškodenie reaktora **o** *50%*. Pozor ale na to, aby hodnota poškodenia neklesla pod *0*.

Poznámka: Pri implementácii nezabudnite na to, že okrem úrovne poškodenia a teploty je potrebné zmeniť aj obrázok, ktorý vizualizuje teplotu reaktora.

Úloha 1.3

Overte správnosť svojej implementácie.

Vytvorte na mape inštanciu reaktora a niekoľkých kladív. Následne reaktor prehrejte a pokúste sa ho opraviť použitím kladiva.

Materiál byl zpracován v rámci projektu "Systémová podpora trvalého profesního rozvoje (CPD) pedagogických pracovníků propojením pedagogické fakulty se školami na Jižní Moravě – **EDUCOLAND**"

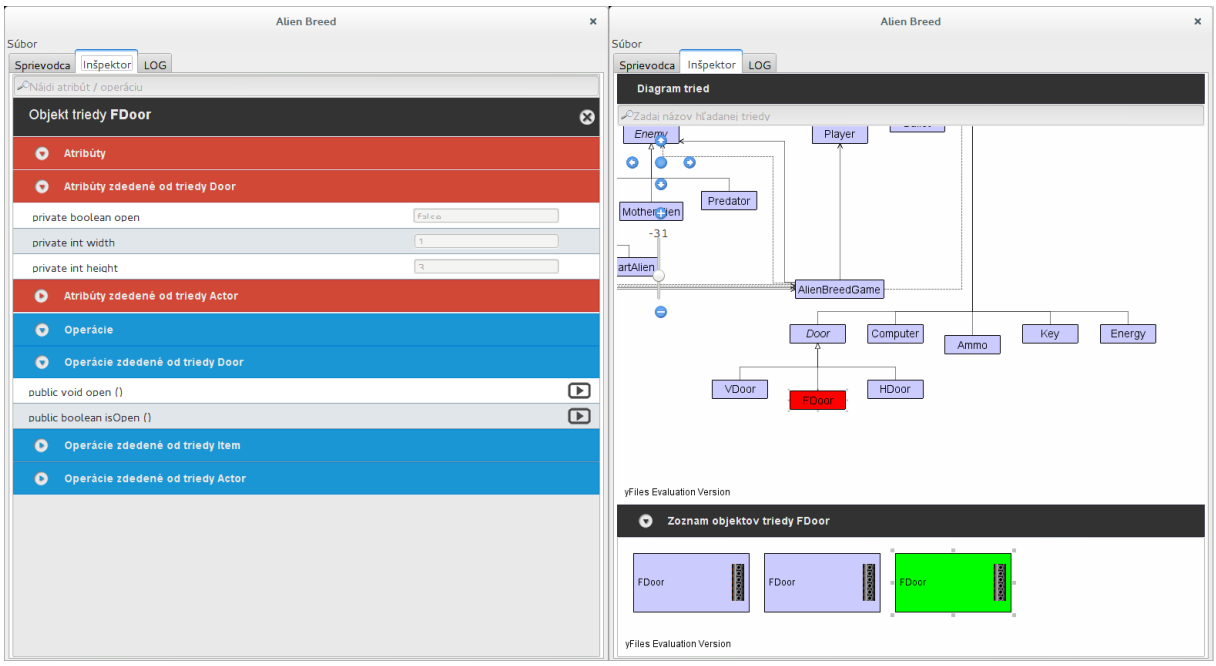
Projekt je spolufinancován Evropským sociálním fondem a státním rozpočtem ČR.

Hodnotenie metodiky

V tejto časti sa pokúsim zhodnotiť výučbu objektového programovania pomocou metodiky *objects first* s využitím nástrojov *Greenfoot* alebo *OAT*.

1. Čo najskôr umožniť študentom programovať

Prvé stretnutia sa dajú realizovať bez nutnosti priameho písania kódu. Aj keď sa síce nejedná o programovanie v „surovej“ forme, riešenie úloh, pri ktorých študenti vytvárajú nové objekty len klikaním myšou a jednotlivé metódy tiež volajú explicitne kliknutím myšou na potrebnú položku, je možné rovnako považovať za programovanie. V tomto práve veľmi napomáhajú prostredia podporujúce metodiku výučby objektového programovania *objects first*.



Obrázok 9: Inšpektor z nástroja OAT umožňuje prezerať aktuálny stav objektu (vľavo) a zobraziť diagram tried aktuálneho projektu (vpravo)

2. Príklady musia predstavovať pre študentov výzvu

Obrovskou výzvou pre študentov je určite možnosť vytvoriť si v rámci predmetu svoju vlastnú počítačovú hru. Na tomto fakte už dlhodobo staviame skladbu predmetov venovaných programovaniu, kedy študenti počas trvania predmetu (kurzu) postupne svoju hru vytvárajú. Pri ich vytváraní samozrejme netreba zabúdať na kreativitu študentov a vytvoriť aj dostatočný priestor na to, aby do hry vedeli okrem pripravených častí vložiť aj svoje vlastné (napr. mapu svojho vlastného sveta alebo vytvorenie svojho vlastného herného prvku). Takýto typ úloh pre

Materiál byl zpracován v rámci projektu "Systémová podpora trvalého profesního rozvoje (CPD) pedagogických pracovníků propojením pedagogické fakulty se školami na Jižní Moravě – **EDUCOLAND**"

Projekt je spolufinancován Evropským sociálním fondem a státním rozpočtem ČR.

nich predstavuje podstatne väčšiu výzvu ako riešenie príkladov typu „*Vypíšte na obrazovku čísla od 1 do 10*“.

Samotné cvičenia sú organizované prírastkovo, kedy študenti vytvárajú výslednú podobu svojej hry postupne každý týždeň v malých celkoch. Je veľmi dôležité, aby každý prírastok tvoril uzavretý celok a projekt bol celý čas spustiteľný. Samozrejme študenti každý týždeň pri riešení jednotlivých úloh aplikujú nové znalosti.

3. Technológie navyše nie sú pri výučbe princípov dôležité

Pokiaľ sa rozhodnete pre výučbu objektového programovania použiť nástroj *BlueJ*, je použitie technológií ako napr. *Java2D* pre kreslenie jednoduchých primitív, čisto na vás. Prostredie vás totiž pri zostavovaní a následnom riešení úloh nijako neobmedzuje v používaní technológií a viete si vystačiť aj so štandardným vstupom a výstupom.

Ak sa však rozhodnete použiť nástroje ako *Greenfoot* alebo *OAT*, základnej znalosti pripraveného softvérového rámca, ktoré tieto prostredia ponúkajú sa už nevyhnete. Tieto rámce sú však dostatočne jednoduché na to, aby nevytvárali neprekonateľnú prekážku, ale zjednodušujú prácu hlavne s použitím herných prvkov.

Záver

Naučiť sa programovať je ťažké. Naučiť programovať niekoho iného je však ešte ťažšie a tento proces predstavuje značnú výzvu už pre samotného učiteľa.

Tak, ako existuje mnoho prístupov k programovaniu, tak existuje aj mnoho metodík na výučbu programovania. Tieto metodiky poskytujú konkrétny spôsob, pomocou ktorého je možné zjednodušiť študentom porozumenie a osvojenie si potrebných programátorských prístupov a techník. A čo je nemenej dôležité – existujú aj hotové prostredia, pomocou ktorých si je možné niektoré prístupy osvojiť ešte jednoduchšie.

V tomto príspevku som sa snažil prezentovať niektoré z možností a nástrojov pre podporu výučby programovania (základov algoritmizácie) a objektového programovania v prostredí základných a stredných škôl. Každý z týchto nástrojov je reprezentantom jedného prístupu k programovaniu a teda v prípade ich použitia sa očakáva istá následnosť – najprv základy algoritmizácie a až potom objektové programovanie. Rudolf Pecinovský však vo svojej knihe [8] ukazuje kombináciu prístupov výučby objektového programovania spolu so základmi algoritmizácie. Jedná sa však o prístup, ktorý je možné aplikovať najskôr až u študentov stredných škôl.

Konečná aplikácia nových metodík pre výučbu programovania je však na učiteľovi samom. Hovoril som o tom, že úlohy pre študentov by mali byť istou výzvou, ale pred tou najväčšou výzvou stojí aj tak sám učiteľ, ktorý sa rozhoduje, či tieto nové možnosti preskúma

Materiál byl zpracován v rámci projektu "Systémová podpora trvalého profesního rozvoje (CPD) pedagogických pracovníků propojením pedagogické fakulty se školami na Jižní Moravě – **EDUCOLAND**"

Projekt je spolufinancován Evropským sociálním fondem a státním rozpočtem ČR.

alebo nie. Čas venovaný skúmaniu sa však oplatí, pretože prínos v podobe nových skúseností a zvýšeniu motivačného faktora u samotných študentov za to rozhodne stojí.

Literatúra

1. Perrotta, C., Featherstone, G., Aston, H. and Houghton, E. (2013). Game-based Learning:
   Latest Evidence and Future Directions (NFER Research Programme: Innovation in Education). Slough: NFER. Dostupné na internete http://www.nfer.ac.uk/publications/GAME01/GAME01.pdf (cit. 29.sep.2014)
2. Pecinovský, R.: Jak efektivně učit OOP, 2005. Dostpuné na intenete http://vyuka.pecinovsky.cz/prispevky/2005-SW_Jak_efektivne_ucit_OOP.pdf (cit. 29.sep.2014)
3. Pecinovský, R.: Výuka objektově orientovaného programování žáků základních a středních škol, Objekty 2003 – sborník příspěvků konference, VŠB, Ostrava 2003. ISBN 80-248-0274-0. Dostupné na internete http://www.vyuka.pecinovsky.cz/prispevky/Vyuka_OOP_zaku_zakladnich_a_strednich_skol.pdf (cit. 29.sep.2014)
4. Biňas, M., Novák, M., Michalko, M., Jakab, F.: Ako motivovať študentov softvérového inžinierstva, v zborníku konferencie SILESIAN MOODLE MOOT 2009, Ekonomická fakulta VŠB-TU Ostrava. Dostupné na internete http://mirek.s.cnl.sk/~mirek/sites/all/modules/pubdlcnt/pubdlcnt.php?file=http://s.cnl.sk/~mirek/files/ako.motivovat.studentov.softveroveho.inzinierstva.pdf&nid=450 (cit. 29.sep.2014)
5. Pattis, R., Roberts, J., and Stehlik, M. Karel the robot: a gentle introduction to the art of programming. Wiley, 1995.
6. Biňas, M.: Robot Karel sa učí jazyk C, príspevok na konferencii OSS Conf 2012, Žilina, Slovensko. Dostupné na internete: http://mirek.s.cnl.sk/~mirek/sites/all/modules/pubdlcnt/pubdlcnt.php?file=http://s.cnl.sk/~mirek/files/oss-conf-2012.pdf&nid=450 (cit. 29.sep.2014)
7. Pecinovský, R.: Základy algoritmizace, Grada. Dostupné na internete: http://publikace.pecinovsky.cz/Zaklady_algoritmizace.pdf
8. Pecinovský, R.: Myslíme objektově v jazyku Java, Grada, 2008, 576s, ISBN 9788024726533
9. Barnes, D. J.; Kölling, M. Objects First with Java: A Practical Introduction Using BlueJ. Prentice Hall, 2002, ISBN: 0-13-044929-6.
10. Livovský J., Biňas M., Porubän J.: Teaching Object-oriented Programming using Object Benches: Practical Experience, v zborníku konferencie SDOT 2013, ISBN 9788086847665. Dostupné na internete http://it4kt.cnl.sk/c/op/download/sdot2013.pdf (cit. 29.sep.2014)

Materiál byl zpracován v rámci projektu "Systémová podpora trvalého profesního rozvoje (CPD) pedagogických pracovníků propojením pedagogické fakulty se školami na Jižní Moravě – **EDUCOLAND**"

Projekt je spolufinancován Evropským sociálním fondem a státním rozpočtem ČR.