BlueJ – vývojové prostředí pro výuku jazyka Java

Rudolf Pecinovský[1]

[1]Amaio Technologies, Inc., Třebohostická 14, 100 00, Praha 10
rudolf@pecinovsky.cz

Abstrakt. Příspěvek seznamuje s vývojovým prostředím *BlueJ*, které bylo vyvinuto speciálně pro počáteční fáze výuky objektově orientovaného programování v jazyku Java. *BlueJ* studentům umožňuje navrhovat diagram tříd vyvíjené aplikace v zjednodušené verzi jazyka UML. Z tohoto diagramu pak automaticky vytváří kostry tříd a naopak všechny změny v těchto třídách následně promítá do diagramu. Hlavní výhodou prostředí *BlueJ* je však jeho interaktivnost. *BlueJ* umožňuje uživateli přímo posílat zprávy jednotlivým třídám a jejich instancím. Kromě toho přidává i řadu dalších funkcí využitelných zejména při výuce základů objektově orientovaného programování – např. interaktivní generování testů. Článek seznamuje se základními vlastnostmi tohoto vývojového prostředí a ukazuje, jaké výhody může přinést jeho nasazení ve výuce.

Klíčová slova: Java, BlueJ, IDE, výuka

1 Motivace

Prostředí *BlueJ* bylo vyvinuto v druhé polovině devadesátých let Michaelem Köllingem a Johnem Rosenbergem jako reakce na tehdejší stav vývojových prostředí použitelných při výuce programování. Hlavními nepříjemnými vlastnostmi, kterých se chtěli ve svém prostředí vyvarovat, byly:

Nedostatek objektové orientace. Dostupná prostředí jsou sice určena pro vývoj objektově orientovaných programů, avšak sama nejsou příliš objektově orientovaná. Při jejich používání se proto studenti nesoustředí na návrh koncepce vytvářené aplikace, ale na soubory a složky. Nekomunikují s objekty, kterým by posílali zprávy, ale komunikují s řádky vytvářeného kódu.

V poslední době sice některá prostředí nabízejí možnost navrhovat UML diagramy tříd vyvíjených aplikací, ale žádné z těch, která jsou dostupné zdarma, nenabízí automatické vygenerování kódu z vytvořeného diagramu a naopak úpravu diagramu na základě změn v kódu.

Přílišná komplikovanost. Dostupná vývojová prostředí jsou zaměřena především na profesionální programátory, takže nabízejí obrovskou variabilitu nejrůznější nastavení, mezi nimiž začínající uživatel často bloudí.

Řada vyučujících proto vývojová prostředí raději nepoužívá a učí své studenty pracovat s příkazovým řádkem. Studenti pak na počátku výuky tráví zbytečně mnoho času studiem věcí, které nejsou pro návrh objektově orientovaného programu nezbytné, a zbytečně tak odvádějí svoji pozornost od toho, co by mělo být základním cílem výuky. Při tomto způsobu výuky se navíc daleko víc soustředí na soubory a řádky kódu než při použití vývojových prostředí zmiňovaných v předchozím bodě..

Nadměrná koncentrace na GUI. Mnohá prostředí nabízí velice propracované nástroje pro tvorbu grafického uživatelského rozhraní. Studenti pak podlehnou jeho magii a tráví spousty času vylepšováním umístění tlačítek v dialogových oknech místo toho, aby se soustředili na návrh logiky aplikace.

2 Základní charakteristika

BlueJ je vývojové prostředí určené pro první rok univerzitní výuky programování. Je maximálně jednoduché, ale na druhé straně obsahuje všechny funkce, které bychom od výukového vývojového prostředí požadovali. Je koncipováno tak, aby pomohlo vypěstovat u studentů základní návyky, na nichž by bylo možno v dalších ročnících stavět.

2.1 Lokalizace

Velmi důležitou vlastností prostředí, určeného pro začínající studenty, je možnost jeho lokalizace. Tuto vlastnost mnozí vyučující podceňují s poukazem na to, že každý posluchač musí přece umět anglicky. Zapomínají však přitom na to, že důležitou vlastností lokalizace je vedle usnadnění orientace také to, že správně lokalizované prostředí učí své uživatele také správné terminologii.

Naučit se pár slovíček potřebných pro ovládání programu dokáže každý student během krátké doby. Pokud je ale po celou dobu práce s prostředím bombardován anglickými termíny, proniknou tyto terminy i do jeho jazyka. Takový student pak nevytváří třídy, ale classy, nepracuje s rozhraními, ale s interfejsy, neumisťuje třídy do balíčků, ale do pekidžů, instance si neprohlíží ale inspektuje atd.

Prostředí *BlueJ* je lokalizováno do 16 jazyků včetně češtiny (a doufám, že dobře, protože jsem se na české lokalizaci podílel ;-)). Všechny lokalizace jsou součástí standardní instalace. Navíc je její příprava tak jednoduchá, že si může kdokoliv vytvořit během několika dní lokalizaci vlastní.

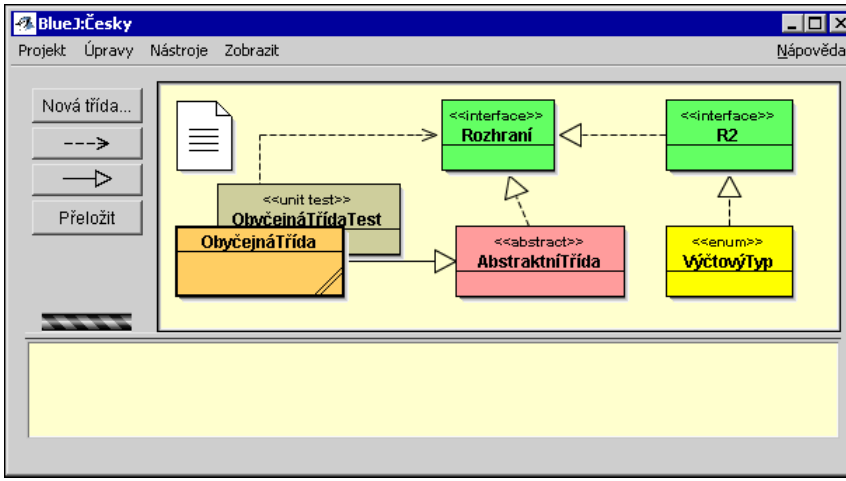
2.2 Diagram tříd

Největší část aplikačního okna *BlueJ* zabírá diagram tříd (viz obr. 1) vytvořený ve zjednodušeném UML. V tomto diagramu nejsou zobrazovány atributy ani metody, ale pouze názvy tříd s případným stereotypem označujícím rozhraní, abstraktní třídu, výčtový typ a testovací třídu.

BlueJ navíc umožňuje jednotlivé druhy tříd vybarvit různými barvami. Originální konfigurace sice této možnosti nevyužívá, ale mně se velice osvědčilo, když v diagramu z dálky svítí, co je která třída zač.

V diagramu tříd zobrazuje *BlueJ* i vzájemnou závislost mezi třídami. Rozeznává přitom obecnou závislost a dědickou či implementační závislost.

- Obecnou závislost, kdy jedna třída používá atribut, jenž je ukazatelem na instanci druhé třídy, zobrazuje prostou čárkovanou šipkou, která vždy vychází z vodorovné hrany závislé třídy a směřuje ke svislé hraně třídy, na níž daná třída závisí.



Obr. 1: Aplikační okno

- Dědičnost tříd a rozhraní a implementaci rozhraní třídou zobrazuje šipkou s trojúhelníkovou hlavou, přičemž tato šipka směřuje ze středu ikony potomka do středu ikony rodiče.

Zobrazování každé z těchto závislostí můžeme nezávisle zapnout či vypnout, takže se nám nestane, že bychom měli složitější diagram počmáraný závislostmi, které při svém množství již nenesou žádnou informaci.

BlueJ dědické závislosti nejenom zobrazuje, ale umožňuje je i jednoduše nastavit nebo zrušit. Po stisknutí tlačítka s dědickou šipkou lze tuto šipku natáhnout od dceřiné třídy k jejímu rodiči, resp. od třídy k implementovanému rozhraní. *BlueJ* pak vzápětí automaticky doplní příslušnou klauzuli do hlavičky závislé třídy nebo rozhraní.

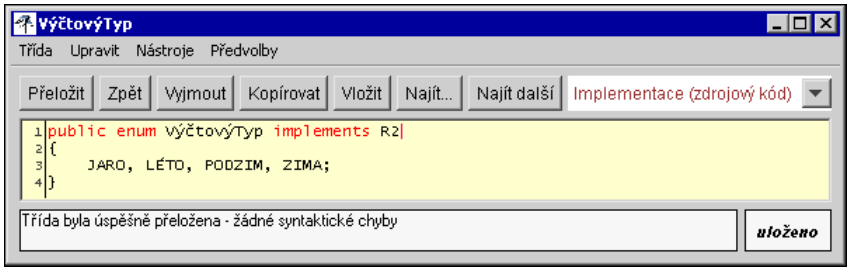
Diagram tříd navíc vždy obsahuje ikonu souboru README.TXT, který můžeme poklepáním na ikonu otevřít a dopsat do něj poznámky k danému projektu a opět jej uložit. Když předáme studentům projekt, aby jej doplnili o požadované vlastní třídy, jej doplňovat dalšími doprovodnými texty, protože všechny potřebné informace naleznou v dokumentací tříd a v souboru README.TXT.

2.3 Editor

Poklepáním na ikonu třídy (případně zadáním příkazu v její místní nabídce) otevřeme nové aplikační okno editoru a v něm zdrojový soubor dané třídy.

Editor je opět maximálně jednoduchý. Vedle běžných operací však nabízí i možnost automatického odsazování obsahu vnořených bloků, příkazy k odsazení a přisazení označeného bloku textu a příkazy k jeho zakomentování a odkomentování.

V konferenci *BlueJ* probíhala v loňském roce velká diskuse o tom, má-li editor nabízet seznam použitelných identifikátorů nebo ne. Autoři obhajovali nepřítomnost on-line nápovědy tím, že studenti si pak nezapamatují správné názvy metod a pouze zkoušejí zadávat vhodně pojmenované metody z nabídnutého seznamu v očekávání, že jedna z nich bude ta pravá.



Obr. 2: Okno editoru

Autoři nicméně poskytují zdrojový kód editoru, takže si jej každý může upravit a přizpůsobit podle svých představ o optimální podobě editoru určeného pro výuku.

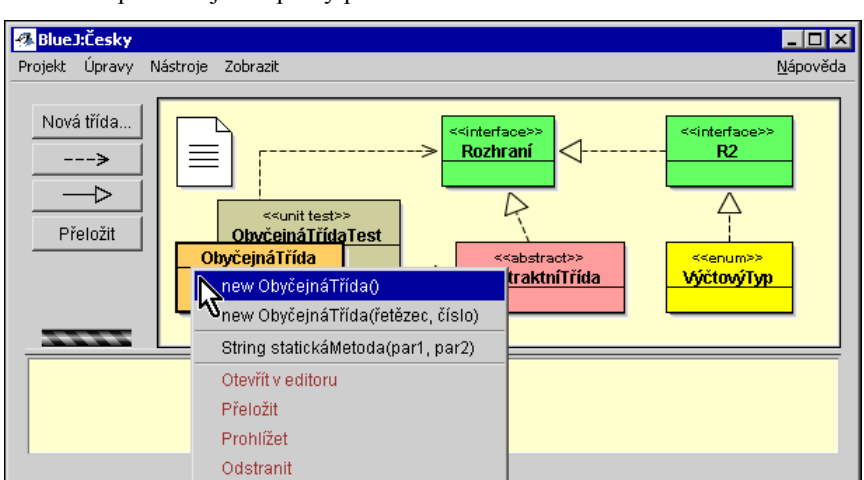
Jednou z důležitých vlastností editoru je, že pracuje ve dvou režimech, které můžeme nastavit zadáním příslušné položky v seznamu zobrazeném na pravém okraji panelu s tlačítky pro nejčastěji používané funkce.

- V režimu *implementace (zdrojový kód)* zobrazuje zdrojový kód třídy, který je možno editovat.
- V režimu *Dokumentace (popis rozhraní)* zobrazuje dokumentaci dané třídy vytvořenou programem *javadoc*.

Nastavený režim si *BlueJ* u každé třídy pamatuje, takže při přípravě projektu pro studenty můžeme nechat všechny třídy, které jsme předpřipravili, zobrazit v dokumentačním režimu, a třídy, které mají studenti za úkol upravit, v režimu implementačním.

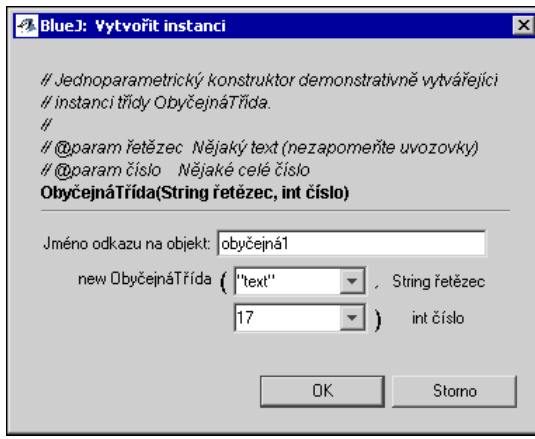
2.4 Třídy a objekty

V horní části místní nabídky třídy je seznam dostupných (tj. nesoukromých) konstruktorů následovaný seznamem dostupných statických metod (viz obr. 3). Tyto příkazy můžeme interpretovat jako zprávy posílané dané třídě.



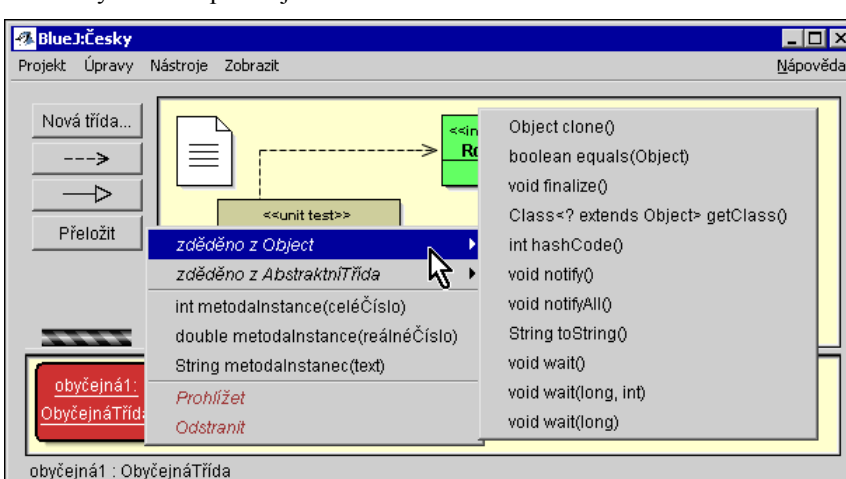
Obr. 3: Místní nabídka třídy

Požádáme-li třídu o vytvoření nové instance, otevře se dialogové okno, v němž po nás *BlueJ* požaduje zadání hodnot parametrů konstruktoru a názvu proměnné, do níž se uloží odkaz na vytvářenou instanci (viz obr. 4). Do horní části dialogového okna opíše dokumentační komentář příslušného konstruktoru následovaný jeho deklarací, ve spodní pak připraví textová pro název proměnné s odkazem na objekt a pro hodnoty jednotlivých parametrů. Vedle každého textového pole navíc napíše, který typ a název parametru, jehož hodnotu do pole máme zapsat.



Obr. 4: Okno pro zadání hodnot parametrů konstruktoru

Po zadání požadovaných hodnot vytvoří *BlueJ* příslušnou instanci a do tzv. *zásobníku odkazů* umístí ikonu představující proměnnou s odkazem na vytvořenou instanci. V místní nabídce této ikony zobrazí příkazy pro zavolání jednotlivých metod instance, přičemž metody zděděné od jednotlivých rodičů mají samostatné podnabídky (viz obr. 5). Prostřednictvím této nabídky mohou uživatelé přímo posílat dané instanci zprávy vedoucí k vyvolání odpovídajících metod.

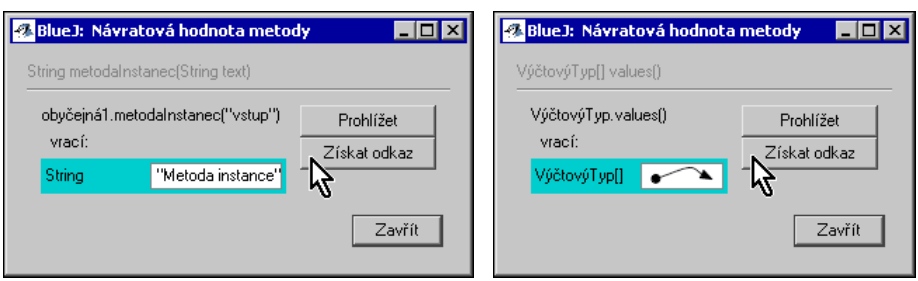


Obr. 5: Místní nabídka odkazu na instanci

Metody vyžadující zadání parametrů otevřou pro jejich zadání obdobné okno jako konstruktory. Parametry objektového typu je přitom možno zadávat buď přímo (např. `java.awt.Color.BLACK`), nebo – je-li odkaz na požadovanou instanci v zásobníku odkazů – klepnutím na příslušný odkaz v zásobníku odkazů. Protože *BlueJ* obsah vstupního textového pole interpretuje, můžeme do něj zadat i příkaz k zavolání konstruktoru či metody, která nám požadovaný odkaz vrátí.

Vrací-li volaná metoda nějakou návratovou hodnotu, otevře *BlueJ* po provedení metody dialogové okno, v němž zobrazí příkaz vyvolávající danou metodu včetně hodnot parametrů a pod ním pak vracenou hodnotu.

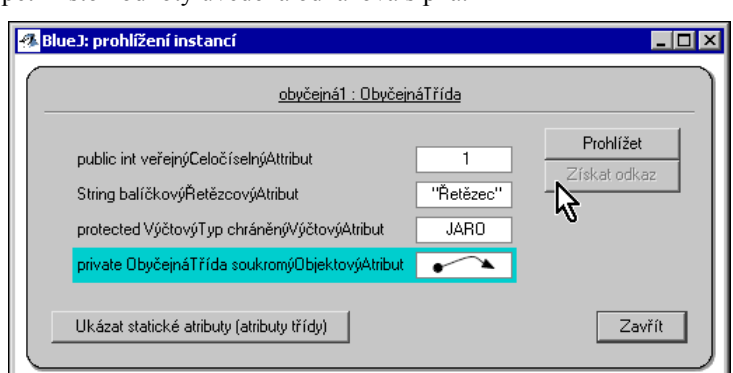
Je-li vracená hodnota objektového typu, zobrazí její hodnotu pouze v případě, jedná-li se o řetězce nebo hodnotu výčtového typu. U hodnot ostatních typů zobrazí *BlueJ* pouze šipku symbolizující, že vracenou hodnotou je odkaz na instanci objektového typu. Nezávisle na typu instance pak pro hodnoty objektových typů nabídne uživateli možnost získat odkaz na jejich hodnotu a umístit jej do zásobníku odkazů (viz obr. 6).



Obr. 6: Dialogové okno s návratovou hodnotou

2.5 Prohlížení tříd a objektů

V místní nabídce ikon v zásobníku odkazů, jakož i v dialogových oknech návratových hodnot objektových typů může uživatel zadat příkaz k prohlížení dané instance. Tím otevře okno prohlížeče, který zobrazí deklarace všech atributů spolu s hodnotami atributů primitivních typů, řetězců a výčtových typů. U atributů ostatních objektových typů je opět místo hodnoty uvedena odkazová šipka.



Obr. 7: Prohlížení instance objektového typu

Instance, na něž odkazují atributy objektových typů, je možné prohlížet, a to nezávisle na jejich přístupových právech. Obsahuje-li instance *veřejný* atribut objektového typu, je možné požádat o umístění příslušného odkazu do zásobníku odkazů. Pro jiné než veřejné atributy je však tato možnost potlačena (viz obr. 7).

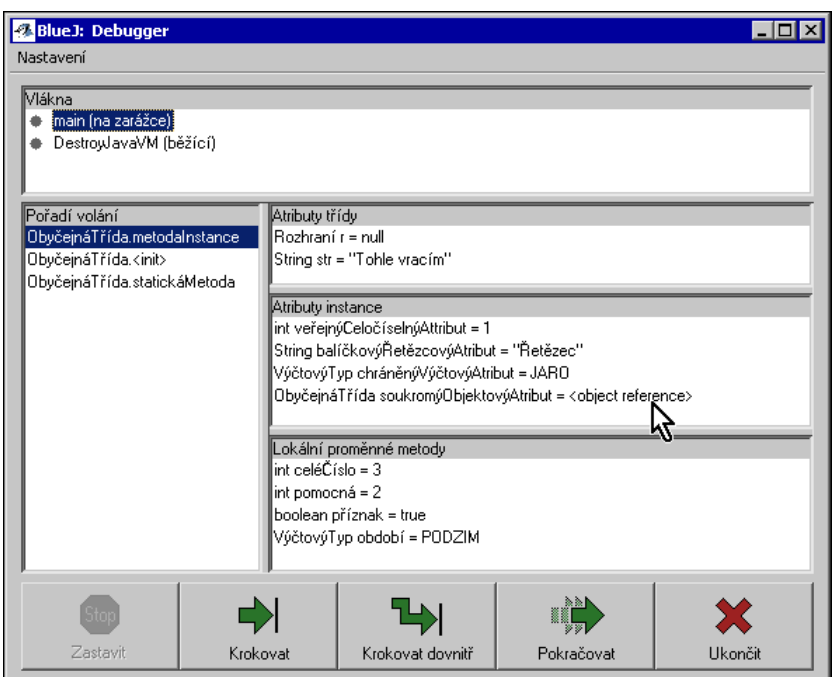
V okně prohlížeče instancí je současně tlačítko umožňující otevřít obdobné okno zobrazující atributy třídy. Toto okno je možno otevřít i zadáním příslušného příkazu v místní nabídce třídy v diagramu tříd.

2.6 Ladění

BlueJ nabízí i poměrně propracované ladicí prostředky. I u nich je opět kladen důraz na názornost a jednoduchost ovládání. *BlueJ* např. neumožňuje některá rafinovaná nastavení zarážek v programu, avšak pro ladění studentských úloh ve vstupních kurzech je vybaven naprosto dostatečně.

BlueJ na rozdíl od většiny profesionálních prostředí nerozlišuje explicitně obyčejné spuštění programu od spuštění v ladicím režimu. Metody tříd a instancí se v něm volají naprosto stejně nezávisle na tom, chceme-li danou metodu pouze spustit nebo ji chceme krokovat.

Do libovolného řádku zdrojového programu můžeme umístit zarážku. K jejímu umístění stačí klepnout v editoru v příslušném řádku do levého sloupce s čísly řádků. *BlueJ* pak na daném místě zobrazí místo čísla řádku malou ikonu připomínající dopravní značku *Stůj, dej přednost v jízdě*.



Obr. 8: Prohlížení instance objektového typu

Narazí-li virtuální stroj při provádění programu na tuto zarážku, otevře *BlueJ* okno debuggeru (viz obr. 8). To obsahuje pět panelů vyhrazených postupně pro spuštěná vlákna, posloupnost volání, atributy třídy, atributy instance a lokální proměnné.

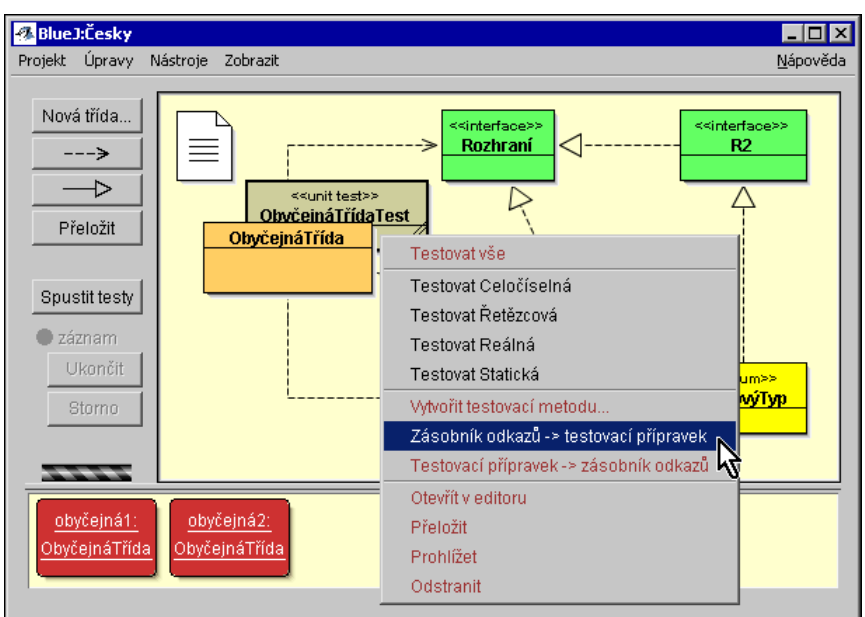
Také debugger zobrazuje pouze hodnoty atributů a proměnných primitivných datových typů, výčtových typů a typu String. U hodnot ostatních objektových typů uvádí pouze informaci o tom, že se jedná o odkaz (viz řádek označený obr. 8 šipkou). Chce-li uživatel získat o příslušné hodnotě podrobnější informace, může na ni poklepat a otevřít tak její prohlížecí okno.

2.7 Testy

Zvláštní pozornost věnovali autoři *BlueJ* maximálnímu usnadnění přípravy a spouštění testů. Vybavili proto *BlueJ* nástroji založenými na knihovně *JUnit* a umožňujícími poloautomatickou přípravu testů podobným předváděcím způsobem, jakým se např. zadávají makra v kancelářských balících.

Chceme-li využívat zabudované testovací nástroje, musíme zaškrtnout příslušné políčko v konfiguračním okně. V levém panelu aplikačního okna se pak objeví tři nová tlačítka určená pro záznam prováděného testu a místní nabídka tříd se rozšíří a příkaz k vytvoření testovací třídy.

Testovací třídu je možno vytvořit i samostatně, ale její vytvoření prostřednictvím místní nabídky testované třídy má dvě drobné výhody: 1. nemusíme vymýšlet název testovací třídy a 2. ikona testovací třídy se v diagramu tříd částečně zasune pod ikonu testované třídy a tyto dvě ikony pak vytvoří „nerozlučnou siamskou dvojici“ usnadňující jejich vyhledání v diagramu tříd.



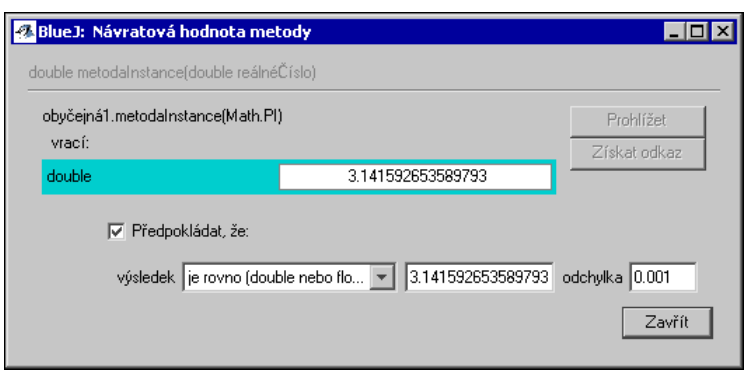
Obr. 9: Místní nabídka testovací třídy

Testy vytvářené za pomoci knihovny *JUnit* jsou koncipované tak, že testovací třída definuje speciální přípravek (test fixture), který se vytvoří před provedením každého testu. *BlueJ* umožňuje vytvořit tento přípravek tak, že připravíme všechny potřebné instance do zásobníku odkazů a pak testovací třídu požádáme, aby obsah zásobníku odkazů uložila jako testovací přípravek (viz obr 9). *BlueJ* pak v testovací třídě deklaruje potřebné atributy a vloží do těla metody `setUp` posloupnost příkazů, které jsme zadali od posledního restartu virtuálního stroje.

Testovací třídu můžeme kdykoliv požádat o nahrání přípravku do zásobníku odkazů a tam s ním buď pracovat nebo jej upravit a uložit v pozměněné podobě.

Vedle příkazů pro práci s přípravkem nabízí místní nabídka testovací třídy příkaz k vytvoření nového testu. Po zadání tohoto příkazu se nás *BlueJ* nejprve zeptá na název testu (zadáme text, který bude v názvu testovací metody následovat předponu `test`), pak naplní zásobník odkazů přípravkem a začne zaznamenávat všechny naše příkazy až do stisku tlačítka **Ukončit**. Po něm definuje v testovací třídě příslušnou metodu `testXxx` a do jejíhož těla vloží zaznamenané příkazy. Stiskneme-li místo něj tlačítko **Storno**, záznam testu zrušíme.

Zavoláme-li při záznamu testu metodu vracející nějakou hodnotu, otevře *BlueJ* dialogové okno, v němž zobrazí vracenou hodnotu a navíc se nás zeptá, má-li tuto hodnotu porovnat s hodnotou námi zadanou (viz obr. 10). Potvrdíme-li požadavek na kontrolu, vloží do vytvářené testovací metody na příslušné místo příkaz `assertEquals`, v němž uvedené hodnoty porovná.

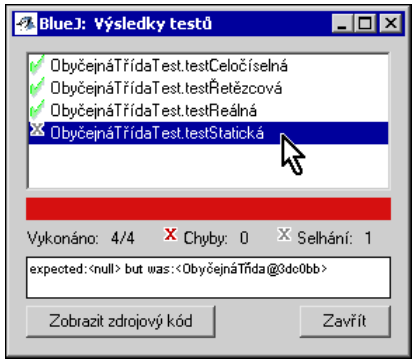


Obr. 10: Nabídka ověření návratové hodnoty v testovací metodě

Výše popsaným způsobem vytvořené testy je možno kdykoliv ručně upravit a doladit. Je ale také možné možnost poloautomatické tvorby testů zcela ignorovat a prostě otevřít testovací třídu v editoru a kód požadovaného testu zapsat z klávesnice.

Z místní nabídky testovací třídy je možné spustit buď konkrétně zadaný test nebo všechny testy v dané testovací třídě. Pro spuštění všech testů ve všech testovacích třídách daného projektu/balíčku slouží tlačítko **Spustit testy** umístěné na levém panelu.

Výsledky testů se zobrazují v okně podobném tomu, jaké známe z balíčku *JUnit* (viz obr. 11). *BlueJ* však toto okno upravuje a umožňuje uživatelům rovnou otevřít editor a zobrazit řádek, který způsobil chybu. To je jistě příjemné, ale mrzí mne, že autoři při přidání tohoto vylepšení zároveň zrušili výpis zásobníku s odkazy na jednotlivá volání vedoucí k dané chybě. Minulé verze jej přitom nabízely.



Obr. 11: Okno zobrazující výsledky provedených testů

2.8 Balíčky a příkazový panel

Prozatím jsem hovořil pouze o projektech, které byly celé v jednom balíčku. *BlueJ* umožňuje i rozdělení celého projektu do několika balíčků a práci s nimi. Diagram tříd každého balíčku se otevírá v samostatném okně.

Automatická aktualizace příkazu package. Sympatickou vlastností *BlueJ* je, že při otevírání balíčku zkontroluje všechny třídy v daném balíčku a v případě potřeby upraví jejich příkazy `package`. To oceníme ve chvíli, kdy připravujeme pro studenty nějaký projekt, který mají doplnit o vlastní třídy, a předpřipravené třídy tohoto projektu máme na svém počítači umístěny v jiném balíčku, než je ten, do nějž je chceme umístit v připravovaném projektu. To se stává často, protože náš strom balíčků bývá složitější a často umísťujeme jednu a tutéž třídu do různých studentských projektů. Ušetříme si tak neustálou kontrolu, jestli jsme deklarace balíčků správně upravili.

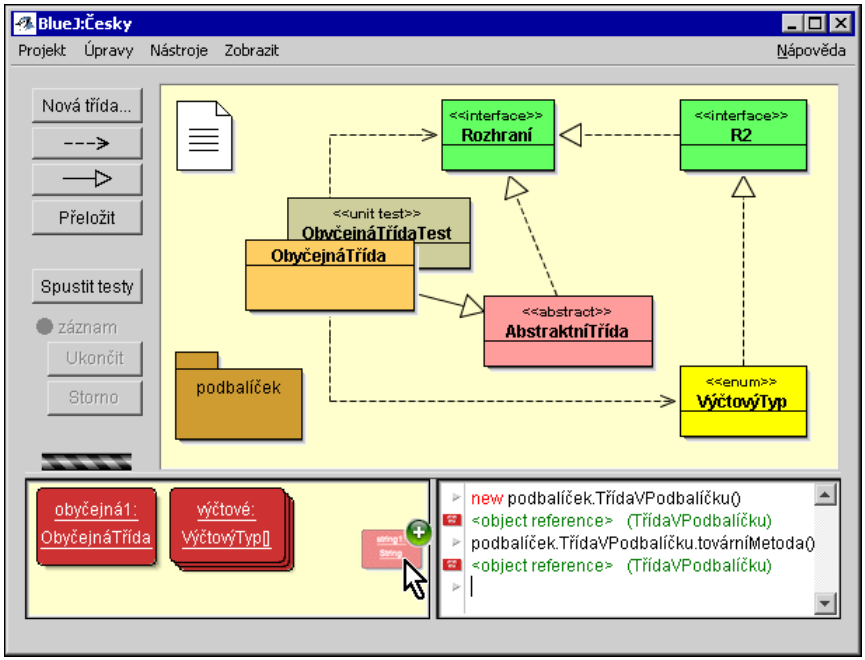
Příkaz `package` za nás *BlueJ* upraví i tehdy, pokud se při práci na nějakém projektu rozhodneme, že do otevřeného balíčku zkopírujeme zdrojový kód třídy, který máme uložen někde jinde.

Nemožnost přímé komunikace mezi okny balíčků. *BlueJ* prozatím neumožňuje současnou práci s diagramy tříd několika balíčků. Není tedy možno natáhnout implementační nebo dědickou šipku od dceřiné třídy v jednom balíčku k rodičovské třídě v jiném balíčku. Při dědění tříd či implementaci rozhraní z jiných balíčků jsme tak omezení na textový zápis do zdrojového souboru.

Není také možné zavolat v jednom okně/balíčku konstruktor či metodu a vrácený odkaz umístit do zásobníku odkazů v jiném okně nebo jej použít k zadání parametru metody vyvolané z místní nabídky třídy či odkazu v jiném okně/balíčku.

Příkazový panel. V poslední verzi autoři vyřešili tento problém tak, že do prostředí přidali příkazový panel, do nějž je možno textově zadávat příkazy jazyka Java. Vyvoláme-li v příkazovém panelu konstruktor či metodu vracející odkaz, zobrazí se vedle textového popisu výsledku malá ikona, kterou je pak možno zkopírovat do místního zásobníku odkazů (viz obr. 12).

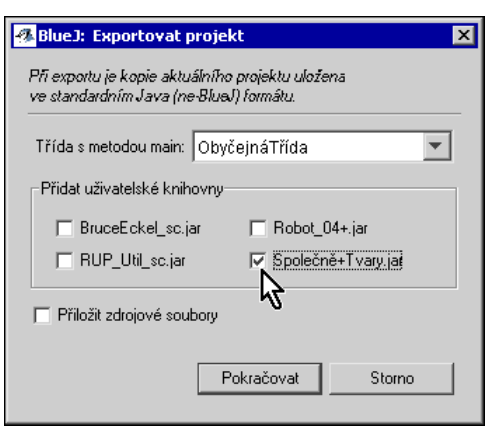
Vzhledem k tomu, že z příkazového panelu můžeme zadáním úplného názvu třídy oslovit třídu v libovolném balíčku včetně standardní knihovny, můžeme tak jednoduše získávat odkazy na instance třídy z libovolného balíčku.



Obr. 12: Odkaz získaný v příkazovém panelu lze zkopírovat do zásobníku odkazů

2.9 Knihovny

BlueJ umožňuje jednoduše vytvářet samostatné aplikace nebo knihovny umístěné v souborech JAR. Stačí zadat příkaz **Soubor → Export**. *BlueJ* pak otevře dialogové okno, v němž se zeptá na případnou hlavní třídu aplikace a na používané knihovny, které se mají do výsledného souboru JAR přibalit (viz obr. 13). Pak už zbývá zadat v dalším okně název a umístění výsledného souboru a tím je vše hotovo.



Obr. 13: Dialogové okno otevřené při exportu do souboru JAR

2.10 Podpora verze 5.0

BlueJ 2.0 plně podporuje verzi J2SE 5.0 (dříve Java 1.5) uvedenou letos na podzim. Umožňuje odlišit v diagramu tříd třídy definující výčtové typy od ostatních typů tříd, zahrnuje do názvů v diagramu tříd plné názvy parametrizovaných datových typů, definuje zvláštní verzi dialogových oken pro volání metod s proměnlivým počtem parametrů. Lze očekávat, že další verze bude schopna využívat i metadat.

2.11 Rozšíření

K používání prostředí *BlueJ* při výuce se na stránkách produktu veřejně hlásí okolo 400 univerzit a školicích středisek po celém světě. Mnozí z vyučujících mají své vlastní představy o tom, co by chtěli na prostředí vylepšit nebo co by chtěli přidat. Autoři *BlueJ* proto zhruba před rokem přišli se speciální knihovnou umožňující rozšíření prostředí o nejrůznější funkce. V současné době jsou na stránkách *BlueJ* k dispozici následující tři rozšíření:

Submitter. Toto rozšíření je součástí standardní instalační sady. Umožňuje zaslání projektu přímo z prostředí *BlueJ*, přičemž je možno používat protokoly ftp, http, mailto a file. Prostřednictvím tohoto rozšíření zasílají studenti výsledky svých prací. *Submitter* lze široce konfigurovat pomocí konfiguračního souboru.

CheckStyle. Umožňuje kontrolovat dodržování formátovacích konvencí, které je možno opět podrobně specifikovat v konfiguračním souboru..

Sequence Diagram Editor. Editor umožňující vytvoření sekvenčních diagramů a jejich propojení s *BlueJ*.

V konferenci k produktu se množí ohlášení závěrečných fází vývoje dalších rozšíření. Lze proto očekávat, že v průběhu tohoto roku počet dostupných rozšířen významně naroste.

3 Závěr

Prostředí *BlueJ* je nástrojem, který umožňuje zefektivnit výuku programování v kurzech, jež pracují se začátečníky nebo s programátory, kteří přecházejí na objektově orientované programování z jiných metodik. Díky své interaktivnosti umožňuje studentům rychleji pochopit mnohé základní pojmy objektového programování a daleko dříve a hlavně daleko lépe vstřebat objektové paradigma.

Ve firmě Amaio Technologies používáme toto prostředí v našich kurzech jazyka Java. Jeho použití se ukazuje jako velmi výhodné nejenom při výuce naprostých začátečníků, kteří se rozhodli vstoupit do světa programování prostřednictvím jazyka Java, ale i při doškolování a přeškolování profesionálních programátorů, kteří mají bohaté zkušenosti ze strukturovaného programování v klasických jazycích typu Pascal či C++, anebo z používání nejrůznějších skriptovacích jazyků, povětšinou PHP. Standardní vývojová prostředí používáme až v pokročilých kurzech určených pro programátory, kteří již mají objektovou technologii zažitou a zajímají se o některé pokročilejší vlastnosti jazyka Java.

Reference

1. Webová stránka prostředí BlueJ: *www.bluej.org*
2. Kölling, M.: The BlueJ Tutorial Version 2.0.1 for BlueJ Version 2.0.x. Online book at *http://www.bluej.org/tutorial/tutorial-201.pdf*.
3. Pecinovský, R.: *Myslíme objektově v jazyku Java 5.0*, Grada, 2004.
4. Van Haaster, K. and Hagan, D.: Teaching and Learning with BlueJ: an Evaluation of a Pedagogical Tool. *Information Science + Information Technology Education Joint Conference*. Rockhampton, QLD, Australia, June 2004.
5. Kölling, M., Quig, B., Patterson, A. and Rosenberg, J.: The BlueJ system and its pedagogy. *Journal of Computer Science Education, Special issue on Learning and Teaching Object Technology, Vol 13, No 4*. Dec 2003.
6. Patterson, A., Kölling, M. and Rosenberg, J.: Introducing Unit Testing With BlueJ. *Proceedings of the 8th conference on Information Technology in Computer Science Education (ITiCSE 2003)*. Thessaloniki, 2003.
7. Nourie, D.: Teaching Java Technology With BlueJ. Online article at *http://java.sun.com/features/2002/07/bluej.html*, July 2002.
8. Kölling, M.: Teaching Object Orientation with the Blue Environment. *Journal of Object-Oriented Programming, Vol. 12 No. 2, 14-23*. 1999.
9. Kölling, M.: *The Design of an Object-Oriented Environment and Language for Teaching*. PhD Thesis, Basser Department of Computer Science, University of Sydney, 1999.

Annotation

BlueJ – IDE for Teaching Java

The paper introduces the integrated development environment *BlueJ*, which was developed especially for introduction courses of programming using Java language. *BlueJ* let students design class diagrams of the developed application in the simplified version of UML language. From these diagrams it automatically constructs skeletons of used classes and reversely it projects all changes in the source code back in the class diagrams. The main advantage of the *BlueJ* environment is its interactivity. It allows sending messages to particular classes and instances. Further more it adds many other features, which are useful for teaching object oriented programming – e.g. interactive test generation. The paper presents the key features of this IDE and it shows, which advantages brings usage of this environment in the teaching process.