The BlueJ Tutorial

verze 1.4
pro BlueJ verze 1.2.x

Michael Kölling
Mærsk Institute
University of Southern Denmark

přeložil Petr Škoda, 1. verze

1 Předmluva

1.1 O BlueJ

Tento tutorial slouží jako úvod k používání programovacího prostředí BlueJ. BlueJ je vývojové prostředí pro jazyk Java® vytvořené speciálně pro potřeby výuky základů objektově orientovaného programování. Toto prostředí bylo navrženo a vytvořeno společným týmem z Monash University v Melbourne, Australia a z University of Southern Denmark v Odense.

Další informace jsou dostupné na http://www.bluej.org.

1.2 Rozsah a cílová skupina

Tento tutorial byl vytvořen pro ty, kteří se chtějí seznámit s používáním prostředí BlueJ. Není zde popsán systémový návrh a ani výzkum, na kterém byl celý systém založen.

Tento text není zaměřen na výuku jazyka Java. Začátečníci by si měli nejdříve prostudovat nějakou jinou příručku o základech Javy.

Na začátku každého oddílu je krátká charakteristika jeho obsahu. Uživatelé s předchozí zkušeností s BlueJ se tak mohou snáze rozhodnout, zda daný oddíl přeskočit. Kapitola 10 obsahuje pouze stručné shrnutí referenčního charakteru.

Kapitola 11 není součástí originálního anglického textu, jsou zde uvedeny poznámky k češtině v BlueJ a Javě obecně.

1.3 Autorská práva, licencování a redistribuce

1.4 Zpětná vazba

Poznámky, otázky, návrhy oprav, kritika a další formy zpětné vazby týkající se systému BlueJ a tohoto tutorialu jsou velice vítány. Maily zasílejte prosím na adresu Michaela Köllinga (mik@mip.sdu.dk).

2 Instalace

BlueJ je distribuován ve třech instalačních formátech: pro Windows, MacOS a pro ostatní operační systémy s podporou Javy. Proces instalace je relativně přímý a jednoduchý.

Předpoklady instalace

Musíte mít nainstalováno J2SE v1.3 (JDK 1.3) nebo novější. Jestliže nemáte JDK, můžete ho získat na adrese http://java.sun.com/j2se/. V MacOS je aktuální verze již předinstalována – nemusíte nic instalovat. Existují dvě verze Javy: „JRE“ (Java Runtime Environment) – pouze pro spouštění aplikací a „SDK“ (Software Development Kit) – pro vývoj aplikací. Pro používání BlueJ potřebujete SDK!

2.1 Instalace pro Windows

Instalační program pro Windows se jmenuje *bluejsetup-xxx.exe*, kde *xxx* značí číslo verze. Například instalační program BlueJ verze 1.2.0 se nazývá *bluejsetup-120.exe*. Tento soubor můžete získat na CD nebo na stránkách BlueJ http://www.bluej.org.

Spusťte tento instalační soubor a vyberte adresář pro instalaci. Poté budete dotázáni, zda si přejete vytvořit zástupce v nabídce Start a na pracovní ploše.

Pro spouštění můžete použít vytvořené zástupce nebo přímo soubor *bluej.exe* v adresáři aplikace BlueJ.

Při prvním spuštění BlueJ jsou vyhledány nainstalované verze JDK. Jestliže je nalezen více než jeden vhodný systém Java SDK (např. současná instalace JDK 1.3.1 a JDK 1.4), zobrazí se dialog s možností výběru. Jestliže není nalezen žádný vhodný systém, budete dotázáni na přesné umístění Java SDK (to v případě, jestliže byly odstraněny záznamy z registru Windows).

Při instalaci je do adresáře BlueJ umístěn rovněž program *wmselect.exe*. Pomocí tohoto programu můžete později změnit umístění používaného JDK. Spusťte *wmselect.exe* a vyberte jinou verzi.

Volba používaného JDK je uložena pro každou verzi BlueJ zvlášť. Změna pomocí *wmselect.exe* se projeví ve všech shodných verzích BlueJ.

2.2 Instalace pro Macintosh

BlueJ pracuje pouze s MacOS X.

Instalační soubor pro MacOS se jmenuje *BlueJ-xxx.sit*, kde *xxx* značí číslo verze. Například instalační soubor verze 1.2.0 se jmenuje *BlueJ-120.sit*. Tento soubor můžete získat na CD nebo na stránkách BlueJ http://www.bluej.org.

Tento soubor může být rozbalen pomocí *StuffIt Expanderu*. Většina prohlížečů tento soubor rozbalí za vás. V opačném případě použijte dvojkliknutí.

Při dekompresi je vytvořena složka BlueJ-xxx. Tuto složku můžete přesunout do složky *Application* (nebo kamkoliv jinam). Žádné další kroky nejsou zapotřebí.

2.3 Instalace pro Linux/Unix a ostatní systémy

Instalační soubor je spustitelný jar archív pojmenovaný b*luej-xxx.jar*, kde *xxx* značí číslo verze. Například instalační soubor verze 1.2.0 se jmenuje b*luej-120.jar*. Tento soubor můžete získat na CD nebo na stránkách BlueJ http://www.bluej.org.

Instalaci spustíte pomocí následujícího příkazu:

```
<jdk-path>/bin/java –jar bluej-120.jar
```

`<jdk-path>` je adresář, kde je JDK nainstalován.

Na obrazovce se objeví dotaz na cílový adresář pro instalaci BlueJ a dotaz na umístění JDK. Cesta k instalačnímu adresáři nesmí obsahovat mezery.

Klikněte na tlačítko *Install* a instalace by měla proběhnout bez dalších problémů.

2.4 Problémy při instalaci

Jestliže narazíte na nějaké problémy při instalaci, přečtěte si prosím *Frequently Asked Questions* (FAQ) na stránkách BlueJ (http://www.bluej.org/help/faq.html) a případně také *How To Ask For Help* (http://www.bluej.org/help/ask-help.html).

3 Začínáme – editace/kompilování/volání metod

3.1 Spuštění BlueJ

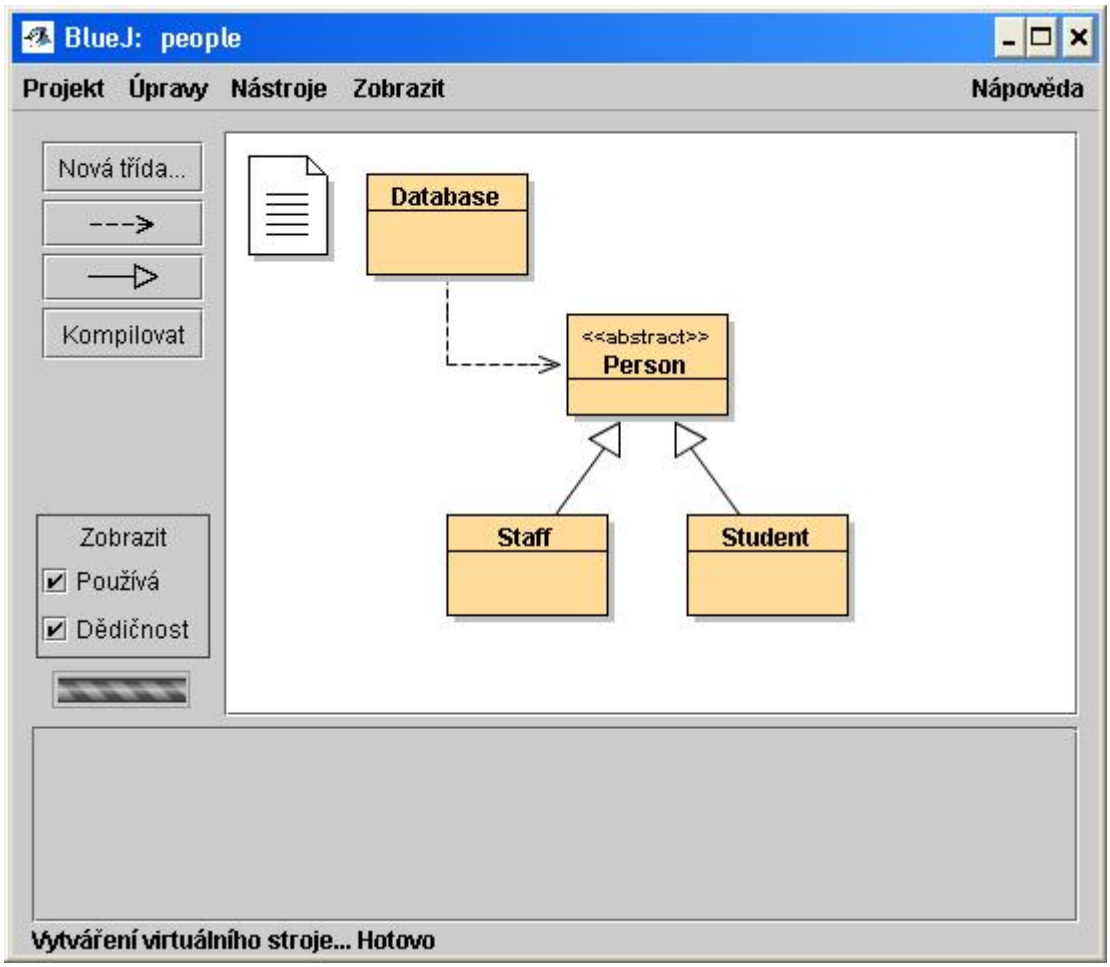
Ve Windows a MacOS je při instalaci vytvořena ikona aplikace *BlueJ*. Spusťte tuto aplikaci.

V Unixu je při instalaci vytvořen v instalačním adresáři spouštěcí skript *bluej*. Pod grafickým rozhraním spustíte BlueJ dvojkliknutím na skript *bluej*. Z příkazové řádky nastartujete BlueJ pomocí následujícího příkazu, jako parametr můžete zadat jméno projektu:

```
$ bluej
```

nebo

```
$ bluej examples/people
```



Obrázek 1: Hlavní okno aplikace BlueJ

3.2 Otevření projektu

Stručně: Jestliže chcete otevřít projekt, vyberte příkaz Otevřít... *z menu* Projekt.

Projekty BlueJ jsou, stejně jako normální Java projekty, tvořeny složkami obsahujícími soubory.

Pro otevření projektu použijte příkaz *Otevřít...* z menu *Projekt.*

Součástí distribuce BlueJ je několik ukázkových projektů, tyto projekty jsou uloženy ve složce *examples.*

V této části tutorialu budeme pracovat s projektem *people,* který se nachází v *...\BlueJ\examples.* Poté, co úspěšně otevřete tento projekt, budete mít před sebou podobné okno jako na obrázku 1.

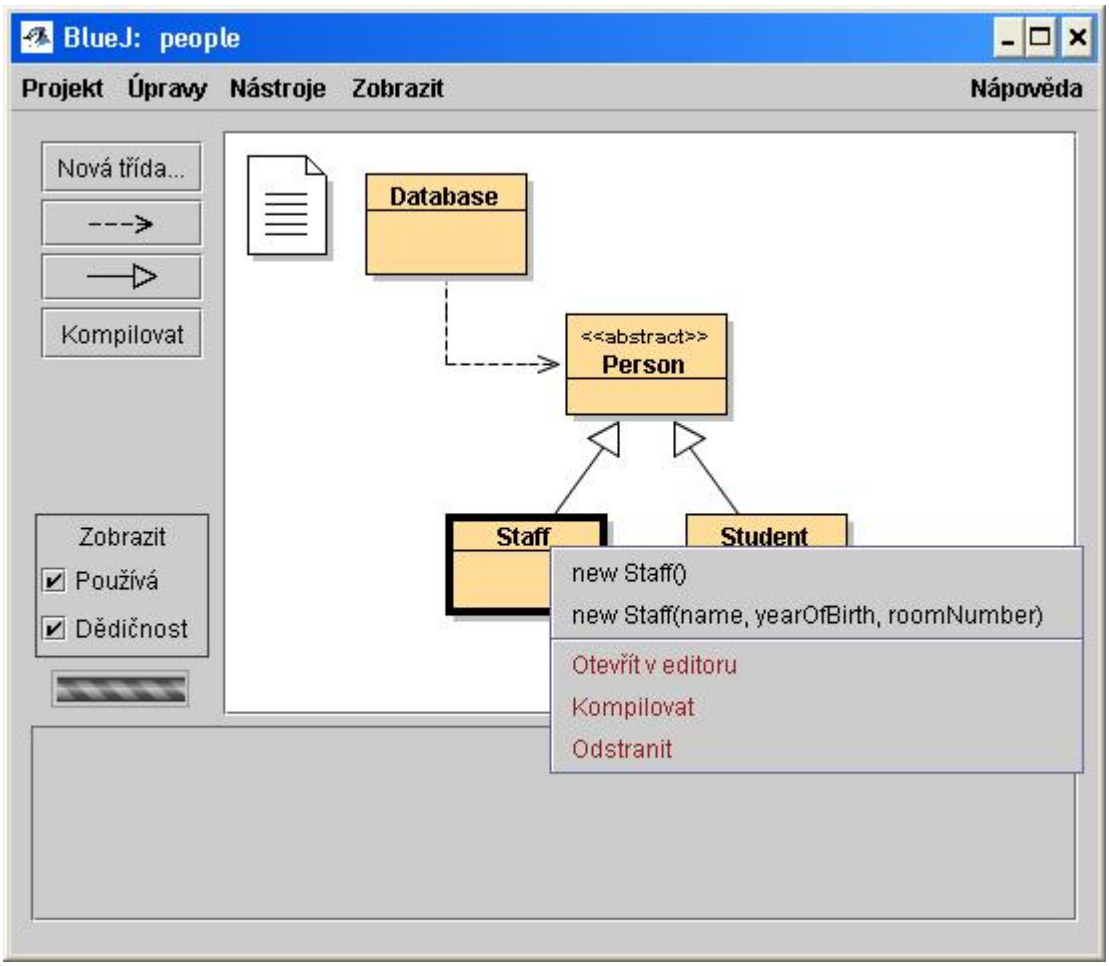
3.3 Vytváření instancí objektů

Stručně: Jestliže chcete vytvořit novou instanci objektu, vyberte konstruktor z kontextového menu třídy.

Jednou z hlavních předností systému BlueJ je to, že nemusíte pouze spouštět celou aplikaci, ale můžete rovněž přímo komunikovat s jednotlivými třídami a instancemi a volat jejich jednotlivé metody. Spouštění je obvykle prováděno tak, že je vytvořena instance objektu a poté jsou volány její metody. Toto je velký přínos pro vývoj a testování aplikací. Můžete testovat nové třídy a jejich metody okamžitě po jejich vytvoření. Není tedy nutné nejdříve napsat kompletní aplikaci.

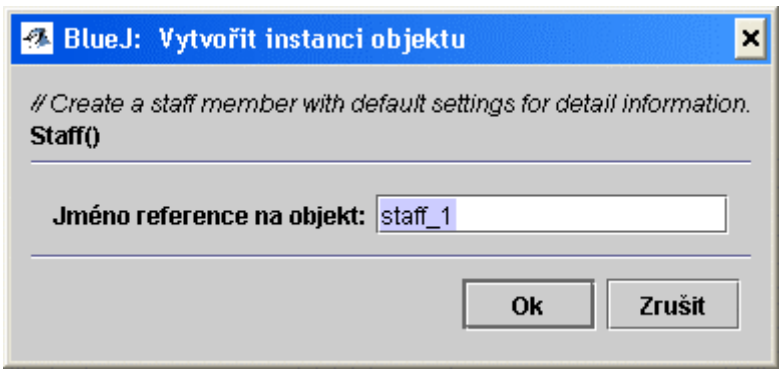
> Poznámka: *Statické metody mohou být volány přímo bez vytvoření instance objektu. Jednou ze statických metod je „main“, můžete tedy normálně spustit aplikaci pomocí volání této statické metody. K tématu se vrátíme ještě později. Nejdříve se ale podíváme na další možnosti, které nejsou běžné v ostatních vývojových prostředích.*

Obdélníky, které vidíte v centrální části hlavního okna (označené jako *Database, Person, Staff* a *Student*), jsou ikony reprezentující jednotlivé třídy aplikace. Každá třída má kontextové menu, které zobrazíte tak, že na ikonu kliknete pravým tlačítkem. Na obrázku je zobrazen příkaz *new* pro volání jednotlivých konstruktorů spolu s dalšími příkazy.



Obrázek 2: Příkazy – třída

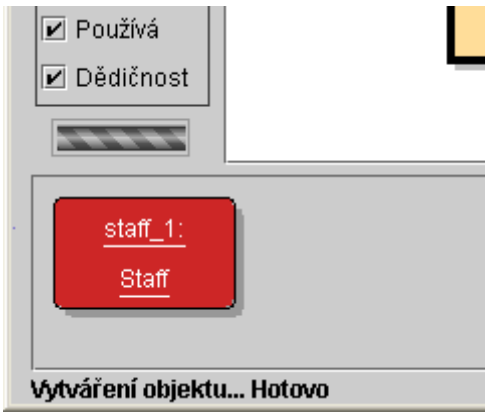
Jestliže si přejete vytvořit instanci třídy *Staff*, klikněte pravým tlačítkem na ikonu *Staff* (objeví se kontextové menu jako na obrázku 2). Menu obsahuje dva konstruktory pro vytvoření instance třídy *Staff*, jeden s parametry a druhý bez parametrů. Vyberte konstruktor bez parametrů, zobrazí se následující dialog.



Obrázek 3: Vytvoření instance objektu bez parametrů

V tomto dialogu jste dotázáni na jméno reference na nově vytvořený objekt. V dialogu je navrženo implicitní jméno vytvořené z názvu třídy. Pro naše potřeby je navržené jméno dostačující, klikněte na *OK* a nová instance objektu bude vytvořena.

Po vytvoření nové instance je reference uložena do seznamu referencí.
To je vše: vyberte třídu, zvolte konstruktor a pojmenujte referenci.



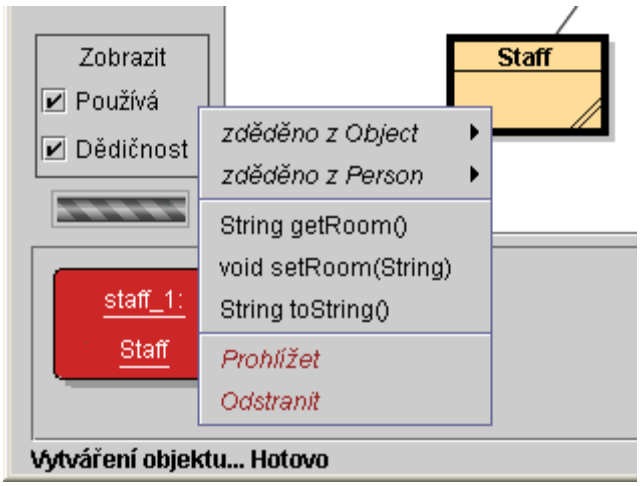
Obrázek 4: Zásobník referencí

U třídy *Person* si všimněte označení <<abstract>>, které značí, že třída je abstraktní. Abstraktní třídu nemůžete použít pro vytvoření nové instance, vyzkoušejte si to (v kontextovém menu nejsou uvedeny žádné konstruktory).

3.4 Volání metod

Stručně: Jestliže chcete zavolat metodu, vyberte ji z kontextového menu třídy nebo reference.

Vytvořili jste instanci objektu, nyní zkuste zavolat jednu z jeho veřejně přístupných (public) metod. Pomocí kliknutí pravým tlačítkem zobrazíte kontextové menu reference, viz obrázek 5. V tomto menu jsou uvedeny všechny veřejné metody spolu s příkazy *Prohlédnout* a *Odstranit* (o těchto příkazech budeme mluvit později).



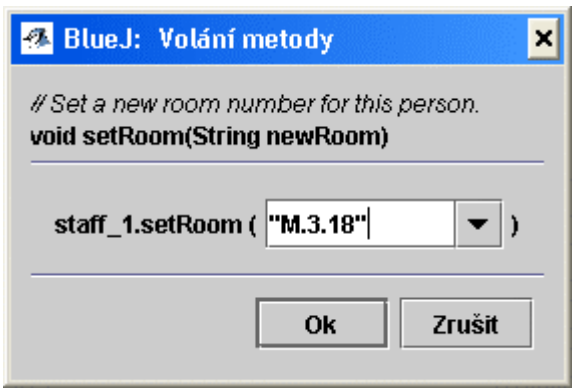
Obrázek 5: Kontextové menu reference

Metody *getRoom* a *setRoom* lze použít pro získání a nastavení jména kabinetu daného učitele. Nyní zkuste zavolat metodu getRoom. Výstup volání je na obrázku 6, hodnota je "(unknown room)" - to proto, že jméno kabinetu zatím nebylo zadáno.

Obrázek 6: Zobrazení výstupu metody getName()

Metody zděděné z nadtřídy jsou zobrazeny v podmenu. V horní části menu na obrázku 5 jsou dvě podmenu, první obsahuje metody zděděné ze třídy *Object* a druhé metody zděděné ze třídy *Person*. Nyní zkuste zavolat metodu *getName* zděděnou ze třídy *Person*. Výsledná hodnota je "(unknown name)", protože učitel nebyl zatím pojmenován.

Nyní přidělíme učiteli jeho kabinet. Metody *getRoom* a *getName* vracely výstupní hodnotu, ale neměli žádné vstupní parametry. Nyní si zkusíme zavolat metodu *setRoom* s jedním vstupním parametrem. Opět použijte kontextové menu reference a zavolejte metodu *getRoom*. Na obrazovce se zobrazí dialog s dotazem na vstupní parametry (viz obrázek 7).



Obrázek 7: Vstupní parametry volání metody

V horní části dialogu je komentář a hlavička metody. Uprostřed jsou textová pole pro zadávání parametrů. Z hlavičky je patrné, že metoda očekává jeden parametr typu *String*. Zadejte proto do textového pole jméno kabinetu spolu s uvozovkami a klikněte na *OK*.

To je vše – metoda nevrací žádnou hodnotu, proto nebyl zobrazen žádný další dialog. Zavolejte znovu metodu *getRoom* a zkontrolujte, zda byla změna provedena.

Nyní si zkuste vytvářet další instance a volat různé metody. Zkuste také konstruktory s parametry. Nepokračujte dál, dokud se důkladně neseznámíte s vytvářením instancí a voláním metod.

3.5 Editování třídy

Stručně: Jestliže chcete editovat zdrojový kód třídy, dvojklikněte na ikonu třídy.

Doposud jsme se zabývali pouze vnějším rozhraním tříd. Nyní je čas podívat se, jak vypadají třídy uvnitř. Implementaci třídy zobrazíte pomocí dvojkliknutí na ikonu třídy, nebo pomocí příkazu *Otevřít v editoru* z kontextového menu třídy. Detailní popis editoru není součástí tohoto tutorialu. Některé jeho součásti budou popsány později.

Nyní si otevřete zdrojový kód třídy *Staff*. Najděte metodu *getRoom* a změňte tělo tak, aby metoda vracela jméno kabinetu spolu s prefixem „room“. Docílíte toho jednoduše tak, že místo

```
return room;
```

napíšete

```
return "room" + room;
```

BlueJ používá pouze „čistou“ Javu, při psaní zdrojového kódu třídy nejsou dána žádná podstatná omezení.

3.6 Kompilace

Stručně: Jestliže chcete zkompilovat třídu, klikněte v editoru na tlačítko Kompilovat. *Jestliže chcete zkompilovat celý projekt, klikněte na tlačítko* Kompilovat *v projektovém manažeru.*

Jestliže se ihned po provedení změn podíváte do okna projektového manažeru, zjistíte, že upravená třída je vyšrafována. Ikona třídy je vyšrafována tehdy, když není zkompilována (tj. aktuální soubor *.class není k dispozici pro spouštění).

> Poznámka: *Možná se teď divíte, jak to, že při prvním otevření projektu* people *byly už všechny třídy zkompilovány. Tento projekt je v distribuci už zkompilován, ostatní projekty jsou většinou distribuovány nezkompilované.*

Na nástrojové liště editoru jsou tlačítka s nejčastěji používanými funkcemi. Jedna z nich je *Kompilovat*. Toto tlačítko vám umožní kompilovat aktuálně otevřenou třídu přímo z editoru. Nyní klikněte na tlačítko *Kompilovat*. Jestliže jste neudělali žádnou chybu, objeví se v dolní části editoru hláška o úspěšné kompilaci. V opačném případě se v dolní části zobrazí chybové hlášení.

Jestliže jste úspěšně zkompilovali změněnou třídu, uzavřete editor.

> Poznámka: *Změny nemusíte ukládat na disk. Zdrojový kód je vždy automaticky uložen před kompilací a při zavření editoru. Manuálně můžete změny uložit pomocí příkazu* Uložit *v menu* Třída.

V nástrojové liště projektového manažeru (tj. hlavní okno) je rovněž tlačítko *Kompilovat*. Pomocí tohoto příkazu zkompiluje celý projekt. (U každé třídy je zjištěno, zda je třeba ji zkompilovat, a poté jsou třídy zkompilovány ve správném pořadí.) Zkuste si to, změňte několik tříd a klikněte na tlačítko *Kompilovat*. Jestliže je při kompilaci nalezena chyba, otevře se editor a zobrazí se odpovídající chybová hláška.

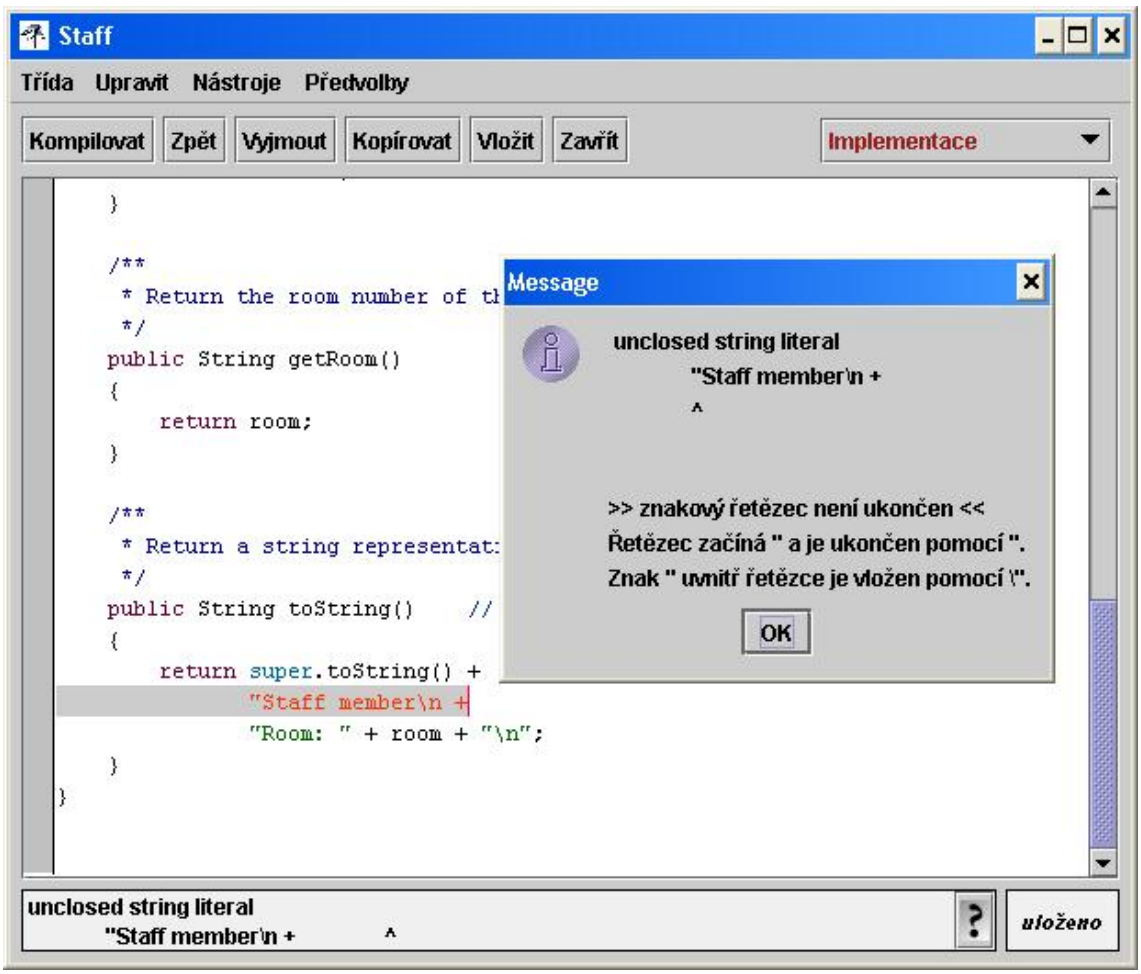
Patrně jste si všimli, že zásobník referencí je opět prázdný. Reference jsou odstraněny vždy, když je změněna implementace libovolné třídy v projektu (tj. otevřete editor a uděláte libovolnou změnu ve zdrojovém kódu).

3.7 Nápověda pro chyby kompilace

Stručně: Jestliže si přejete zobrazit podrobnější nápovědu k chybě kompilátoru, klikněte na tlačítko s otazníkem.

Začátečníci mají velmi často problémy s pochopením standardních chybových hlášek kompilátoru. Snažili jsme se proto poskytnout podrobnější nápovědu.

Otevřete znovu editor, udělejte nějakou syntaktickou chybu a zkompilujte třídu.
V informační oblasti editoru by měla být zobrazena chybová hláška. Nyní klikněte na tlačítko s otazníkem (vpravo dole).



Obrázek 8: Detailní nápověda k chybové hlášce kompilátoru

V této fázi zatím neexistuje detailní nápověda pro všechny chybové hlášky a některé nejsou úplně korektní. V každé další verzi BlueJ je tato nápověda rozšiřována.

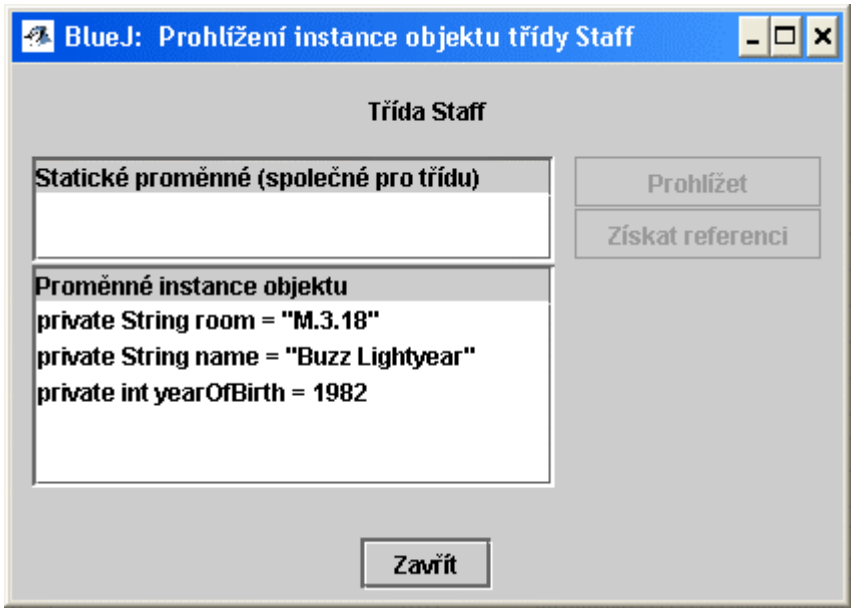
4 Něco navíc...

V této kapitole se budeme zabývat několika dalšími možnostmi vývojového prostředí BlueJ. Nejsou sice nezbytné, ale přesto bývají využívány velmi často.

4.1 Prohlížení

Stručně: Prohlížení objektu slouží, podobně jako debugging, k zobrazení vnitřního stavu instance objektu.

Při volaní metod jste se již setkali s příkazem *Prohlížet* (viz obrázek 5). Tento příkaz umožňuje prohlížení stavu vnitřních proměnných instance objektu. Zkuste vytvořit novou instanci třídy *Staff* pomocí konstruktoru se vstupními parametry. Potom vyberte příkaz *Prohlížet* z kontextového menu nové reference. Zobrazí se dialog obsahující typ a hodnoty členských proměnných (viz obrázek 9).

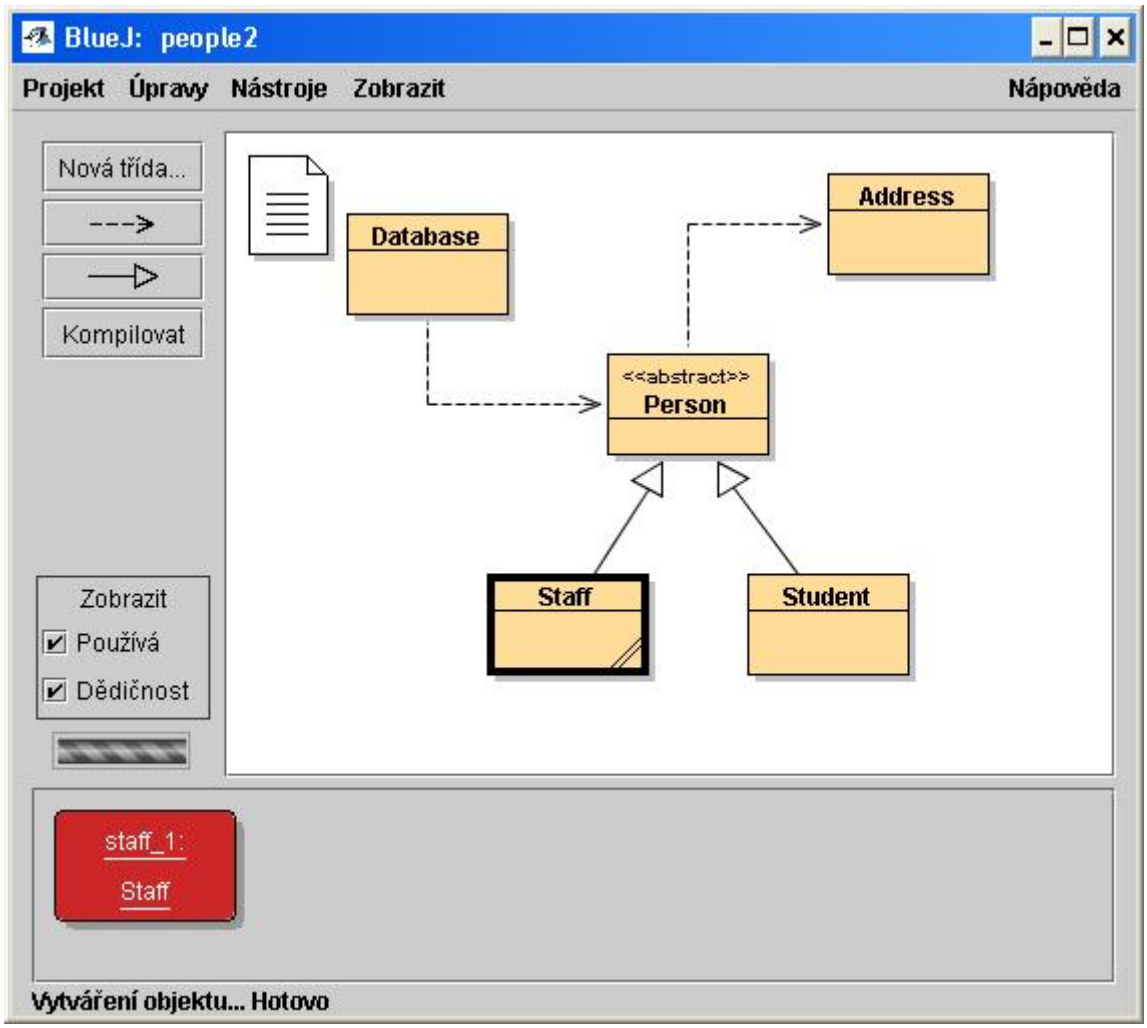


Obrázek 9: Prohlížení instance

Prohlížení je vhodné pro zjištění, zda při předchozím volání metody proběhla požadovaná změna vnitřního stavu instance. Prohlížení je tedy jednoduchý nástroj pro debugging.

V našem příkladu má třída *Staff* pouze vnitřní proměnné jednoduchého typu (tj. primitivní typy a řetězce znaků). Můžete tedy rychle a snadno zjistit, zda jste zadali správné parametry konstruktoru.

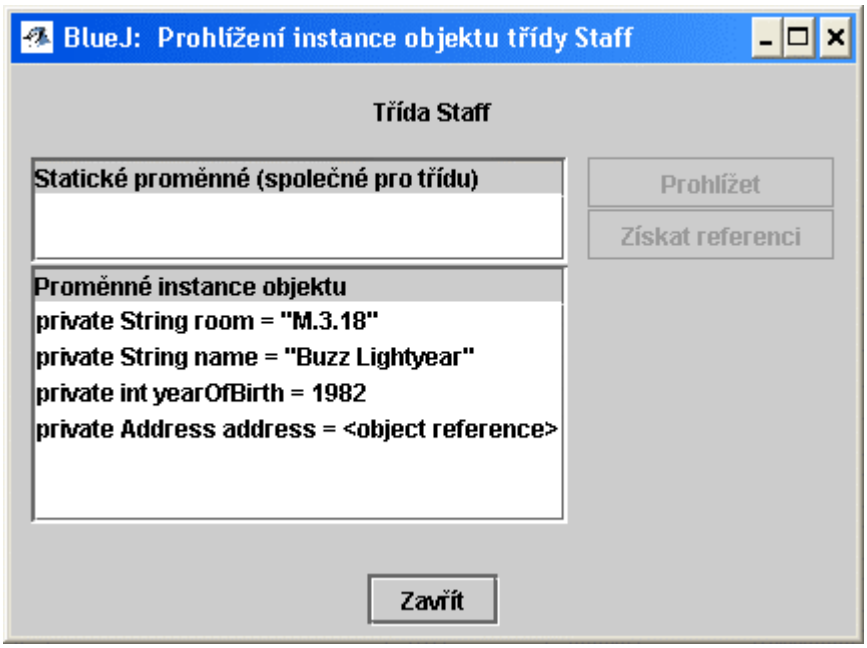
V případě komplexnějších tříd mohou proměnné obsahovat odkazy na jiné instance objektů. Podívejme se nyní na tento případ, otevřete projekt *people2*, který je rovněž součástí standardní distribuce BlueJ (viz obrázek 10). Jak vidíte, projekt obsahuje novou třídu *Adress*. Jedna z proměnných třídy *Person* obsahuje odkaz na uživatelský typ *Adress*.



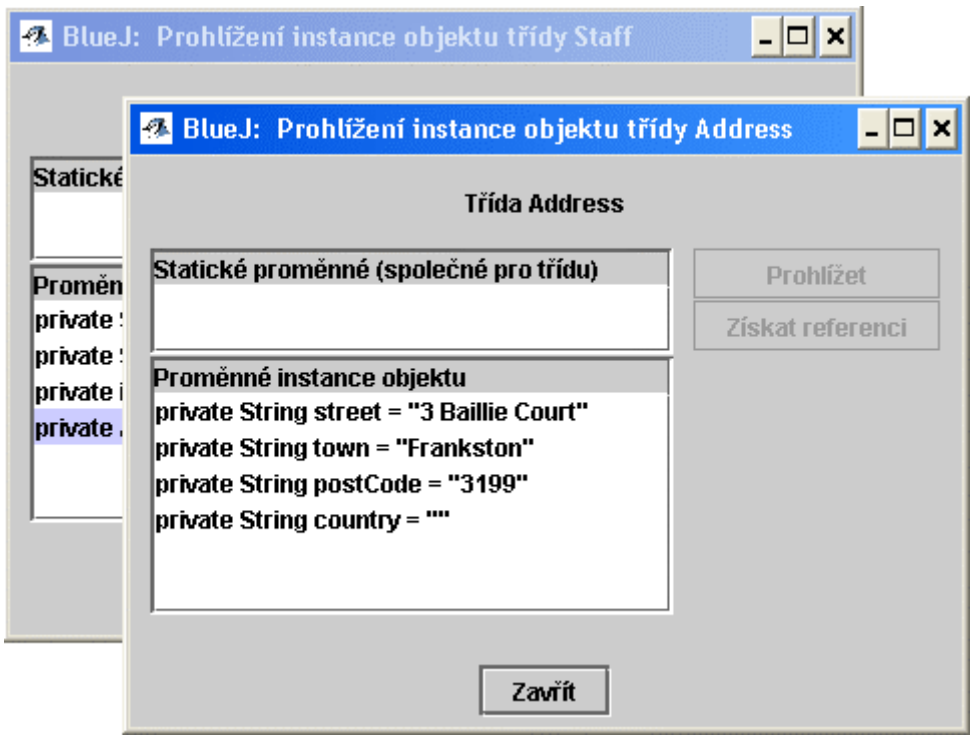
Obrázek 10: Projektové okno *people2*

Pro potřeby dalšího výkladu – prohlížení vnitřních proměnných – vytvořte instanci objektu třídy *Staff*, potom zavolejte metodu *setAddress* (najdete ji v podmenu *zděděno z Person*). Zadejte adresu učitele, metoda vytvoří instanci třídy *Address* a uloží odkaz do vnitřní proměnné *address* instance *Staff*.

Nyní si prohlédněte vnitřní stav nové reference *staff_1*. Dialog s prohlížením je na obrázku 11. Jedna z proměnných instance třídy *Staff* se jmenuje *adress*. Jak můžete vidět, zobrazená hodnota je <*object reference*>. Protože se jedná o komplexní objekt, nemůže být jeho hodnota zobrazena přímo. Jestliže chcete získat více informací o hodnotě této proměnné, označte ji pomocí kliknutí myši a zmáčkněte tlačítko *Prohlížet*. Automaticky se otevře další okno s podrobnými informacemi o této instanci třídy *Address* (viz obrázek 12).



Obrázek 11: Prohlížení instance



Obrázek 12: Prohlížení vložené instance

4.2 Předávání reference jako vstupní parametr metody

Stručně: Instance objektu může být předána jako vstupní parametr metody pomocí kliknutí na ikonu reference.

Reference na instanci objektu může být předána jako parametr volání metody jiného objektu. Například: vytvořte instanci *Database*. Objekt *Database* slouží k uložení seznamu osob. Tento objekt má dvě metody, první pro přidání osoby do seznamu a druhou, která vypíše všechny osoby uvedené v seznamu. (Ve skutečnosti by objekt s názvem databáze obsahoval asi mnohem více užitečných metod.)

Jestliže nemáte vytvořenou žádnou instanci třídy *Staff* nebo *Student*, vytvořte je nyní. Pro následující příklad potřebuje mít v zásobníku referencí alespoň jednu instanci *Database* a několik instancí *Student* nebo *Staff*.

Nyní zavolejte metodu *addPerson* z kontextového menu reference typu *Database*. Z hlavičky metody se dozvíte, že metoda očekává jeden parametr typu *Person*. (Třída *Person* je abstraktní, proto nemůžete vytvořit instanci třídy *Person* pomocí konstruktoru. Třídy *Student* a *Staff* jsou odvozeny od třídy *Person*, to znamená, že zdědily automaticky rozhraní třídy *Person* a můžou být tedy použity na místě, kde je očekávána třída *Person*.) Jestliže chcete použít referenci jako parametr metody, napište do textového pole její název, nebo (což je mnohem rychlejší) umístěte kurzor do textového pole parametru a klikněte levým tlačítkem na ikonu reference. Tato metoda nevrací žádnou hodnotu, a proto není žádná zobrazena. Nyní můžete zavolat metodu *listAll* a uvidíte výpis všech osob v dané databázi, při této operaci se automaticky otevře terminálové okno.

Zkuste si tento postup zopakovat pro několik dalších osob.

5 Vytvoření nového projektu

V této kapitole se budeme v krátkosti zabývat vytvářením nových projektů.

5.1 Vytvoření projektové složky

Stručně: Jestliže chcete vytvořit nový projekt, použijte příkaz Nový... *z menu* Projekt.

Jestliže chcete vytvořit nový projekt, použijte menu ***Projekt – Nový...***, zobrazí se dialog s dotazem na jméno a umístění nového projektu. Vyzkoušejte si to nyní. Po kliknutí na tlačítko *OK* bude vytvořen adresář se jménem projektu a otevře se nové hlavní okno projektu.

5.2 Vytváření tříd

Stručně: Jestliže chcete vytvořit novou třídu, použijte tlačítko Nová třída *a zadejte jméno třídy*.

Třídu můžete vytvořit pomocí kliknutí na tlačítko *Nová třída*, která je umístěna na nástrojové liště. Budete dotázáni na název nové třídy, musíte zadat platný Java identifikátor.

Můžete si rovněž vybrat z několika připravených šablon: třída, abstraktní třída, rozhraní nebo applet. Tato volba ovlivňuje pouze obsah souboru při jeho vytvoření, později můžete typ libovolně měnit pomocí úprav zdrojového kódu.

Po vytvoření třídy je do diagramu automaticky přidána nová ikona. Ikony tříd, rozhraní, abstraktních tříd a appletů jsou navzájem vizuálně odlišeny. Šablony tříd obsahují pouze ukázkový kód, lze je zkompilovat, ale nedělají mnoho. Zkuste si vytvořit několik tříd a rozhraní a zkompilujte je.

5.3 Vytváření závislostí

Stručně: Jestliže chcete vytvořit šipku reprezentující závislost, klikněte na tlačítko s šipkou a vyberte propojované ikony, můžete také přímo upravit zdrojový kód.

Diagram ukazuje závislosti mezi třídami (případně rozhraními) graficky pomocí různých šipek. Existují dva druhy šipek, jeden pro vztah dědičnosti („extends“ a „implements“), druhý pro obecnou vazbu mezi objekty.

Vztahy lze definovat graficky nebo pomocí úprav zdrojového kódu. Jestliže přidáte šipku graficky, jsou provedeny změny ve zdrojovém kódu automaticky. Jestliže provedete změny ve zdrojovém kódu, jsou po uložení automaticky promítnuty změny také do grafické podoby.

Jestliže chcete vytvořit vztah graficky, klikněte na odpovídající tlačítko se šipkou a táhněte šipku od jedné ikony ke druhé (nebo na ně postupně klikněte).

Přidání šipky „dědičnost“ automaticky vygeneruje ve zdrojovém kódu ***extends*** nebo ***implements*** (v závislosti na tom, zda se jedná o třídu, nebo rozhraní).

Při vytvoření vztahu „používá“ není generován žádný kód (jestliže je cíl umístěn v jiném balíčku, je automaticky generován příkaz ***import***). Pokud je v diagramu šipka „používá“ a ve zdrojovém kódu není tato vazba použita, je při kompilaci zobrazeno varovné hlášení.

Přidávání šipek ve zdrojovém kódu je velice jednoduché, probíhá totiž zcela automaticky po uložení změn (pamatujte si, že při zavření editoru je zdrojový kód uložen).

5.4 Odstraňování prvků

Stručně: Jestliže chcete odstranit třídu (nebo rozhraní), vyberte příkaz Odstranit ***z kontextového menu daného prvku. Jestliže chcete odstranit šipku, vyberte z menu*** Úpravy ***příkaz*** Odstranit šipku ***a klikněte na šipku.***

Jestliže chcete odstranit třídu (nebo rozhraní) z diagramu, označte jej a vyberte z menu příkaz ***Úpravy – Odstranit***. Můžete také použít kontextové menu, kde vyberete příkaz ***Odstranit***. Když chcete odstranit šipku, vyberte z menu příkaz ***Úpravy – Odstranit*** a klikněte na šipku.

6 Debugging

Toto kapitola přibližuje nejdůležitější aspekty používání debuggeru v systému BlueJ. Při rozhovoru s učiteli programování je velmi často slyšet, že by nebylo špatné během prvního roku výuky ukázat žákům základy práce s debuggerem. Studenti mají ale většinou dost starostí s editorem, kompilátorem a spouštěním, takže nakonec není nikdy dost času na další komplikovaný nástroj.

Proto jsme se rozhodli vytvořit co nejjednodušší debugger. Cílem bylo, aby základní seznámení s debuggerem nezabralo víc než 15 minut a aby mohli studenti používat debugger bez dalšího vysvětlování samostatně. Podívejme se nyní, jestli se nám to povedlo.

Nejdříve ze všeho jsem omezili funkcionalitu tradičních debuggerů na tři základní úkoly:

nastavení breakpointů,
krokování kódem,
prohlížení proměnných.

Každý z těchto úkolů je relativně jednoduchý. Podívejme se nyní postupně na jednotlivé úkoly.

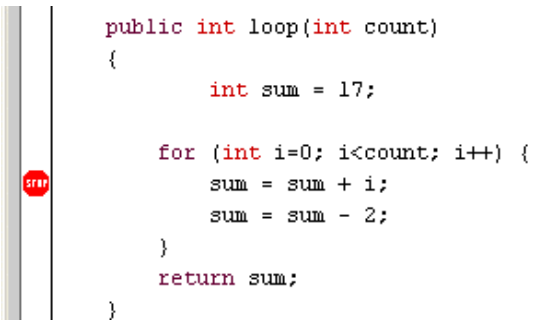
Nejprve si otevřete projekt *debugdemo*, který je rovněž součástí standardní distribuce. Tento projekt obsahuje několik tříd určených k demonstraci základních funkcí debuggeru – k ničenu jinému se ani nehodí.

6.1 Nastavení breakpointů

Stručně: Jestliže chcete nastavit breakpoint, klikněte do oblasti pro označení breakpointů v levé části editoru.

Breakpoint označuje místo v kódu, kde se dočasně přeruší vykonávání. Při tomto přerušení vykonávání můžete prohlížet vnitřní stav objektů. Často vám to pomůže pochopit, co se vlastně při vykonávání programu děje.

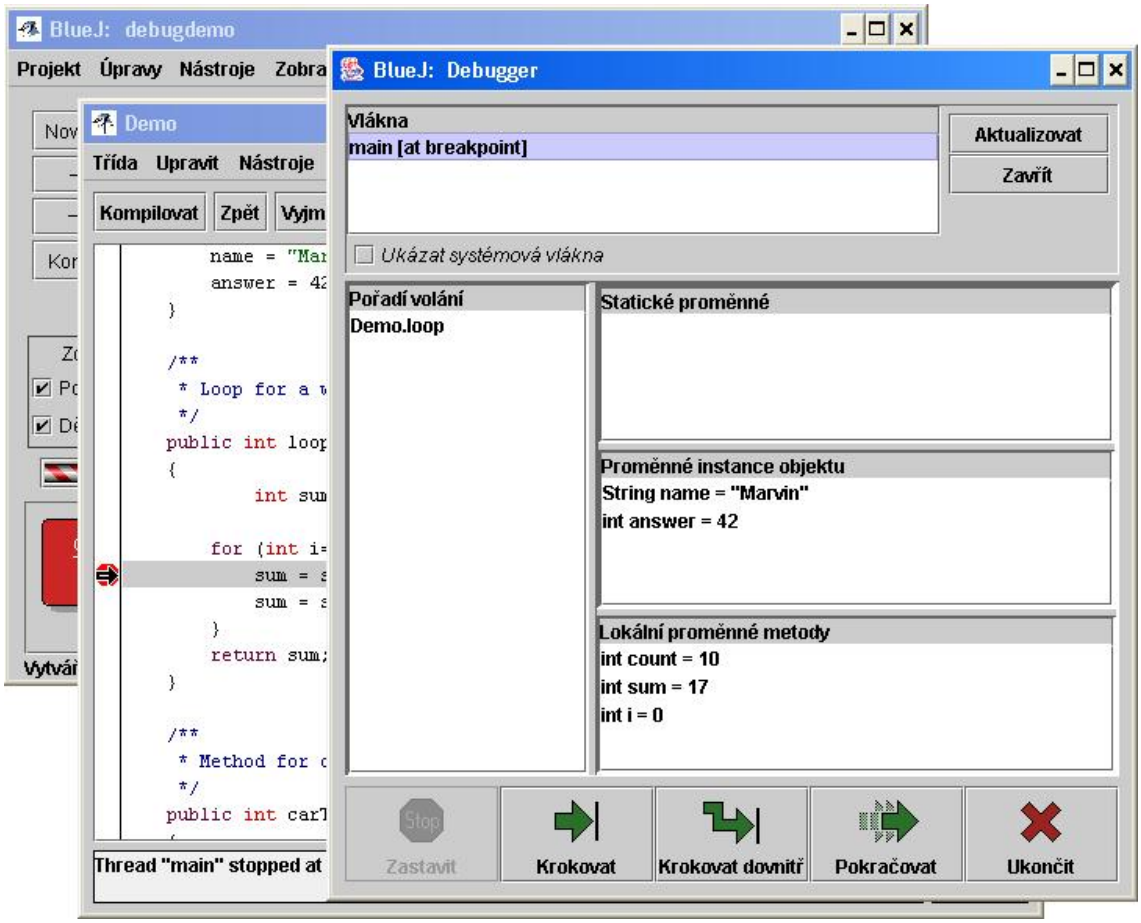
V editoru se vedle textu nachází oblast pro nastavování breakpointů. Kliknutím do této oblasti nastavíte v daném místě breakpoint. Breakpoint je označen pomocí malé ikonky se symbolem stop. Nyní si to sami vyzkoušejte. Otevřete třídu *Demo*, najděte metodu *loop* a někde v těle této metody nastavte breakpoint. Vedle kódu by se měl objevit symbol stop.



Obrázek 13: Breakpoint

Vykonávání kódu se zastaví na každém řádku, kde je nastaven breakpoint. Nyní si to vyzkoušejme.

Vytvořte instanci třídy *Demo* a zavolejte metodu *loop* s parametrem například 10. Jakmile dojde vykonávání programu na řádek s breakpointem, objeví se okno editoru se zvýrazněným řádkem a zároveň se zobrazí okno s debuggerem. Ukázka je na obrázku 14.



Obrázek 14: Okno debuggeru

Zvýraznění v editoru označuje řádek, kde bylo přerušeno vykonávání. (Vykonávání programu je přerušeno *před* tímto řádkem.)

6.2 Krokování kódem

Stručně: Jestliže chcete krokovat kódem, použijte tlačítka Krokovat *a* Krokovat dovnitř *ve spodní části debuggeru.*

Nyní, když se zastavilo vykonávání (což mimochodem dokazuje, že se metoda skutečně vykonávala a že se vykonávání dostalo až na řádek s breakpointem), můžete pokračovat po jednotlivých řádcích a sledovat postup vykonávání. Dosáhnete toho opakovaným klikáním na tlačítko *Krokovat*. V editoru můžete sledovat jak se mění označení aktuálního řádku. Při jednotlivých krocích se postupně mění hodnota vnitřních proměnných (sledujte například proměnné *sum*). Jestliže už vás nezajímá krokování, můžete obnovit normální vykonávání pomocí tlačítka *Pokračovat*.

Zkusme to znovu s jinou metodou. Nastavte breakpoint ve třídě *Demo*, v metodě *carTest*), na řádku s přiřazením

```
places = myCar.seats();
```

Zavolejte tuto metodu. Vykonávání se zastaví před tímto řádku. Při kliknutí na tlačítko *Krokovat* se vykoná celý řádek a vykonávání se zastaví opět před následujícím řádkem. Když použijete *Krokovat dovnitř*, bude se postupně vykonávat každý řádek metody *seats* (nikoliv jako předtím, kdy byla metoda *seats* vykonána v jediném kroku). Klikněte tedy na *Krokovat dovnitř* a krokujte postupně metodou *seats* třídy *Car*. Všimněte si, jak se mění okno debuggeru.

Když řádek neobsahuje volání metody, chovají se *Krokovat* a *Krokovat dovnitř* identicky.

6.3 Prohlížení proměnných

Stručně: Prohlížení proměnných je snadné – jejich hodnota je zobrazena automaticky v debuggeru.

Při používání debuggeru je důležité, aby jste byli schopni sledovat aktuální stav vnitřních proměnných objektů.

Je to velmi jednoduché – většinu z toho jste již viděli. Nepotřebujete žádné speciální příkazy; statické proměnné, proměnné instance aktuálního objektu a lokální proměnné jsou automaticky zobrazeny a aktualizovány.

V levé části debuggeru můžete vidět pořadí, v jakém byly metody volány. Poslední metoda byla volána uživatelem, v první metodě se zastavilo vykonávání.

Jestliže vyberete kliknutím metodu, můžete prohlížet proměnné této metody a příslušného objektu. Zkuste si nyní nastavit breakpoint v metodě *Car.seats()* a zavolejte metodu *carTest()* instance třídy *Demo*. Zobrazí se debugger s tímto pořadím volání:

```
Car.seats
Demo.carTest
```

To znamená, že metoda *Car.seats* byla zavolána z *Demo.carTest*. Nyní klikněte na *Demo.carTest* a bude zobrazen zdrojový kód této metody a stav jejích proměnných.

Hodnota proměnné *myCar* je zobrazena jako *<object reference>*, tímto způsobem jsou zobrazeny všechny objekty kromě třídy String. Jestliže chcete prohlížet tuto proměnnou, dvojklikněte na ni (otevře se stejné prohlížecí okno jako v kapitole 4.1). Ve skutečnosti není žádný rozdíl mezi prohlížením zde a prohlížením v zásobníku referencí.

6.4 Zastavit a ukončit

Stručně: Tlačítka Zastavit *a* Ukončit *jsou používána pro dočasné a trvalé ukončení vykonávání.*

Občas se stane, že program běží příliš dlouho, a vy nevíte, zda je vše v pořádku. Může se jednat o nekonečnou smyčku, nebo jen bude vykonávání programu trvat déle. Lze to

jednoduše zjistit. Zavolejte metodu *longloop* instance třídy *Demo*. Vykonání této metody trvá relativně delší dobu.

Nyní si otevřete debugger (pomocí příkazu z menu *Zobrazit – Zobrazit Debugger*, nebo kliknutím na pohybující se indikátor činnosti Java VM).

Nyní klikněte na tlačítko *Zastavit* a vykonávání programu bude pozastaveno. Můžete zkusit krokovat několik řádků, můžete také použít *Pokračovat* a opět *Zastavit*. Jestliže nechcete pokračovat ve vykonávání, klikněte na tlačítko *Ukončit*. Moc se na příkaz *Ukončit* nespoléhejte, tato funkce může zanechat aplikaci a celý systém BlueJ v nekonzistentním stavu. Jestliže vše nefunguje tak, jak má, restartujte raději celé prostředí BlueJ.

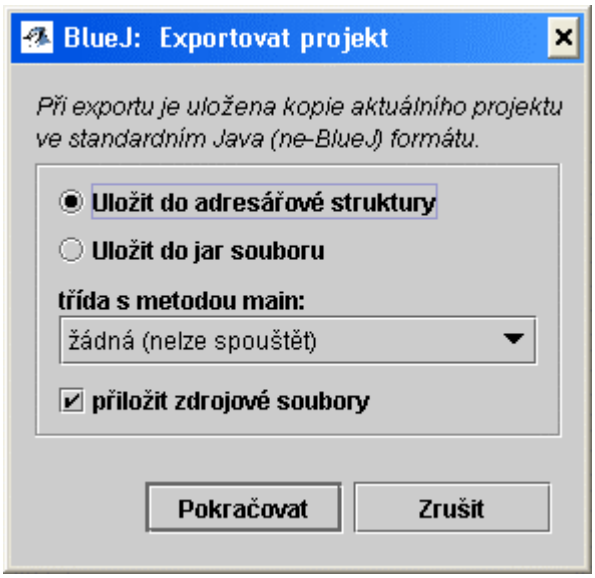
7 Vytváření samostatných aplikací

Stručně: Jestliže chcete vytvořit samostatnou aplikaci, vyberte příkaz Export... *z menu* Projekt.

V BlueJ můžete vytvářet spustitelné jar soubory. Spustitelné jar soubory jsou ve většině systémů (Windows a MacOS X) spouštěny pomocí dvojkliknutí, nebo pomocí příkazu `java –jar <file-name>.jar` (Unix, příkazový řádek DOS).

Vyzkoušíme si to nyní na projektu *hello*. Tento projekt je součástí standardní distribuce, otevřete ho nyní a zkompilujte všechny třídy. Vyberte z menu *Projekt* příkaz *Export...*.

Na obrazovce se zobrazí dialogové okno s volbami pro export (viz obrázek 15). Označte volbu *Uložit do jar souboru* a vyberte třídu s metodou main - tato metoda musí mít hlavičku `public static void main(String[] args)`.



Obrázek 15: Exportovat projekt

Tento projekt obsahuje pouze jednu metodu, proto se nemůžete při výběru splést. Soubory se zdrojovým kódem nejsou obvykle přikládány do spustitelných jar archívů.

Nyní klikněte na tlačítko *Pokračovat*, poté zadejte umístění a jméno jar souboru (v tomto případě použijte jméno *hello*). Spustitelný soubor jar je vytvořen.

Jestliže daná aplikace používá grafické rozhraní, můžete ji spustit dvojkliknutím. Aplikace hello používá pouze textový vstup/výstup, musíte ji tedy spustit pomocí textového terminálu.

Otevřete příkazový řádek DOS a přejděte do složky, kam jste umístili soubor *hello.jar*. Aplikaci spustíte příkazem

```
java –jar hello.jar
```

8 Vytváření appletů

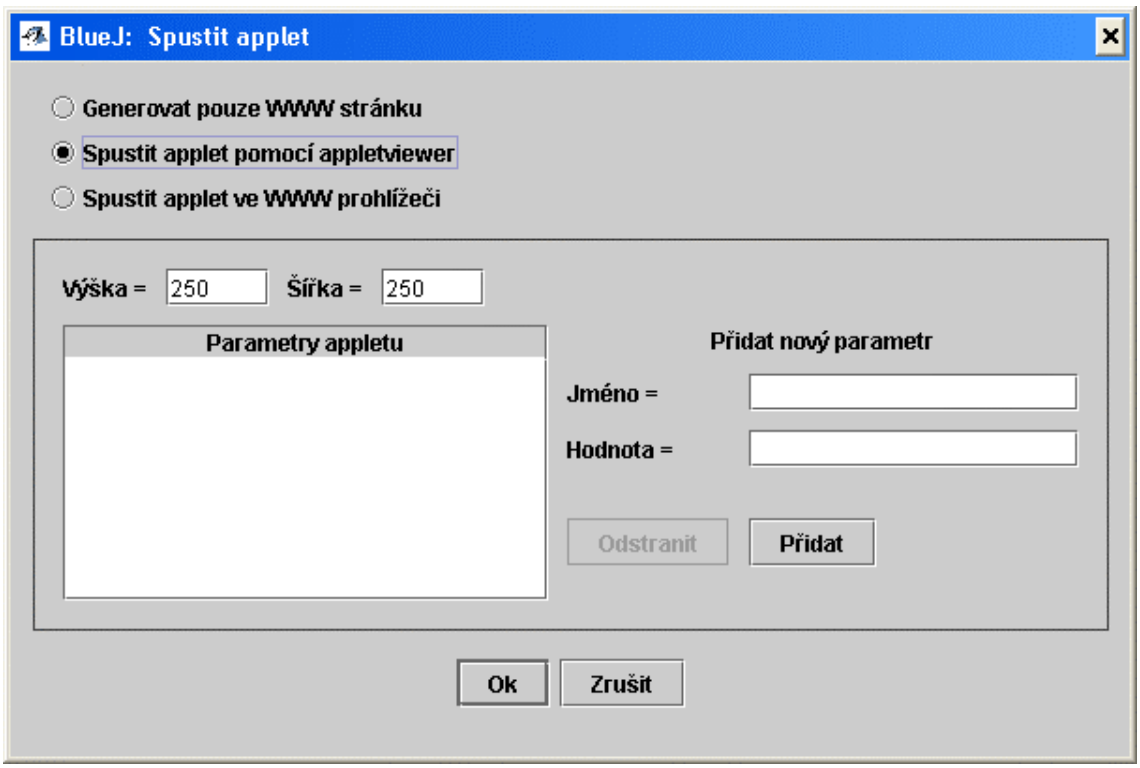
8.1 Spouštění appletů

Stručně: Jestliže chcete spustit applet, vyberte z kontextového menu appletu příkaz Spustit applet.

BlueJ umožňuje vytváření a spouštění appletů a aplikací. Jeden ukázkový applet je součástí standardní distribuce, najdete ho ve složce *examples*. Nejdříve si vyzkoušíme spouštění appletu. Nyní otevřete projekt *appletdemo*.

Tento projekt obsahuje pouze jednu třídu, která je pojmenována *CaseConverter*. Ikona třídy je odlišena pomocí nápisu *<<applet>>*. Zkompilujte tuto třídu a v jejím kontextovém menu klikněte na příkaz *Spustit applet*.

Na obrazovce se zobrazí dialog s několika volbami (viz obrázek 16).



Obrázek 16: Spouštění appletu

Můžete si vybrat, zda má být applet spuštěn ve WWW prohlížeči, nebo v prohlížeči appletů (můžete také nechat vygenerovat pouze html stránku). Ponechte implicitní nastavení a klikněte na *OK*. Po chvíli se zobrazí appletviewer s appletem CaseConverter.

Appletviewer je standardní součástí instalace Java SDK. Při spouštění appletů ve WWW prohlížeči mohou nastat problémy s různými verzemi Javy. Poslední verze prohlížečů doplněné o aktuální verzi Javy pracují obvykle bezchybně.

Ve Windows a MacOS používá BlueJ automaticky výchozí prohlížeč. V prostředí Unix je prohlížeč nastaven pomocí konfiguračního souboru *bluej.def*.

8.2 Vytváření appletů

Stručně: Jestliže chcete vytvořit applet, použijte při vytváření nové třídy šablonu Applet.

Už víte, jak spustit applet. Teď si ukážeme, jak vytvořit applet.

Z menu *Úpravy* vyberte příkaz *Nová třída...*, zvolte šablonu *Applet*. Nyní zkompilujte nově vytvořenou třídu a spusťte applet. To je vše, nebylo to zase tak těžké, že?

Applety jsou (stejně jako ostatní třídy a rozhraní) vytvářeny pomocí přednastavených šablon. Nově vytvořený applet můžete upravit pomocí editoru, zdrojový kód obsahuje všechny nejčastěji používané metody spolu s komentářem. Výkonná část kódu je umístěna do metody *paint*.

8.3 Testování appletů

Někdy může být vhodné vytvořit instanci appletu přímo pomocí konstruktoru a umístit referenci v zásobníku referencí. V prostředí BlueJ nemůžete spustit applet přímo, můžete ale volat jednotlivé metody. Toho můžete využít při testování jednotlivých metod, které nejsou vázány na specifické vlastnosti appletů.

9 Ostatní operace

9.1 Otevírání ne-BlueJ balíčků v BlueJ

Stručně: Projekty, které nebyly vytvořeny v prostředí BlueJ, můžete otevřít pomocí příkazu Otevřít ne-BlueJ... *z menu* Projekt.

BlueJ umožňuje otevírání balíčků, které nebyly vytvořeny v prostředí BlueJ. Takové balíčky otevřete pomocí příkazu *Otevřít ne-BlueJ...* z menu *Projekt*, budete dotázáni na složku, kde se nachází balíček.

9.2 Přidávání existujících tříd do projektu

Stručně: Jestliže chcete importovat do projektu již existující třídu, vyberte příkaz Nová třída ze souboru... *z menu* Úpravy.

Velmi často je potřeba přidat do projektu již existující třídu. Tato činnost je usnadněna pomocí příkazu *Nová třída ze souboru...* z menu *Úpravy*, obsah vybraného souboru s příponou *java* je použit pro vytvoření nové třídy v aktuálním projektu.

Alternativní způsob je zkopírovat nový soubor s třídou přímo do složky projektu. V diagramu bude nová třída zobrazena až při příštím otevření projektu.

9.3 Volání *main* a ostatních statickým metod

Stručně: Statické metody mohou být volány z kontextového menu třídy.

Otevřete projekt *hello* umístěný ve složce *examples*. Tento projekt obsahuje pouze jednu třídu *Hello*, tato třída má standardní metodu *main*.

Klikněte pravým tlačítkem na ikonu třídy Hello, v kontextovém menu můžete vidět kromě konstruktoru také statickou metodu *main*. Zavolejte tuto metodu přímo bez předchozího vytvoření instance. Všechny statické metody mohu být volány tímto způsobem. Standardní metoda *main* očekává jako vstupní parametr pole řetězců znaků. Jako parametr použijte

```
{"one", "two", "three"}
```

tento zápis je shodný se standardní syntaxí jazyka Java. Vyzkoušejte si to nyní.

> Poznámka: Ve standardní Javě není povoleno používání konstant typu pole jako vstupního parametru metod. Mohou být použity pouze pro inicializaci proměnných. Prostředí BlueJ je zaměřeno na interaktivitu volání metod, a proto je zde dovoleno používat konstantu typu pole jako vstupní parametr metody.

9.4 Dokumentace projektu

Stručně: Jestliže chcete generovat a zobrazit dokumentaci projektu, použijte příkaz Dokumentace projektu *z menu* Nástroje.

Dokumentace k projektu je generována ve standardním *javadoc* formátu. Jestliže chcete generovat a poté zobrazit dokumentaci projektu, použijte z menu *Nástroje* příkaz *Dokumentace projektu.* Dokumentace je zobrazena v externím www prohlížeči.

Podobně můžete v editoru zobrazit dokumentaci aktuálně otevřené třídy. Dokumentaci zobrazíte pomocí přepínače *Implementace/Popis rozhraní.*

9.5 Dokumentace standardních tříd

Stručně: Dokumentace API standardních tříd Javy může být zobrazena pomocí příkazu Knihovna tříd Java *z menu* Nápověda.

Dokumentace API standardních tříd Javy může být zobrazena pomocí příkazu *Knihovna tříd Java* z menu *Nápověda*. Nápověda je zobrazena v externím www prohlížeči.

Dokumentace JDK může být nainstalována také lokálně (offline). Detailní popis je na www stránkách BlueJ.

9.6 Vytváření instancí z knihoven tříd

Stručně: Jestliže chcete vytvořit instanci standardní třídy, použijte příkaz Volat třídu z knihovny... *z menu* Nástroje.

BlueJ umožňuje vytvoření instance třídy, která není součástí projektu. Můžete například vytvořit instanci třídy *java.lang.String*, nebo *java.util.ArrayList.* Tato funkce může usnadnit experimentování s objekty ve standardních knihovnách.

10 Shrnutí

Začínáme

1. Projekt otevřete pomocí příkazu *Otevřít...* z menu *Projekt.*
2. Novou instanci objektu vytvoříte tak, že vyberete konstruktor z kontextového menu třídy.
3. Metodu můžete zavolat z kontextového menu třídy nebo reference.
4. Zdrojový kód třídy zobrazíte dvojkliknutím na ikonu třídy.
5. V editoru zkompilujete třídu kliknutím na tlačítko *Kompilovat*. Celý projekt zkompilujete kliknutím na tlačítko *Kompilovat* v projektovém manažeru.
6. Jestliže si přejete zobrazit podrobnější nápovědu k chybě kompilátoru, klikněte na tlačítko s otazníkem.

Něco navíc...

7. Prohlížení objektu slouží k zobrazení vnitřního stavu instance objektu.
8. Instance objektu může být předána jako vstupní parametr metody pomocí kliknutí na ikonu reference.

Vytvoření nového projektu

9. Nový projekt vytvoříte pomocí příkazu *Nový...* z menu *Projekt.*
10. Novou třídu vytvoříte tlačítkem *Nová třída*.
11. Jestliže chcete vytvořit šipku reprezentující závislost, klikněte na tlačítko s šipkou a vyberte propojované ikony, můžete také přímo upravit zdrojový kód.
12. Třídu odstraníte pomocí příkazu *Odstranit* z kontextového menu daného prvku.
13. Šipku odstraníte pomocí příkazu *Odstranit šipku* z menu *Úpravy*.

Debugging

14. Jestliže chcete nastavit breakpoint, klikněte do oblasti pro označení breakpointů v levé části editoru.
15. Kódem krokujete pomocí tlačítek *Krokovat* a *Krokovat dovnitř* umístěných ve spodní části debuggeru.
16. Hodnoty proměnných jsou v debuggeru zobrazeny automaticky.
17. Tlačítka *Zastavit* a *Ukončit* jsou používána pro dočasné a trvalé ukončení vykonávání chodu programu.

Vytváření samostatných aplikací

18. Samostatnou aplikaci vytvoříte pomocí příkazu *Export...* z menu *Projekt.*

Vytváření appletů

19. Applet spustíte pomocí příkazu *Spustit applet* z kontextového menu třídy.
20. Jestliže chcete vytvořit applet, použijte při vytváření nové třídy šablonu *Applet*.

Ostatní operace

21. Projekty, které nebyly vytvořeny v prostředí BlueJ, můžete otevřít pomocí příkazu *Otevřít ne-BlueJ...* z menu *Projekt.*

11 Poznámky překladatele

11.1 Čeština v Javě

Čeština v JDK pro Windows od firmy SUN funguje správně od verze 1.4. Jestliže chcete doinstalovat podporu pro jinou verzi JDK, podívejte se na stránky http://www.java.cz nebo http://www.rdv.vslib.cz/skodak.

11.2 Česká lokalizace BlueJ

Česká lokalizace je součástí standardní distribuce BlueJ od verze 1.2.1.

Lokalizace pro daný jazyk je aktivována manuálně v souboru *bluej.defs*, soubor upravte následovně:

```
#bluej.language=english
#bluej.language=afrikaans
#bluej.language=chinese
bluej.language=czech
#bluej.language=german
```

Upravený soubor můžete získat také na adrese http://www.rdv.vslib.cz/skodak.