Západočeská univerzita v Plzni
Fakulta aplikovaných věd
Katedra informatiky a výpočetní techniky

# Bakalářská práce

# Srovnání stávajících IDE pro jazyk Java

Plzeň 2016 Petr Zeman

# Prohlášení

Prohlašuji, že jsem bakalářskou práci vypracoval samostatně a výhradně s použitím citovaných pramenů.

V Plzni dne 3. května 2016

Petr Zeman

# Poděkování

Rád bych poděkoval panu Ing. Tomáši Potužákovi, Ph.D. za čas, rady a připomínky poskytnuté při vytváření této bakalářské práce.

# Abstract

Title: Comparison of existing IDE for Java Language

The aim of this thesis is to research the existing integrated development enviroments for Java language and then comparison of several of them. For the comparison, the BlueJ and Greenfoot IDEs, which are designed to learning programming, and Eclipse and Netbeans IDEs, which are intended for application development, were chosen. All selected IDEs are described in this thesis in detail and compared, so it can help to readers in choosing right application for their own work.

Key words: Java, IDE, BlueJ, Greenfoot, Eclipse, Netbeans

# Abstrakt

Cílem této bakalářské práce je provést rešerši existujících vývojových prostředí pro jazyk Java a následné porovnání několika z nich. K tomu byla zvolena prostředí BlueJ a Greenfoot, která jsou určená k výuce programování, a prostředí Eclipse a Netbeans, která jsou určená pro vývoj aplikací. Všechna uvedená prostředí jsou v práci podrobně popsána a porovnána, což může čtenáři pomoci při volbě správného prostředí pro vlastní práci.

Klíčová slova: Java, vývojové prostředí, BlueJ, Greenfoot, Eclipse, Netbeans

# Obsah

# 1 Úvod

Tématem bakalářské práce je srovnání stávajících IDE (Integrated Development Enviroment — Integrované vývojové prostředí) pro jazyk Java. Práce si klade za cíl nejprve čtenáře seznámit s běžnými vlastnostmi a nástroji, které vývojová prostředí obvykle obsahují, a tedy i s tím, co vše mu mohou umožnit. Řada těchto nástrojů je během let neustále vyvíjena a zdokonalována, aby co nejvíce ulehčila práci programátorů při psaní zdrojových kódů.

V další části práce nastiňuje seznam dostupných vývojových prostředí, ať už zamýšlených pro výuku nebo vývoj aplikací, která lze v současné době nalézt na trhu. Z kandidátských vývojových prostředí jsou následně vybrána čtyři (dvě výuková a dvě pro vývoj aplikací). Tato prostředí jsou v další části práce důkladně prozkoumána a otestována na základě zvolených kritérií. Produkty hodnotí nejen autor této práce, ale i další dotazovaní s odlišnými programátorskými zkušenostmi. Díky tomu by měl čtenář získat poznatky o tom, které vývojové prostředí by mohlo být nejvhodnější pro jeho práci vzhledem k jeho zkušenostem a zvyklostem.

# 2 Základní pojmy

Tato kapitola definuje několik základních pojmů, které by měl čtenář znát před započetím čtení této práce, jelikož jsou tyto pojmy v práci často zmiňovány.

## 2.1 Jazyk Java

Jazyk Java je objektově orientovaný programovací jazyk vyvinutý firmou Sun Microsystems, jehož syntaxe vychází z jazyka C a C++. Zvláštností toho jazyka je, že se nepřekládá do strojového kódu daného počítače, ale do jazyka zvaného bajtkód (bytecode). Program je pak interpretován pomocí virtuálního stroje (JVM – Java Virtual Machine), díky čemuž lze Java programy využívat na různých zařízeních s různým operačním systémem bez nutnosti nového překladu. V současnosti se využívá technologie *HotSpot*, která se kombinuje s *JIT (Just-In-Time)* kompilátory, které přeloží bajtkód do strojového jazyka počítače, což urychlí běh samotného programu [1].

Jazyk Java je jednodušší oproti jazyku C hlavně díky tomu, že programátor nemusí řešit správu paměti. To za něj řeší právě JVM, jenž aplikace žádají o přidělení a uvolnění paměti. K tomu JVM využívá Garbage Collector, který automaticky vyhledává úseky paměti vhodné k uvolnění. Díky tomu se předchází nechtěným únikům paměti při případném neuvolnění v programu [2].

## 2.2 Java prostředí

Prostředí Java je dostupné ve dvou verzích, a to ve verzi JRE a JDK. JRE (Java Runtime Enviroment) je běhové prostředí Javy, které je nutné mít nainstalované pro možnost spouštět na počítači programy psané v Javě. Pro možnost vyvíjet a následně spouštět své vlastní programy je ovšem nutné mít nainstalované druhé prostředí, tedy JDK (Java Development Kit), což je prakticky běhové prostředí obohacené o vývojové nástroje (překladač, nástroj pro generování dokumentace z komentářů javadoc atp.) [3].

## 2.3 Vývoj aplikací

Jak bylo řečeno, vyvíjet aplikace lze na počítačích, které obsahují JDK. Pak je již možné vytvářet zdrojový kód, který bude vykonávat určitou činnost. Ten je možné napsat v podstatě v libovolném textovém editoru, který neukládá žádné informace o formátování, ale pouze samotný text. Příkladem takového editoru je jednoduchý Poznámkový blok obsažený v systému Windows. Napsat ovšem i krátký program v tomto editoru není vůbec jednoduché, protože nám práci nijak neusnadní [4].

Možností je tedy doinstalovat sofistikovanější editor, který kód alespoň trochu zpřehlední. Mezi takové editory patří například PSPad, Note++, Scite nebo Vim. Tyto editory umožňují zvýraznit syntaxi zvoleného jazyka a tím barevně odlišit například datové typy, řetězce, komentáře a další úseky zdrojového kódu, čímž se kód stává lépe čitelnější. Některé editory umožňují i automatické odsazování, statické doplňování kódu či dokonce překlad a spuštění programu (s využitím vnějších překladačů). Pokud překlad a spuštění nedovolují, je nutné to provádět manuálně zadáváním příkazů v příkazové řádce systému [5].

Při psaní programů ovšem programátor potřebuje mnohem více podpořit svůj vývoj. Potřebuje řadu nástrojů, které lze všechny používat z jednoho místa a umožní mu snadněji vytvářet zdrojový kód nebo rychleji odhalit případné chyby. A právě kvůli tomu byla vytvořena vývojová prostředí.

## 2.4 Vývojové prostředí

Vývojové prostředí (IDE) je samostatný software umožňující efektivní vývoj aplikací. IDE shrnuje schopnosti mnoha nástrojů a funkcí pro vývoj, překlad a ladění programu do jednoho softwaru. Umožňuje tedy programátorovi přistupovat z jednoho místa ke všem nástrojům, které může během své práce potřebovat. Cílem IDE je tedy co nejvíce ulehčit a urychlit práci programátora [6].

## 2.5 Vývojová platforma

Vývojovou platformu lze definovat jako sadu frameworků a služeb, které dohromady vytváří infrastrukturu, kterou lze využít jako základ našich aplikací. Tato minimální sada modulů je často označována jako *Rich client platform (RCP)*. RCP přináší řadu funkcionalit, které jsou nedílnou součástí každé klientské aplikace, jako je grafické rozhraní a práce s ním, práce s daty, ukládání stavu a mnoho dalšího. Je tedy možné využít těchto již odzkoušených funkcionalit a přidat pouze vlastní moduly specifické pro naši aplikaci. Tím lze ušetřit velké množství času a práce [7, 8].

Některé příklady vývojových platforem jsou zmíněny i v této práci, jelikož slouží jako základ některým zde popisovaným vývojovým prostředím.

# 3 Vlastnosti IDE a jejich nástroje

Vlastnost, kterou mají IDE společnou je ta, že pracují nad celými projekty. Než tedy programátor začne vytvářet zdrojový kód, musí nějaký projekt založit. Tím na disku vznikne adresář, který bude obsahovat veškeré soubory týkající se tohoto projektu. Tyto soubory už IDE ukládají sama podle toho, jak mají definovanou adresářovou strukturu. Standardně ukládají zdrojové a binární soubory do vlastních podadresářů. To platí i pro jednotkové testy a metadata týkající se projektu. Neplatí to ovšem pro všechna vývojová prostředí. Například BlueJ tuto hierarchii naprosto ignoruje a ukládá všechny soubory do jednoho adresáře [9].

Zdrojové soubory (třídy), které spolu nějak funkčně souvisí, se v Javě sdružují do jednotlivých balíků. Hlavním důvodem je přehlednost, protože velké projekty mohou obsahovat klidně i stovky tříd. Jak jsou třídy rozloženy do jednotlivých balíků, IDE zobrazují prostřednictvím okna *Package Explorer (průzkumník balíků)*. Ten je obvykle zobrazen ve výchozím vzhledu prostředí, stejně jako například okno pro chybová hlášení nebo konzole [10].

Jelikož jsou ovšem IDE velmi intuitivní, dovolují většinou uživatelům nastavit si vzhled prostředí podle toho, co potřebují během vývoje programů sledovat. Hlavní a nejdůležitější okno zde ovšem bude vždy, a tím je editor, který je spolu s dalšími hlavními nástroji rozebrán v následujících bodech práce.

## 3.1 Editor

Editorem se u vývojových prostředí nazývá prostor, ve kterém lze zapisovat zdrojový kód. Jednotlivé editory se od sebe mohou lišit pouze vzhledem a způsobem zobrazování zapisovaného kódu. Co mají ovšem společné je použití neproporcionálního fontu písma. U takového písma mají všechny znaky stejnou šířku, takže je kód zarovnaný a dobře čitelný. Příkladem takového písma je například `Courier New`, `Consolas` nebo `Lucida` [11].

Samozřejmě je možné si editor nastavit podle svých zvyklostí. A tím nemusí být jen nastavení fontu písma, ale i definování vlastních klávesových zkratek, odsazení apod. S editorem souvisí i následující 3 body této práce.

## 3.2 Zvýraznění syntaxe

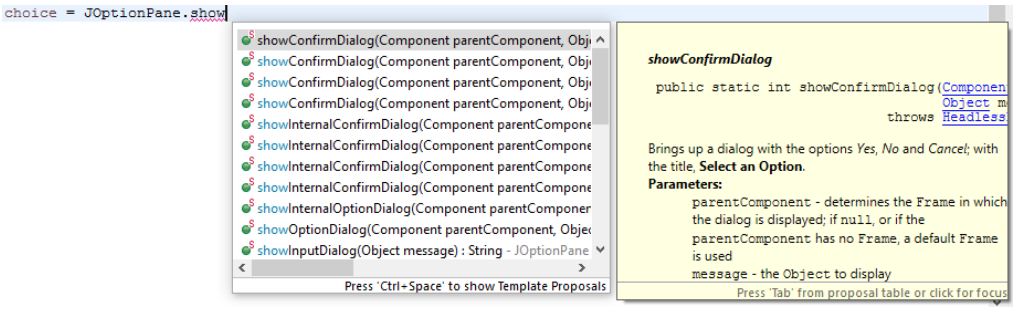
Zvýraznění syntaxe standardně obsahují všechna vývojová prostředí. Tato jednoduchá funkce zajišťuje změnu barvy, případně i změnu řezu písma u datových typů, komentářů, řetězců a dalších úseků, čímž je kód mnohem čitelnější (viz Obrázek 3.1).

```
this.window = window;        //Input parameter
this.setPreferredSize(new Dimension(WIDTH, MainWindow.HEIGHT));
this.setBackground(Color.WHITE);
setLayout(null);
listGame = new GameList(this.window, this.client);
```

Obrázek 3.1: Příklad zvýraznění syntaxe jazyka Java

## 3.3 Kontextová nápověda

Kontextová nápověda je v dnešní době standardní funkcí většiny vývojových prostředí. Jedná se o práci ulehčující nástroj, který programování výrazně zefektivní. Během psaní kódu totiž našeptává programátorovi, co je možné dále doplnit či využít vzhledem k dosud napsanému výrazu. U nalezených možností, jako jsou například metody, může zobrazovat i další informace, např. Java dokumentaci. Nápověda se zobrazuje buď automaticky nebo se dá vyvolávat manuálně stisknutím klávesové zkratky [12]. Příklad kontextové nápovědy je vidět na Obrázku 3.2.




Obrázek 3.2: Kontextová nápověda v prostředí Eclipse

Poprvé se tato funkce (ve formě jak ji dnes známe) objevila v roce 1996 v prostředí Visual Basic 5.0 firmy Microsoft. Funkce dostala pojmenování

*IntelliSense*. Později obdobu této funkcionality implementovala i další vývojová prostředí [13].

## 3.4 Kontrola kódu

Kdyby se programátor dozvěděl o všech chybách až během překladu kódu, musel by je všechny zpětně dohledávat a opravovat, což by jeho práci zpomalilo. Proto IDE kontrolují kód už během jeho zápisu a informují programátora o nalezených chybách. Řádek, ve kterém se vyskytla chyba, barevně označí, nebo u něj zobrazí určitou chybovou či varovnou značku. Chybami mohou být například [14]:

- chybějící závorky,
- zaměnění velkých/malých písmen, překlepy,
- chybějící návratová hodnota,
- chybějící importy,
- špatné vstupní parametry metody,
- porovnávání různých datových typů apod.

Kromě chyb se zobrazují i různá varování. Ty sice neznemožňují překlad kódu, ale upozorňují na to, že by bylo vhodné kód upravit (např. odstranit nepoužívané proměnné).

## 3.5 Překlad a spouštění programu

Samozřejmostí u vývojových prostředí je možnost kompilovat a spouštět program jedním stisknutím tlačítka. Díky tomu je programátor osvobozen od opakovaného procesu zadávání kompilačních příkazů v příkazové řádce, což mu velmi zpříjemní práci. Výstupy i případné vstupy programu jsou přesměrovány do vlastního terminálu prostředí.

## 3.6 Generování dokumentace

Automatické generování dokumentace provádí nástroj Javadoc, který je součástí prostředí JDK. Tento nástroj umí procházet zdrojový kód a vyhledávat speciální víceřádkové komentáře uvozené značkami `/**` a `*/` (příklad uveden na Obrázku 3.3). Tyto komentáře mohou být v IDE barevně odlišeny od běžných komentářů a vztahují se k třídám, metodám, konstruktorům či proměnným třídy a instance. Měly by obsahovat předdefinované značky, díky kterým vznikne stejná dokumentace, jako je k dispozici pro třídy API (Application Programming Interface — aplikační programové rozhraní). V případě, že programátor vytváří tyto komentáře už během vývoje programu, stačí mu po jeho dokončení vygenerovat dokumentaci ve formátu HTML tímto nástrojem [15].

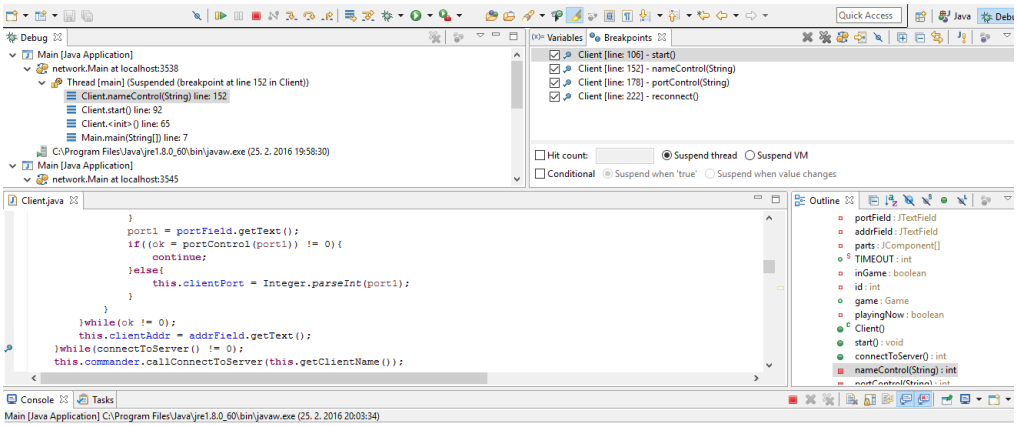
```
/**
 * Description of countingMethod(int a, int b, char c)
 * @param a Description of a
 * @param b Description of b
 * @param c Description of c
 * @return d Description of d
 */
```

Obrázek 3.3: Příklad dokumentačního komentáře v jazyce Java [15]

## 3.7 Debugger

Debugger je nástroj určený pro ladění programu. Tento nástroj urychluje celkový vývoj programů, protože napomáhá programátorovi rychle najít chyby v jeho vytvořeném kódu. V editoru lze totiž u jakéhokoliv řádku kódu nastavit tzv. *breakpoint (bod přerušení)* a spustit proces ladění. Po spuštění programátorovi umožní najednou sledovat zdrojový kód, výpisy programu v terminálu a rovněž hodnoty definovaných proměnných (viz Obrázek 3.4). Program pracuje, jako by byl standardně spuštěn, ovšem v nastavených bodech přerušení se pozastaví. Tím umožní programátorovi sledovat, jakých hodnot nabývají proměnné v daný okamžik, s jakými parametry je vyvolána sledovaná metoda, jaký je obsah paměti a další. Díky tomu dokáže snáze

zjistit příčinu chyby a kód opravit. Program je možné ladit i po krocích, tedy příkaz po příkazu sledovat, co se v programu děje [16].




Obrázek 3.4: Příklad ladění programu v Eclipse

## 3.8 Testovací nástroje

K vytvoření co nejkvalitnějšího programu je nutné ho opakovaně testovat a modifikovat podle výsledků testů. K testování se používají takzvané *Unit (jednotkové) testy*. Tyto testy, které si musí programátor sám vytvořit, slouží k otestování jednotlivých tříd a metod. Standardem pro Unit testování v Javě jsou nástroje JUnit a TestNG, které jsou integrovány s nejrozšířenějšími vývojovými prostředími. Pro kvalitní otestování je ovšem nutné provést kromě Unit testů i statickou analýzu kódu a výkonové testy [17, 18].

Pro statickou analýzu kódu lze použít nástroj FindBugs nebo PMD. Oba tyto nástroje jsou integrované s mnoha vývojovými prostředími. Při své analýze se zaměřují především na duplicitní a nepoužívaný kód, odhalí rovněž i prázdné klauzule try/catch/finally, chybějící dokumentaci apod. [18].

Výkonové testy vyhledávají tzv. „úzká místa“ programu a umožňují je odstranit už během jeho vývoje. Tyto testy předchází případnému zpomalování některých operací při rostoucí zátěži programu. Pro výkonové testy lze využít například nástroj JMeter [18].

## 3.9 Buildovací nástroje

Buildovací nástroje jsou zodpovědné za sestavení zdrojového kódu v jazyce Java do výsledné aplikace, podobně jako nástroj *Make* v jazyce C. Příkladem může být nástroj Ant nebo novější Maven. Oba jsou již většinou součástí nejrozšířenějších vývojových prostředí [18].

## 3.10 Verzovací nástroje

Verzovací nástroje umožňují vytvářet jednotlivé verze (zálohy) zdrojových kódů a důležitých souborů. Jasně definují úložiště, do kterého se budou soubory ukládat a umožní s ním pracovat přímo v prostředí. V případě problémů s vyvíjeným programem se tedy lze snadno vrátit ke starší, fungující verzi. Při práci na týmových projektech jsou verzovací systémy ideální ke sdílení zdrojových kódů s ostatními spolupracovníky [19].

Verzovací systémy se rozdělují do dvou kategorií, a to na centralizované a distribuované (decentralizované). U centralizovaných systémů se jedná o klasické spojení klient-server, kdy pouze na serveru je uložena kompletní historie verzí. Centralizované verzovací systémy jsou například CVS (Control Version System) či SVN (Subversion). Distribuované systémy se liší tím, že všechny „uzly“ jsou rovnocenné. Všechny obsahují kompletní historii změn, takže se lze snadno vrátit k libovolné verzi bez nutnosti dotazování se serveru. Tento způsob používají například systémy Git či Mercurial [19].

# 4 Rešerše existujících IDE

Při průzkumu trhu lze nalézt velké množství vývojových prostředí pro jazyk Java. Průzkum ovšem trochu komplikuje rozšířenost daných prostředí. Hledání totiž často směřuje ke stále stejným výsledkům, a to k nejpoužívanějším prostředím jako je Netbeans, Eclipse, IntelliJ apod. Podle očekávání se potvrdilo, že prozatím neexistuje žádná knižní literatura zabývající se tímto tématem, respektive jsem žádnou takovou nenalezl. Proto jsem obrátil pozornost k webovým zdrojům, které sice neprovádějí žádná srovnání, ale alespoň poskytují seznam několika vývojových prostředí [20, 21].

## 4.1 NetBeans

Netbeans je bezplatné vývojové prostředí nyní již vlastněné firmou Oracle Corporation. Samotné IDE je naprogramováno v jazyce Java, pro který je i primárně určeno. Umožňuje ovšem i programování v jiných jazycích jako je PHP, C/C++, HTML5 a další. I díky tomu jde o jedno z nejoblíbenějších a nejrozšířenějších vývojových prostředí, které může pracovat na různých operačních systémech. Charakteristický je snadným použitím a tendencí k jednoduchosti. To dokazuje i tím, že je ho možné nainstalovat pouze s balíky technologií, které bude programátor skutečně využívat. Netbeans rovněž disponuje poměrně velkou komunitou uživatelů a vývojářů, která přispívá k vývoji modulů rozšiřujících toto IDE [22, 23].

## 4.2 Eclipse

Eclipse je software vyvinutý firmou IBM a stejně jako Netbeans je neplacený. Jeho hlavní výhodou je nepřeberné množství pluginů, pomocí kterých lze rozšířit. Řada jich je vyvíjena pouze pro toto prostředí a žádné jiné, z čehož Eclipse těží. Primárně je určen pro vývoj v jazyce Java, ale lze rozšířit i o další jazyky. V současnosti se jedná o pravděpodobně nejpoužívanější IDE, což je dáno hlavně díky zmíněnému množství pluginů, o které lze rozšířit [22].

## 4.3 IntelliJ IDEA

Vývojové prostředí IntelliJ IDEA od firmy JetBrains je dostupné ve dvou verzích. Konkrétně se jedná o placenou verzi Ultimate Edition a bezplatnou verzi Community Edition. Bezplatná verze nabízí základní programovací možnosti pro jazyk Java, Scala a Groovy. Placená verze oproti tomu obsahuje širší nabídku nástrojů a podporovaných technologií. Příkladem mohou být nástroje pro práci s databázemi, podpora vývoje aplikací v Java EE či podpora jazyků SQL, Javascript, PHP a dalších. IntelliJ IDEA je stejně jako Netbeans či Eclipse jedno z nepoužívanějších prostředí současnosti [24].

K používání verze Ultimate Edition je zapotřebí vlastnit jednu z licencí. Jejich seznam a popis je uveden v Tabulce 4.1.

Tabulka 4.1: Licence pro verzi Ultimate Edition [25]

| **Licence** | **Popis použití** | **Cena (na rok)**[1] |
|---|---|---|
| Komerční | Určena pro komerční použití ve firmách a organizacích. | 12 900 Kč/osoba |
| Individuální | Určena pro jednotlivce, který si licenci hradí z vlastních (nefiremních) zdrojů. Umožňuje komerční použití. | 3 990 Kč/osoba |
| Studentská | Určena pro nekomerční použití, vzdělávání a akademický výzkum studentů a učitelů. Licence se váže na jednotlivce. | Zdarma (po schválení) |
| Akademická | Určena pro akreditované výukové organizace (výuka ve třídách). Stačí jedna licence pro neomezený počet uživatelů. | Zdarma (po schválení) |
| Open-source | Určena pro vývoj open-source projektů. | Zdarma (po schválení) |

Společnost JetBrains v minulosti uvolnila platformu IntelliJ, na které je založena řada vývojových prostředí této společnosti (IntelliJ IDEA, PHPStorm, AppCode a další). Tuto open-source platformu pokrývá licence *Apache 2.0*,

[1]Ceny platí pro první rok používání produktu, v dalších letech jsou již ceny nižší. Licence, které jsou k dispozici zdarma, musí být každoročně obnovovány a schváleny.

což znamená, že na jejích základech lze bezplatně stavět open-source i komerční produkty. Využít ji ovšem lze pouze pro aplikace založené na editoru kódu (vývojová prostředí) [26].

## 4.4 JDeveloper

JDeveloper je bezplatné vývojové prostředí distribuované firmou Oracle Corporation. Oproti jiným vývojovým prostředím se ovšem nesnaží být univerzální a nabízet možnosti rozšíření prostředí. O tom svědčí malé množství dostupných pluginů, ze kterých většinu napsal Oracle sám. Prostředí se váže na další software od Oraclu, především na databáze, což je jeho hlavní produkt. JDeveloper nelze brát jako klasické IDE. Spíše je potřeba na něj nahlížet jako na vnitřní vývojový prostředek Oraclu, který je dostupný každému, komu se může hodit [22].

## 4.5 BlueJ

BlueJ je bezplatné vývojové prostředí, které umožňuje snadno a rychle vyvíjet programy v jazyce Java. Je to jednoduché prostředí, které se výrazně liší od ostatních IDE na trhu. Vyvinut byl především pro výuku a vývoj malých projektů. BlueJ má záměrně menší a jednodušší rozhraní, takže začátečník není zahlcen množstvím nástrojů, které by prozatím stejně nevyužil. Jeho hlavní výhodou je, že je interaktivní. Umožňuje vizuálně vytvářet instance objektů, volat metody a kontrolovat jejich hodnoty, díky čemuž lze lépe pochopit objektové orientované programování (OOP) [27].

## 4.6 Android Studio

Android Studio je poměrně nové, volně dostupné IDE od Googlu. Vytvořené je na základech IntelliJ IDEA Community Edition, takže obsahuje řadu jeho nástrojů (navigace v kódu, analýza kódu, našeptávání, refaktoring a další). Jak napovídá název, je určené k vývoji aplikací pro mobilní systém Android. Dostupné je ve formách balíků pro různé operační systémy. Na stejném kódu

jako Android Studio je postaven i Android plugin pro IntelliJ IDEA, takže veškeré funkce a změny jsou dostupné i pro uživatele tohoto prostředí [28, 29].

## 4.7 JCreator

Prostředí JCreator bylo vytvořeno společností Xinox Software a je kompletně napsané v jazyce C++. Podle tvrzení výrobce je právě kvůli tomu rychlejší a produktivnější než jeho konkurence napsaná v jazyce Java. Je to jednoduché, ale zároveň výkonné prostředí vhodné pro začátečníky i profesionály. Dostupné je pouze pro operační systém Windows, a to ve dvou verzích. Bohužel obě tyto verze jsou placené (jejich ceny jsou uvedeny v Tabulce 4.2) [30].

Tabulka 4.2: Ceny produktů JCreator [30]

| **Produkt** | **Počet osob** | **Cena**[2] |
|---|---|---|
| JCreator LE | 1 | 35 $ |
| JCreator Pro | 1 | 79 $ |
| JCreator Pro (akademická licence) | 25 | 725 $ (29 $/osoba) |
| | Neomezený počet | 2500 $ |

## 4.8 JBuilder

JBuilder je vývojové prostředí původně vyvinuté společností Borland Corporation, později však převzaté společností Embarcadero Technologies. Jedná se o placené prostředí, které je dostupné ve dvou verzích, a to konkrétně ve verzi R2 Professional a R2 Enterprise. Obě verze podporují vývoj Java EE aplikací, umožňují modelování UML diagramů a obsahují nástroje pro sledování zatížení paměti a CPU. Verze Enterprise obsahuje navíc řadu rozšíření a umožňuje nasazení a údržbu uvnitř podniku. Obě verze jsou z roku 2008 a další vývoj tohoto prostředí se neplánuje. Ceny[3] obou verzí jsou uvedeny v Tabulce 4.3 [31].

[2]Cena se vztahuje vždy ke konkrétní verzi produktu. V případě, že chce uživatel přejít na vyšší verzi produktu, je mu k dispozici upgrade za nižší cenu.

[3]Informace o cenách získána z mailové komunikace se společností zastupující Embarcadero Technologies v České Republice.

Tabulka 4.3: Verze prostředí JBuilder

| **Verze** | **Cena** |
|---|---|
| JBuilder 2008 R2 Professional | 13 800 Kč (bez DPH) |
| JBuilder 2008 R2 Enterprise | 41 400 Kč (bez DPH) |

## 4.9 SnapCode

Jedná se o jednoduché, volně dostupné IDE vytvořené hlavně pro výuku. Umožňuje snadno a rychle vytvářet Java aplikace i úplným začátečníkům. Obsahuje totiž grafický editor a možnost pracovat s komponentami pouhým klikáním myši bez nutnosti psát zdrojový kód. Pro pokročilejší uživatele nabízí i klasický textový editor obsahující standardní nástroje jako ostatní IDE [32].

## 4.10 Navicoder

Navicoder je bezplatné IDE určené pouze pro nekomerční použití. Toto prostředí si zakládá na rychlé době odezvy, tudíž možnosti rychleji vytvářet a ladit kód programu. Naprogramované je v jazyku C++, díky čemuž je rychlejší než jiná IDE založená na jazyku Java. Dostupné je pouze pro operační systémy Windows [33].

## 4.11 jGRASP

jGRASP je bezplatné vývojové prostředí implementováno v jazyce Java, které lze spustit na všech operačních systémech s JVM. Vytvořené je speciálně za účelem automatického generování softwarových vizualizací, které umožňují začátečníkům lépe pochopit základy programování. Díky tomu jej lze využít jako výukový nástroj na středních či vysokých školách. jGRASP vytváří diagramy tříd či diagram složitosti kódu a umožňuje intuitivní zobrazování hodnot vícerozměrných polí, rozsáhlých binárních stromů, front apod. [34].

## 4.12 DrJava

DrJava je jednoduché vývojové prostředí určené především pro studenty, zároveň však obsahuje výkonné funkce pro pokročilejší uživatele. Jedná se o bezplatné prostředí aktivně vyvíjené skupinou JavaPLT group na Rice University. Dostupné je ke stažení jako spustitelná aplikace pro operační systémy Windows a Mac a jako `.jar` soubor pro jiné operační systémy [35].

## 4.13 Greenfoot

Greenfoot je dalším volně dostupným vývojovým prostředím spadající do kategorie výukových nástrojů pro jazyk Java. Začátečníkům umožňuje snadné vytváření aplikací s 2D grafikou, jako jsou vizuální simulace či interaktivní hry. Založen je na koncepci nástroje BlueJ, takže s ním má mnoho společného. Na rozdíl od něj ovšem obsahuje vlastní třídy `World` a `Actor`, jež jsou rodiči veškerých tříd vytvořených programátorem. Třída `World` (svět) slouží pro reprezentaci grafického plátna, ve kterém se děje veškerá vizualizace. Třída `Actor` (herec) slouží pro reprezentaci objektů, které se ve vytvářené aplikaci mohou vyskytnout. Greenfoot umožňuje interaktivní práci s vytvořenými objekty, díky čemuž mohou začátečníci lépe pochopit objektové programování. Nainstalovat jej lze na všech operačních systémech. Využívat lze i verzi portable (obsahuje vlastní JDK) pro spouštění aplikace z přenosných médií USB [36].

## 4.14 SkyIDE

SkyIDE je další vývojové prostředí umožňující vývoj projektů v jazyce Java. Rovněž podporuje i další jazyky jako C++, PHP, JavaScript a další. Jedná se o vývojové prostředí, které není zapotřebí instalovat, jelikož je ke stažení již jako spustitelná aplikace. Díky tomu lze jednoduše přenášet na přenosných discích a spouštět na jakémkoliv PC s operačním systémem Windows. SkyIDE není nijak náročné na paměť a výkon počítače, takže lze bez problémů spouštět i na starších PC sestavách. Poslední nalezená verze je ovšem z roku 2009, lze tedy předpokládat, že vývoj tohoto prostředí byl ukončen [37].

## 4.15 Asterix IDE

Asterix IDE je jednoduché vývojové prostředí podporující jazyky Java, HTML a C/C++. Nainstalováno může být na různých operačních systémech. Jedná se o poměrně nové IDE, jelikož první verze byla vydána v roce 2013. Otázkou ovšem je, zda vývoj tohoto prostředí aktivně pokračuje, neboť poslední nalezená verze je z května roku 2014 [38].

## 4.16 tIDE

Vývojové prostředí tIDE je jednoduchý, ale efektivní nástroj umožňující programování v jazyce Java. Jako u některých výše zmíněných IDE není k jeho používání nutná instalace. Díky tomu jej lze snadno používat na různých počítačích s různými operačními systémy. Poslední verze byla vydána v prosinci roku 2015, což značí, že vývoj tohoto prostředí aktivně pokračuje [39].

## 4.17 Zeus IDE

Zeus je vývojové prostředí podporující řadu programovacích jazyků, určené pro systémy Windows. Jedná se o placený nástroj nabízející 45 denní lhůtu na vyzkoušení zdarma. Základní cena tohoto produktu je 89,95 $. Po zaplacení obdrží uživatel licenční kód, který nikdy nevyprší. V případě, že vyjde nová verze, mohou být některé funkce registrovaným uživatelům dostupné jako bezplatný upgrade. Jinak mají nárok na slevu 45 % na zakoupení plnohodnotné verze. Ke stažení je dostupná i bezplatná verze Zeus Lite, která některou funkcionalitu postrádá. Ta je ovšem označována spíše jako editor než vývojové prostředí [40].

## 4.18 Jenuity

Jenuity je jednoduché, bezplatné vývojové prostředí určené pro středně pokročilé programátory. Není nijak náročné na počítačové zdroje, takže lze bez problémů využít i na starších počítačích s různými operačními systémy. Poprvé bylo Jeunity uvedeno na trh v roce 2008. Poslední nalezená verze je

z roku 2011, takže vývoj tohoto prostředí pravděpodobně již nepokračuje [41].

## 4.19 JotAzul

JotAzul je jednoduché vývojové prostředí určené pro operační systém Windows. Inspirováno je prostředím BlueJ, z čehož plyne, že je určené pro studenty a začínající programátory. Ke stažení je zdarma, ovšem poslední nalezená verze je z roku 2005, takže s velkou pravděpodobností nedokáže konkurovat jiným výukovým nástrojům, které se neustále vyvíjí [42].

## 4.20 Další aplikace

Tato práce se zaměřuje výhradně na vývojová prostředí vytvořená pro jazyk Java. Proto ve výše uvedených prostředích není obsažené Visual Studio, které v základní verzi tento jazyk nepodporuje. Je však možné doinstalovat plugin, který základní podporu Javy zajistí. Dále existují aplikace jako jEdit, JSource, Sublime a další, které jsou označovány spíše jako textové editory, ale vzhledem k tomu, jakou škálu funkcí v dnešní době nabízejí, se pomyslné hranici mezi editorem a IDE výrazně přibližují.

# 5 Popis porovnávaných prostředí

Z kapitoly 4 je zřejmé, že je možné vybírat z celé řady vývojových prostředí. Úkolem této práce je alespoň dvě z uvedených prostředí prozkoumat a navzájem porovnat. Nemá ovšem smysl navzájem porovnávat prostředí určená pro různé účely. Rozhodl jsem se proto podrobit průzkumu celkem čtyři IDE, a to dvě určené pro vývoj a dvě určené pro výuku začátečníků.

Z první kategorie jsem se rozhodl pro IDE Eclipse a Netbeans, a to z prostých důvodů. Obě jsou neustále aktivně vyvíjena a zdokonalována, tudíž má toto srovnání pro čtenáře větší informační hodnotu, než srovnání starších aplikací. Obě jsou rovněž dostupné zdarma všem, kteří mají zájem je pro svou práci použít. Dalším důvodem je jejich rozšiřitelnost, jelikož pro obě prostředí existuje velké množství pluginů. I když nebyl nalezen žádný oficiální žebříček, který by informoval o míře rozšířenosti jednotlivých IDE, patří tato dvě prostředí mezi nejpoužívanější a nejznámější.

Z kategorie výukových prostředí byla první volba jasná — BlueJ. Zde si troufnu říci, že jde o nejznámější výukové prostředí na trhu. Zvolit druhé prostředí bylo o něco složitější. Přestože zde bylo uvedeno několik prostředí, které se označují za výukové, tak některé z nich (např. jGRASP) nepůsobí jako jednoduché IDE pro začátečníky. Proto bylo zvoleno prostředí Greenfoot, které je sice vyvíjeno stejnou skupinou jako samotný BlueJ, ale přistupuje k výuce odlišným způsobem.

## 5.1 Eclipse

Projekt Eclipse původně vznikl v listopadu roku 2001 vytvořením konsorcia společností IBM, Borland, QNX Software Systems a dalších společností dodávajících software. Koncem roku 2003 tento projekt podporovalo více než 80 společností. Počátkem února roku 2004 došlo k reorganizaci na neziskovou organizaci Eclipse Foundation, která je v současnosti podporována jednotlivci a organizacemi napříč softwarovým průmyslem [43].

Původní konsorcium vzniklo poté, co společnost IBM uvolnila Eclipse platformu jako open-source. Ta je vydána pod licencí *Eclipse Public Licence*, což umožňuje komukoliv postavit na této platformě vlastní aplikaci. Mimo

stejnojmenného IDE je na Eclipse platformě postavena i řada komerčních a open-source aplikací [44, 45].

### 5.1.1 Instalace

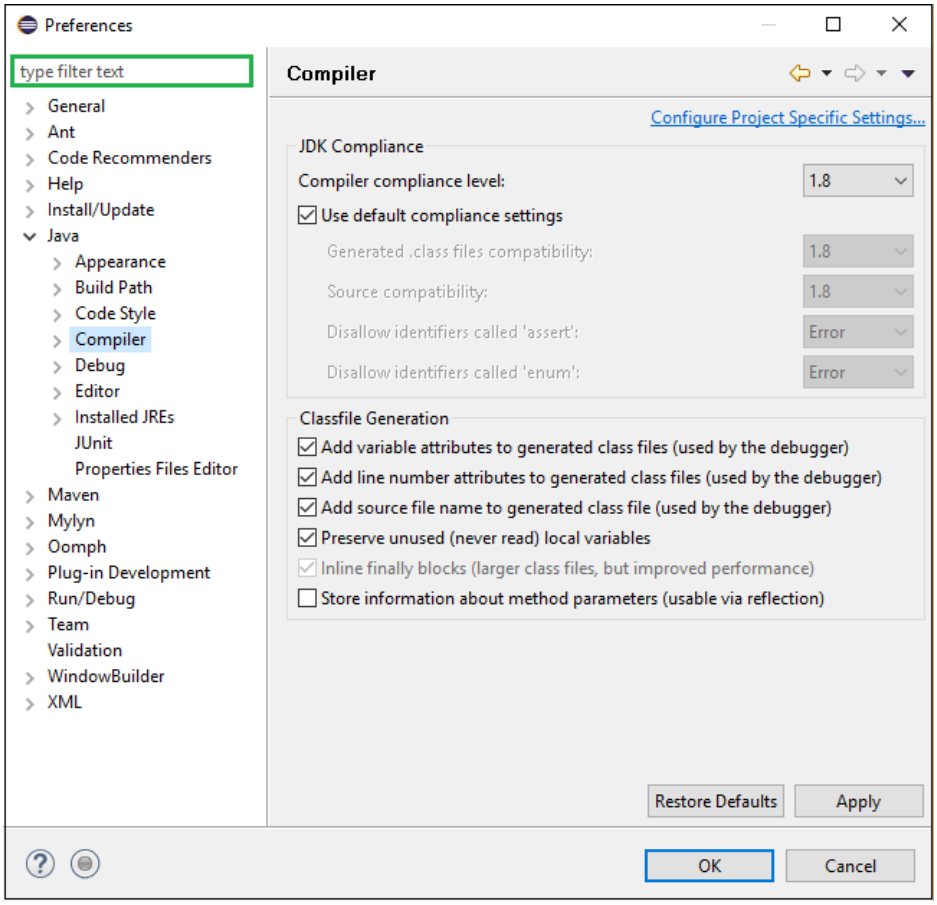
Nejjednodušší způsob instalace či updatu je využít předpřipravený *Installer*, který provede celou instalaci za uživatele. Ten musí pouze zvolit systém, na který se bude Eclipse instalovat, a balíček, který si přeje používat. Balíčky obsahují různé nástroje a technologie podle zaměření vývoje (Java, Java EE, C/C++, PHP a další).

Poslední verzí, kterou lze nainstalovat, je Eclipse Mars 2 (4.5.2 — vydána 12. 2. 2016), která je v této práci popisována.

### 5.1.2 Spuštění a konfigurace

Při spuštění aplikace je uživatel dotázán, kde chce vytvořit *workspace (pracovní prostor)*. Jedná se o místo na disku (složku), kam bude Eclipse ukládat všechny projekty, nastavení apod. Při dalším spuštění Eclipsu není nutné vybírat již existující workspace, ale je možné vytvořit další. Eclipse ovšem musí již při spuštění znát, kam bude ukládat jednotlivé projekty a soubory. Tento systém pravděpodobně používá pouze Eclipse, u ostatních prostředí lze projekty a soubory ukládat kamkoliv. Po zvolení workspace se již začnou načítat všechny moduly, které jsou zapotřebí k fungování aplikace. Celý proces spuštění trvá několik desítek vteřin v závislosti na konkrétním počítači a počtu načítaných pluginů.

V Eclipsu si programátor může nastavit prakticky všechno podle svých požadavků. Ať už jde o verzi Javy, formátování kódu, klávesové zkratky, debugger, generování kódu apod. Veškeré nastavení lze provést po otevření menu *Window ->Preferences*. Z Obrázku 5.1 je vidět, že je nastavení rozdělené do různých sekcí, které ještě obsahují další podsekce. Ovšem orientovat se v tom, kde se některá nastavení skrývají, nemusí být vůbec jednoduché. Je zde proto přidán i filtr, který zobrazí jen ty sekce, jež obsahují hledaný výraz. Tato nastavení se vztahují k celému vývojovému prostředí. Možné je i obdobným způsobem nastavit specifické vlastnosti vždy pro konkrétní projekt (*Project ->Properties*).

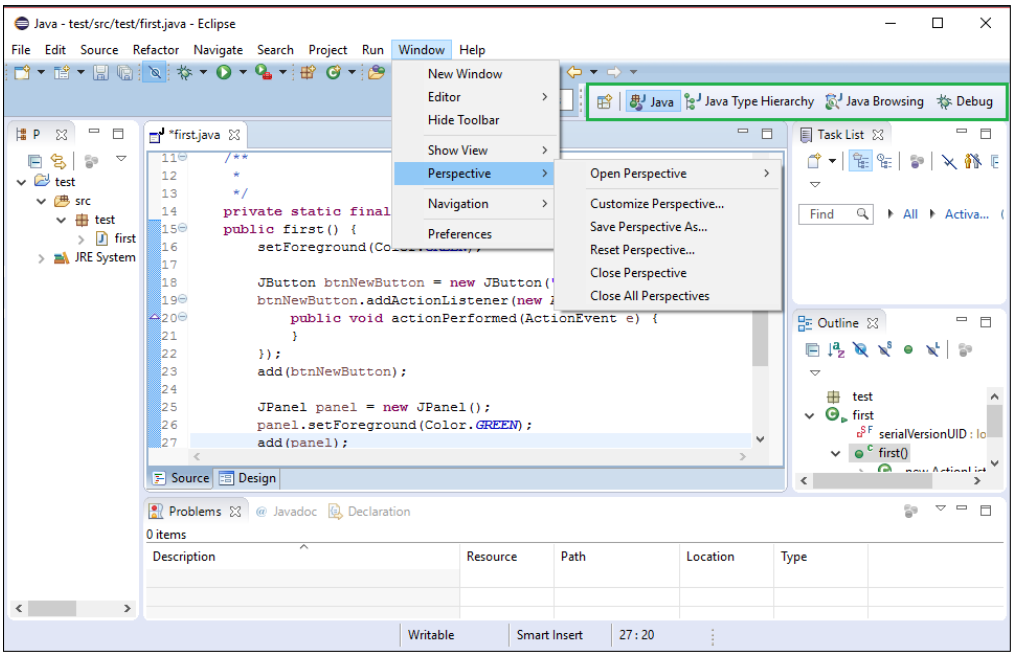



Obrázek 5.1: Eclipse: nastavení vlastností prostředí

Další věcí, kterou by programátor mohl nastavovat, je vyobrazení různých oken (pohledů), které si přeje při práci sledovat. Eclipse to za něj řeší možností přepínat mezi různými *perspektivami* podle toho, jakou činnost zrovna provádí (implementace kódu, ladění apod.). Každá perspektiva obsahuje seskupení různých oken, která mohou s činností souviset. Samozřejmě je možné si tyto perspektivy nastavit podle vlastních požadavků a uložit. Vše je patrné na Obrázku 5.2.

### 5.1.3 Generování kódu

Při vývoji programátor ocení jakoukoliv funkci, která mu usnadní a urychlí práci, a těch Eclipse nabízí celou řadu. Co se týče často se opakujících částí



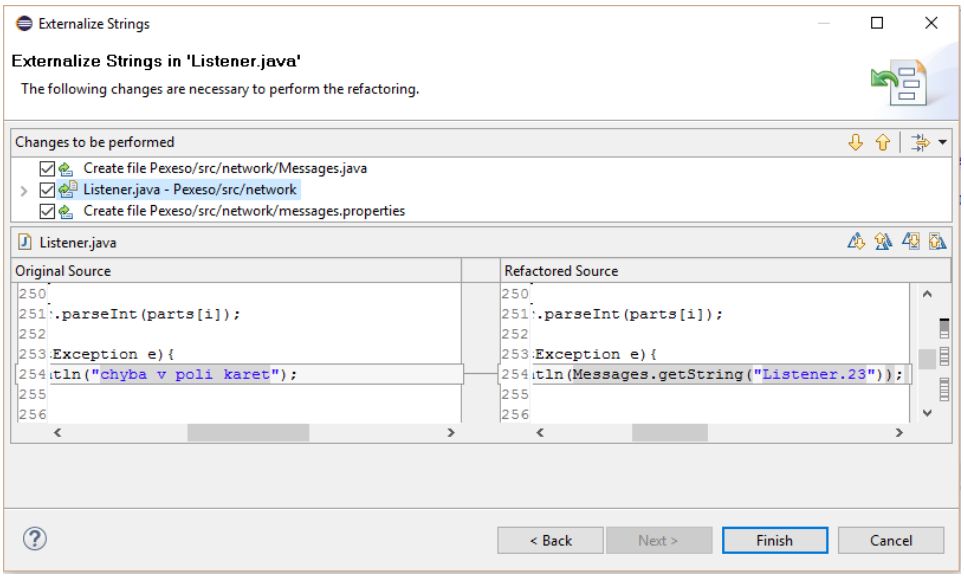

Obrázek 5.2: Eclipse: možnosti perspektiv

kódu, nechybí možnost generovat gettery/settery, konstruktory či dokumentační komentáře. Dále je možné využít funkci pro překrytí/implementování metod. Ta umožňuje zvolit metody (ať již se jedná o zděděné metody určené k překrytí či metody dané rozhraním), jejichž hlavičky budou doplněny do zdrojového kódu. Programátorovi pak stačí pouze doplnit těla těchto metod.

Za zmínku stojí i funkce *Organize imports* a *Clean up*. První zmíněná funkce odstraní z kódu již nepoužívané importy. Ta druhá zasahuje do kódu mnohem více, jelikož ho upravuje podle definovaných nastavení. Těmi může být odstranění nepoužívaných importů, odstranění nepoužívaných lokálních proměnných, doplnění chybějících `@Override` notací a další akce zvolené programátorem.

Poslední zajímavá funkcionalita se skrývá pod názvem *Externalize Strings*. Při jejím použití jsou veškeré řetězce (např. výpisy na obrazovku) v editovaném zdrojovém souboru nahrazeny proměnnými, které jsou definované v samostatném souboru. V definované balíku vzniknou rovnou dva nové soubory, konkrétně `xxx.properties` a `xxx.java`. První tedy obsahuje zmíněné definované proměnné, druhý obsahuje algoritmus k načítání konkrétních řetězců ze souboru. Takovýchto souborů může být v projektu více, záleží jen na tom, do kolika souborů si programátor řetězce rozdělí. Výhodou toho po-

užití je uložení řetězců na jednom místě, tudíž jejich jednodušší editace bez nutnosti procházet zdrojový kód. Jelikož se kód může často zásadně měnit, je zde přidána i možnost kontrolovat aktuálnost řetězců, tedy zda nějaký nechybí nebo nepřebývá. Kontrola se může provádět nad daným souborem, balíkem nebo celým projektem. Jak se po externalizaci změní samotný kód, je ukázáno na Obrázku 5.3.

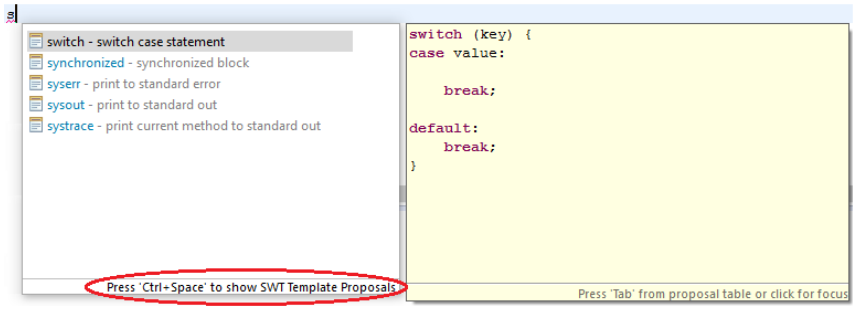



Obrázek 5.3: Eclipse: externalizace řetězců

### 5.1.4 Doplňování kódu

Automatické doplňování, nebo-li *Content Assist*, nabízí programátorovi k nahlédnutí části kódu, které zapadají do kontextu již napsaného výrazu. Standardně je nastaveno, aby se návrhy kódu zobrazovaly po napsání znaku „.“, ale je možné je zobrazit kdykoliv pomocí klávesové zkratky (standardně Ctrl+mezerník).

Při prvním vyvolání kontextové nápovědy nabízí Content Assist prakticky všechny možné návrhy, zapadající do předešlého kódu, jako jsou objekty a metody rozhraní API, definované šablony apod. Nápovědu lze ovšem procházet i cyklicky, a to opakovaným vyvoláním klávesové zkratky. Tím se zobrazované návrhy omezí pouze na ty, jež spadají do určité kategorie (např. definované šablony). Omezení nabízených návrhů vysvětluje Obrázek 5.4.




Obrázek 5.4: Eclipse: omezení nápovědy pouze na šablony

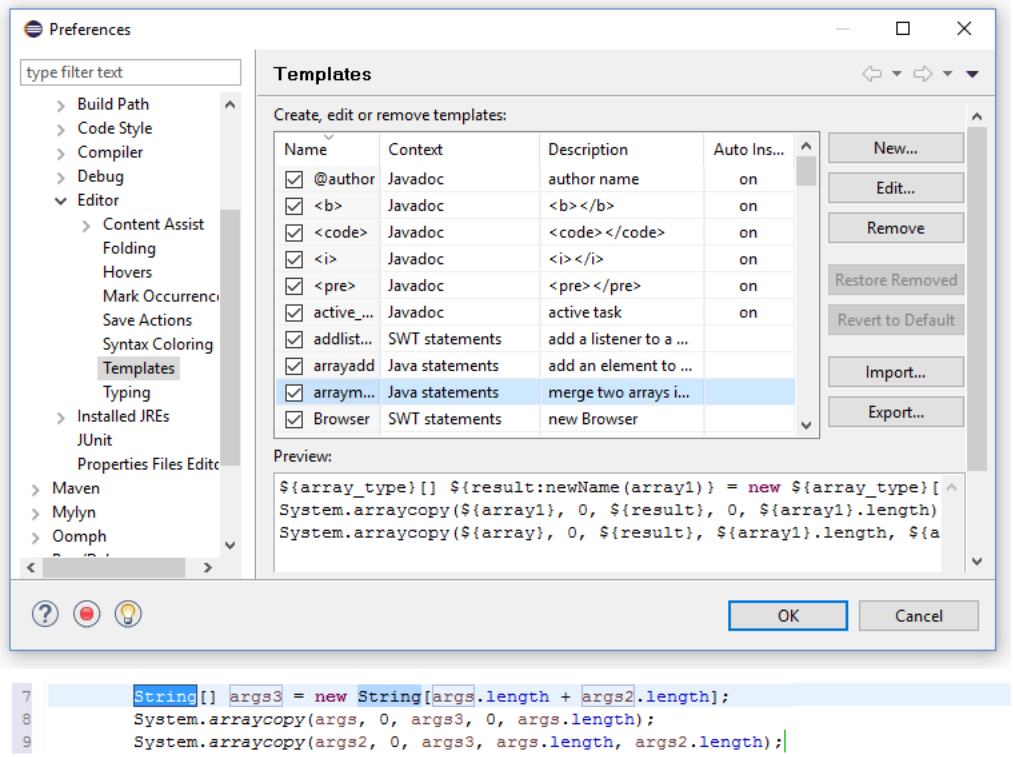
Výše zmíněné šablony umožňují programátorovi pod určitou slovní zkratku skrýt libovolný, často se opakující úsek kódu. Doplnění kódu se vyvolá stejnou klávesovou zkratkou jako kontextová nápověda. Pamatuje-li si tedy programátor zkratku daného kódu, stačí mu ji napsat a stisknout Ctrl+mezerník. Pokud je pod danou zkratkou uložen pouze jeden kód, je automaticky doplněn. V opačném případě je zobrazena klasická kontextová nápověda, která umožní vybrat konkrétní kód. Eclipse již obsahuje předdefinované šablony, lze je ovšem upravovat či vytvořit své vlastní. Příklad šablony a doplněný kód je uveden na Obrázku 5.5.

### 5.1.5 Refaktoring

Eclipse obsahuje řadu funkcí, jež umožňují provádět změny v celém projektu bez zdlouhavého upravování jednotlivých tříd. Mezi tyto funkce patří například přejmenování proměnných či metod, změny návratových typů, zavádění návrhových vzorů, přesouvání metod či tříd v rámci projektu apod. Přesouvat části kódu napříč projektem lze i v panelu zobrazujícím hierarchii projektu, a to jednoduchým přetažením např. třídy do jiného balíku. Veškeré závislosti se přitom v projektu sami aktualizují. Pokud během provádění změn dojde k nějakým problémům, je proces pozastaven a o vzniklých problémech je informován programátor.

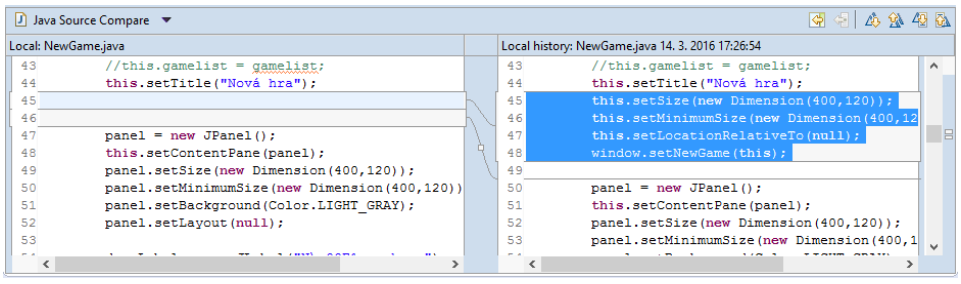
### 5.1.6 Historie kódu

Eclipse si udržuje *historii refaktoringu*, do které ukládá informace o tom, kdy a k jakým změnám v projektu došlo. Tedy například, že před několika dny byl v projektu změněn název metody. Změny ovšem není možné navracet.




Obrázek 5.5: Eclipse: kód šablony pro sloučení polí

Rovněž disponuje *lokální historií*. V té jsou uložené jednotlivé revize souborů daného projektu. Je tedy možné porovnat některou třídu s její starší revizí a případně části kódu či celou třídu obnovit. V případě, že by tedy programátor něco nedopatřením smazal, nemusí se bát, že by o svou práci přišel. Eclipse při porovnávání souborů viditelně označuje části kódu, které se liší a umožňuje je jednoduše sloučit (viz Obrázek 5.6).




Obrázek 5.6: Eclipse: porovnávání kódu s lokální historií

### 5.1.7 Další možnosti

Samozřejmostí je možnost využívat v prostředí debugger pro ladění aplikací. Jako velmi užitečné shledávám možnost přidávat podmínky k breakpointům a tím omezit pozastavení programu. Eclipse používá k buildování programů vlastní builder, možné je však využít i integrované nástroje Ant, Maven či Gradle. S buildováním souvisí i možnost vytvářet specifické běhové konfigurace. Ty lze spouštět i pro testování částí projektu, jelikož Eclipse umožňuje vytvářet a spouštět JUnit testy.

Pokud jde o verzovací systémy, Eclipse v sobě již obsahuje Egit (Git klient pro Eclipse). Formou pluginů je možné doinstalovat i další známé verzovací systémy jako Subversion (plugin Subversive) či Mercurial (plugin MercurialEclipse). V případě týmových projektů se může hodit i integrovaný nástroj Mylyn, umožnující správu úkolů v projektu. U menších projektů si lze jednoduše přidávat úkoly k jednotlivým řádkům kódu.

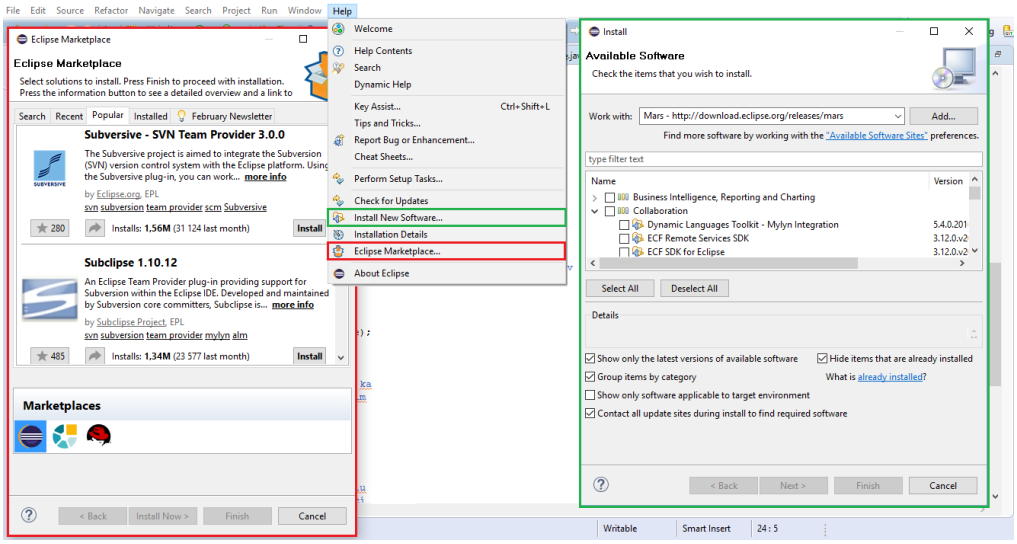
Jelikož je v této práci porovnáván Eclipse s IDE Netbeans, které již integruje GUI designer, nabízelo by se jeho srovnání s nějakým dostupným pluginem pro Eclipse. Těch je ovšem k dispozici více a jejich kvalita se může výrazně lišit, tudíž by srovnání s jedním z nich nebylo příliš objektivní.

### 5.1.8 Možnosti rozšíření

Jak už bylo jednou zmíněno, pluginů dostupných pro Eclipse je opravdu mnoho, takže lze doinstalovat téměř cokoliv. Vybírat lze ať již z pluginů vydaných organizací Eclipse či těch od externích vývojářů. Otázkou ovšem zůstává, kolik těchto rozšíření je opravdu kvalitních a aktuálních.

Rozšíření lze přímo v IDE doplnit dvěma způsoby. První možností je využití MarketPlace klienta, což je rozhraní umožňující procházet webové stránky (dalo by se říci databázi) s uloženými rozšířeními pro Eclipse. Pluginy jsou rozděleny do určitých kategorií. Rovněž je zde o nich uveden krátký popis včetně počtu jejich stažení. Druhý způsob je použití funkce „Instalovat nový software“. Zde se vyhledávají rozšíření na základě konkrétní URL adresy úložiště Eclipse. V případě, že ji programátor zná, je toto použití opravdu jednoduché. V opačném případě může být dohledání konkrétního pluginu poměrně složité. Hlavně vypsání všech rozšíření ze všech již uložených adres úložiště může zabrat velmi dlouhou dobu (klidně i několik minut).

Co všechno prostředí obsahuje, je možné vidět v detailech instalace. Zde je možné pluginy i odinstalovat. Eclipse si vede historii konfigurací, do které přidává záznam v okamžiku, kdy dojde v prostředí k instalaci respektive odinstalaci některého pluginu. Tuto sekci týkající se pluginů shrnují Obrázky 5.7 a 5.8.



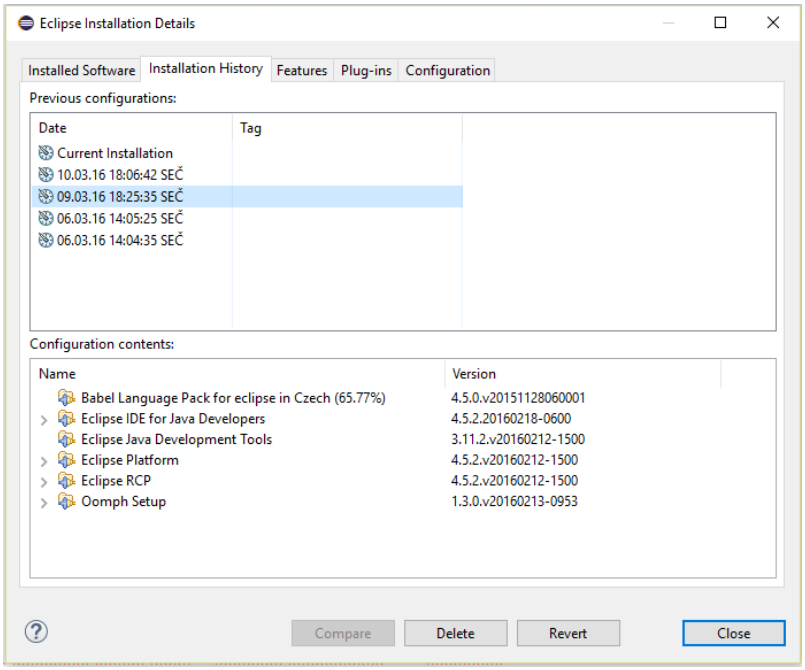

Obrázek 5.7: Eclipse: možnosti doinstalování rozšíření

### 5.1.9 Dokumentace a zdroje informací

Eclipse disponuje opravdu zdařilou dokumentací, která je dostupná nejen na domovských stránkách[1], ale rovněž integrována v samotném prostředí. Možné je ji otevřít v dalším okně aplikace nebo ji připnout jako postranní panel do prostředí. V prvním případě je dokumentace mnohem lépe čitelná, v druhém případě ji má programátor neustále na očích, záleží tedy jen na něm jaký způsob si zvolí. Obě možnosti jsou vyobrazené na Obrázku 5.9.

Dokumentace je rozdělena do různých sekcí, jež velmi podrobně popisují jednotlivé funkce a možnosti prostředí či obsahují různé tutoriály. Za povšimnutí stojí i sekce *What's new* informující o změnách a novinkách v používané verzi IDE. Samozřejmostí je možnost vyhledávání, které odkazuje do jednotlivých sekcí dokumentace.

[1] https://eclipse.org

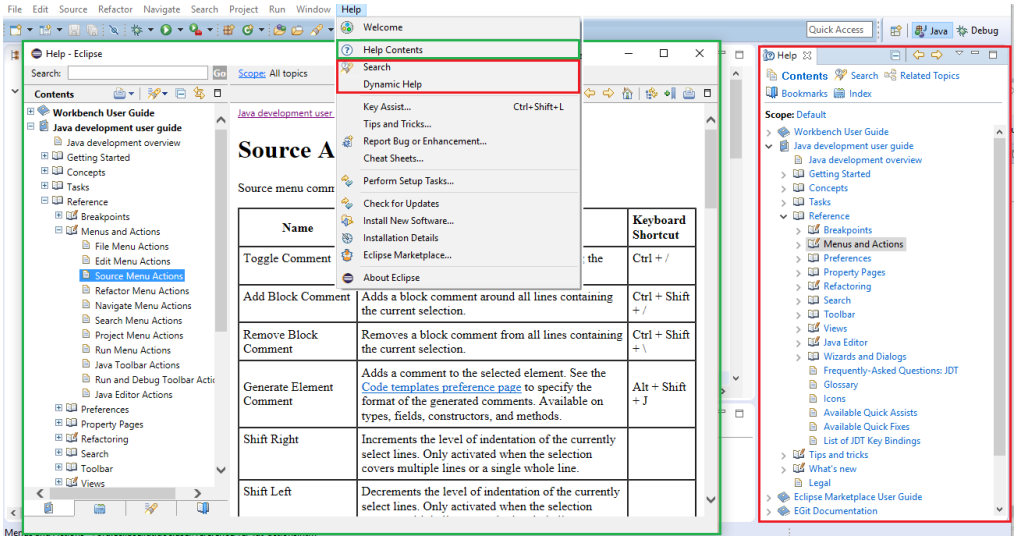



Obrázek 5.8: Eclipse: historie instalovaných pluginů

V případě, že by někomu dokumentace nestačila, je možné se obrátit na další zdroje informací. Z knižních publikací je možné využít například starší publikaci [46] či novější a obsáhlejší publikaci věnující se verzím 4.2 a 4.3 [47]. Publikací je celá řada, ale zdá se, že dosud žádná nebyla vydána v českém jazyce. Pro jejich čtení je tedy nutná pokročilá znalost angličtiny.

V dnešní době ovšem spatřuji jako hlavní zdroj informací internet a to hlavně z důvodu aktuálnosti. Odpovědi na otázky lze proto hledat na oficiálních stránkách organizace Eclipse, které odkazují na komunitní fórum, *Eclipsepedii*, video kanál, novinky apod. Vzhledem k rozšířenosti Eclipsu je velmi pravděpodobné, že požadované informace nalezneme i na dalších programátorských fórech[2].

[2]Příkladem může být velmi populární www.stackoverflow.com.




Obrázek 5.9: Eclipse: nápověda v prostředí

## 5.2 Netbeans

V roce 1996 vznikl v České Republice studentský projekt Xelfi, jenž měl za cíl vyvinout vývojové prostředí pro jazyk Java, podobné IDE Delphi. Xelfi bylo vůbec první Java vývojové prostředí napsané v Javě. Později byl název změněn na Netbeans. Netbeans byl původně vydáván jako komerční produkt. V roce 1999 však Netbeans získala (prostřednictvím nákupu aktiv) společnost Sun Microsystems a v následujícím roce jej uvolnila jako open-source. V roce 2010 Oracle získal společnost Sun (s tím i Netbeans) a rozhodl se jej nadále sponzorovat. Dokonce jej považuje za oficiální IDE pro jazyk Java [48].

Stejně jako u Eclipse je k dispozici rovněž open-source platforma Netbeans (RCP). Ta může být základem různých desktopových Swing aplikací. Jelikož je sama platforma modulární, umožňuje vytvářet robustní a rozšiřitelné aplikace. Webové stránky organizace se snaží udržovat přehled aplikací postavených na Netbeans platformě. Ten v současné době čítá stovky aplikací [49, 50].

Distribuce Netbeans jsou vydávány pod licencemi *Common Development and Distribution License 1.0* a *GNU General Public License v2*. Některá rozšíření však mohou podléhat licencím třetích stran [51].

### 5.2.1 Instalace

Před instalací si musí uživatel nejdříve zvolit, jaký z předpřipravených balíků bude využívat. Které balíky jsou k dispozici je patrné z Obrázku 5.10. Pro Javu to jsou balíky *Java SE*, *Java EE* či případně balík *All*, který shrnuje technologie všech balíků a navíc umožňuje i vývoj mobilních aplikací. Netbeans je možné stáhnout s již přidaným modulem pro překlad prostředí do jiného jazyka, ovšem v případě češtiny je stejně jako u pluginu pro Eclipse přeložena jen určitá část.

Poslední verzí, kterou lze nainstalovat, je Netbeans 8.1 (vydána 4. 11. 2015), která je v této práci popisována.

NetBeans IDE Download Bundles in community contributed languages[1]

| Supported technologies * | Java SE | Java EE | HTML5/JavaScript | PHP | C/C++ | All |
|---|---|---|---|---|---|---|
| NetBeans Platform SDK | • | • | | | | • |
| Java SE | • | • | | | | • |
| Java FX | • | • | | | | • |
| Java EE | | • | | | | • |
| Java ME | | | | | | • |
| HTML5/JavaScript | | • | • | • | | • |
| PHP | | | • | • | | • |
| C/C++ | | | | | • | • |
| Groovy | | | | | | • |
| Java Card™ 3 Connected | | | | | | • |
| Bundled servers | | | | | | |
| GlassFish Server Open Source Edition 4.1.1 | | • | | | | • |
| Apache Tomcat 8.0.27 | | • | | | | • |

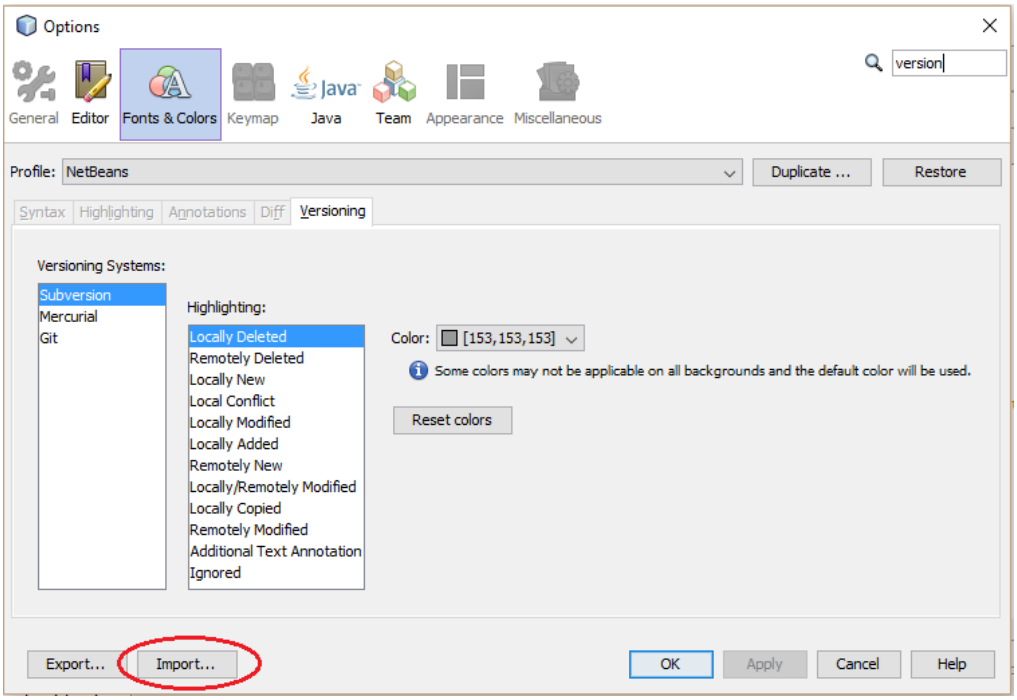
Obrázek 5.10: Netbeans: instalační balíky [52]

### 5.2.2 Spuštění a konfigurace

Pojem workspace u IDE Netbeans úplně odpadá, jelikož je možné ukládat projekty kamkoliv. Proces spuštění aplikace rovněž trvá několik desítek vteřin, jako je tomu u prostředí Eclipse. Konkrétní čas nelze uvést, jelikož je závislý na konkrétním počítači a počtu načítaných modulů, který se samozřejmě může lišit.

Co se týče konfigurace, i Netbeans umožňuje nastavit prostředí podle vlastních požadavků. Možnosti nastavení jsou poněkud omezenější než je tomu u Eclipsu, ovšem nejobvyklejší nastavení (doplňování kódu, klávesové zkratky apod.) pochopitelně nechybí. Okno pro nastavení působí velmi přehledně, jelikož jednotlivá nastavení nejsou schovaná ve složité stromové struktuře.

I přesto nechybí vyhledávač, jenž zvýrazní pouze ty sekce, které obsahují hledaný výraz. Velmi užitečná může být možnost importovat konfiguraci z již existujícího `.zip` archivu. Vše je vyobrazeno na Obrázku 5.11.

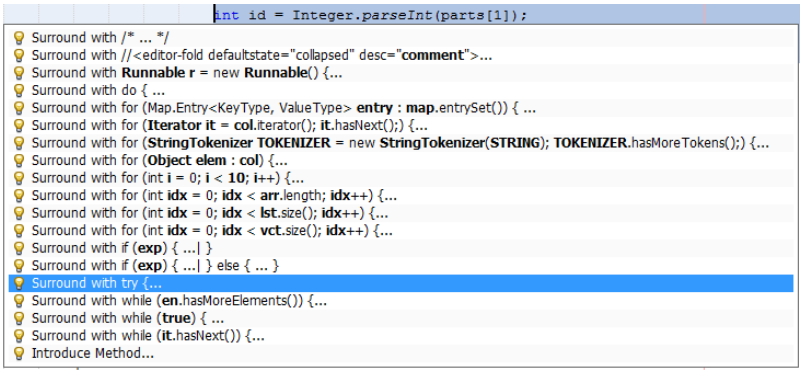



Obrázek 5.11: Netbeans:konfigurace prostředí

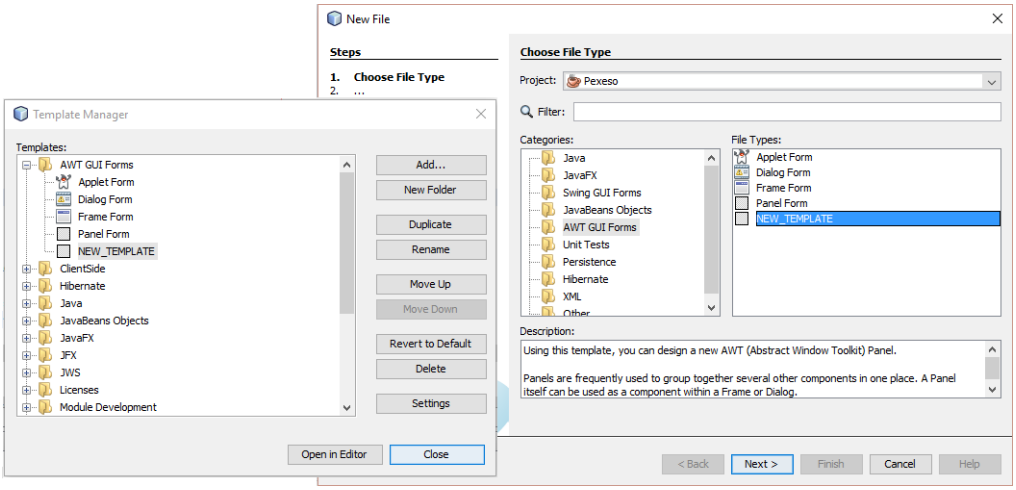
### 5.2.3 Generování kódu

Možnosti generování kódu jsou prakticky stejné jako u IDE Eclipse. Lze tedy generovat gettery/settery a konstruktory, provádět správu importů, formátovat kód apod. Nechybí ani možnost „obalit“ označený kód kódovými bloky (viz Obrázek 5.12). I Netbeans umožňuje přesunout řetězce do externího souboru `.properties`. Z tohoto souboru se poté načítají využitím třídy *ResourceBundle*.

Při vytváření nového souboru programátor ocení celou řadu šablon, jež definují kód, který bude vygenerován v nově vzniklém souboru. Tyto šablony lze snadno editovat či vytvářet své vlastní. Použití vlastní šablony při vytváření souboru ukazuje Obrázek 5.13.




Obrázek 5.12: Netbeans: možnosti doplnění kódových bloků




Obrázek 5.13: Netbeans: vytvoření souboru z vlastní šablony

### 5.2.4 Doplňování kódu

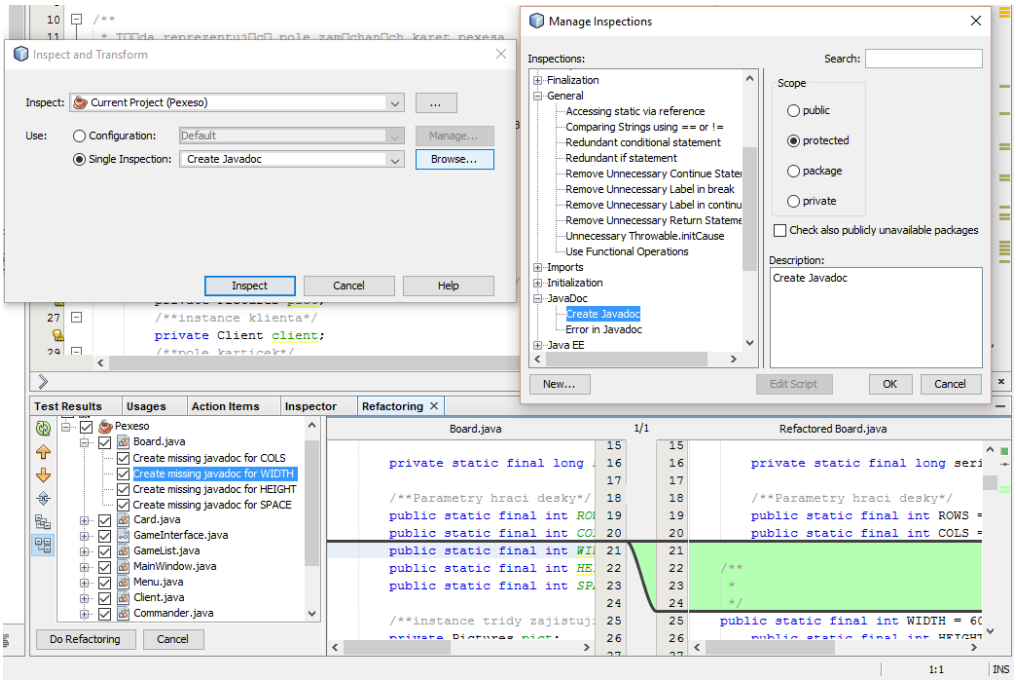
Používání kontextové nápovědy je téměř stejné jako v prostředí Eclipse. Pro její zobrazení je nastavena stejná klávesová zkratka a rovněž se zobrazuje po napsání znaku „.“. Pochopitelně je možné si uložit vlastní nastavení. V čem se ovšem kontextová nápověda liší je její procházení. Při prvním vyvolání jsou zobrazeny návrhy, které Netbeans považuje za relevantní vzhledem k dosud napsanému kódu. Při druhém vyvolání klávesové zkratky již zobrazí všechny návrhy, které je možné doplnit vzhledem k napsanému výrazu. Není tedy možné cyklicky přepínat nápovědu a omezovat tak návrhy podle konkrétních kategorií, jako je tomu u Eclipsu.

Možnost vytvářet a používat šablony samozřejmě nechybí. Co chybí, je již

zmíněná možnost omezit kontextovou nápovědu pouze na definované šablony. Je tedy nutné si pamatovat konkrétní zkratku, pod kterou se hledaný kód může skrývat.

### 5.2.5 Refaktoring a analýza kódu

Refaktoring v prostředí Netbeans umožňuje provádět téměř stejné operace, jako je tomu u prostředí Eclipse. Jedinečná je zde možnost provádět kontrolu (využitím statické analýzy kódu) a případně změny nad souborem, balíkem či celým projektem. Pod kontrolou je možné si představit například dohledání chybějící dokumentace, jak je ukázáno na Obrázku 5.14. Nalezené problémy jsou nejprve pouze vypsány a veškeré změny musí potvrdit programátor.




Obrázek 5.14: Netbeans: statická analýza kódu a refaktoring

### 5.2.6 Historie kódu

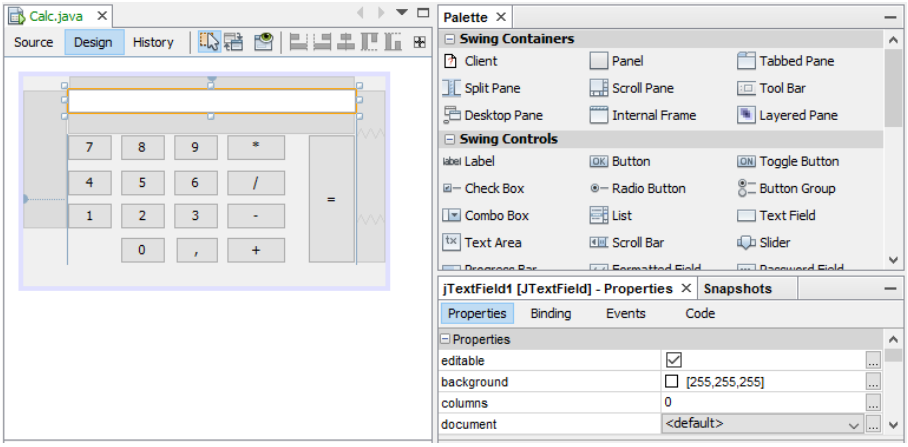
I Netbeans si vede *lokální historii* revizí jednotlivých souborů. S tou souvisí i nástroj pro porovnávání dvou souborů, který je s historií spojený. Ten ba-

revně odlišuje části kódu, které se liší a umožňuje jednoduše provádět změny v souboru. Prakticky se jedná o stejnou funkcionalitu jako obsahuje Eclipse. Nástroj pro porovnání je pochopitelně možné využít i pro dva různé soubory.

Co Netbeansu schází, je *historie refaktoringu*, ve které by bylo možné nalézt změny provedené nad celým projektem. K dispozici je pouze historie smazaných souborů a možnost je v projektu obnovit.

### 5.2.7 GUI builder

Důležitou součástí prostředí je již vestavěný grafický builder. Ten umožňuje jednoduše vytvářet grafické rozhraní bez nutnosti psát veškerý zdrojový kód. Stačí pouze vybírat prvky z předpřipravené „palety“, která obsahuje různé AWT a Swing komponenty a vkládat je na plátno. Kód se při tom generuje automaticky podle toho, jaké vlastnosti se přidaným prvkům nastaví. Netbeans se rovněž automaticky stará o zarovnání přidaných komponent, což je možné vidět na Obrázku 5.15.



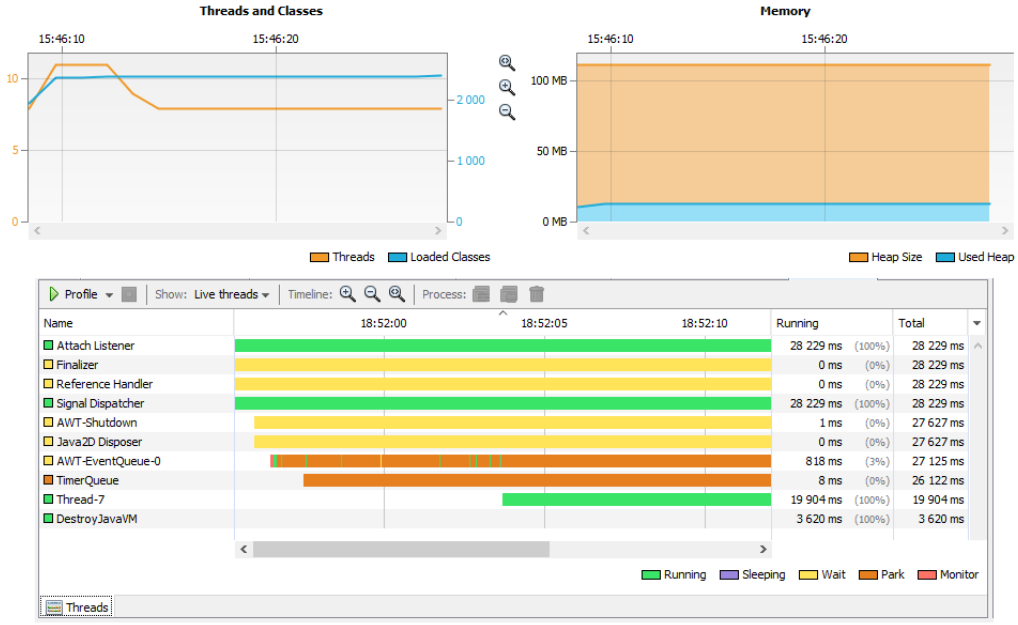

Obrázek 5.15: Netbeans: integrovaný GUI builder

### 5.2.8 Další možnosti

Samozřejmostí je možnost využívat v prostředí debugger pro ladění aplikací. Zde oceňuji možnost pořizovat snímky GUI aplikací a zobrazovat informace o jednotlivých komponentách na snímku. To může být užitečné nejen při

ladění, ale například i při vytváření uživatelských příruček pro vytvořenou aplikaci.

Pro testování jednotek je zde přidána podpora nejen pro framework JUnit, ale rovněž i pro TestNG. S testováním souvisí i možnost profilování, tedy možnost provádět za běhu programu dynamickou analýzu, která umožní sledovat například využití paměti, vytížení CPU, počet běžících vláken, dobu provádění jednotlivých metod a mnoho dalšího. Příklad výstupu dynamické analýzy ukazuje Obrázek 5.16.



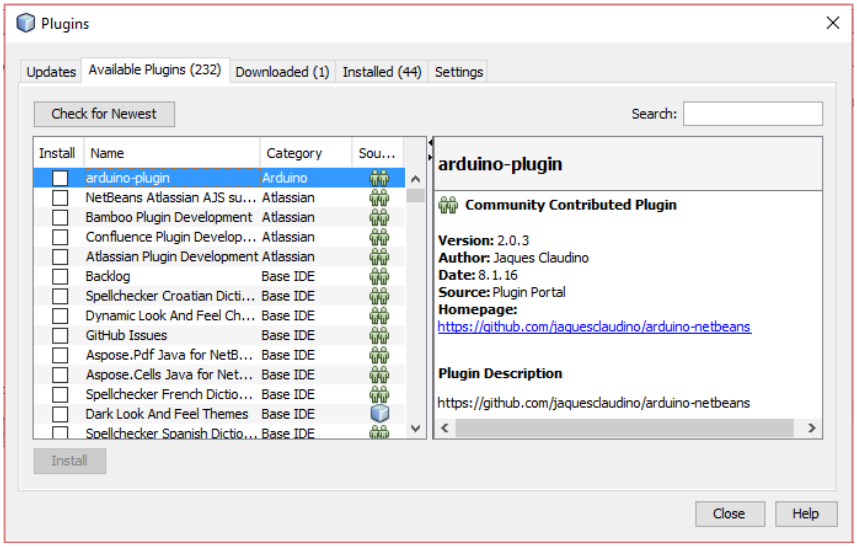

Obrázek 5.16: Netbeans: integrovaný profiler

K sestavení programů je standardně využíván Ant builder. Možné je ovšem využít i nástroj Maven, který je již integrován v prostředí. K dispozici je rovněž plugin pro buildovací nástroj Gradle. Možnost vytvářet vlastní běhové konfigurace pochopitelně nechybí.

Pro podporu týmových projektů Netbeans obsahuje verzovací systémy Subversion, Git a Mercurial. Zde popisovaná verze 8.1 lze rozšířit ještě o systém CVS. Starší verze bylo možné rozšířit i o systém ClearCase. Pro současnou verzi plugin dosud k dispozici není [53].

### 5.2.9 Možnosti rozšíření

Jak ukazuje Obrázek 5.17, i Netbeans umožňuje provádět správu pluginů přímo v prostředí. Vybírat zde lze pouze z pluginů, které jsou dostupné pro používanou verzi IDE. Teoreticky je možné doinstalovat i rozšíření, která nejsou pro používanou verzi určena, a to jejich dohledáním a stažením z Netbeans plugin portálu a následnou instalací v sekci *Downloaded*. Rozšíření ovšem nemusí pracovat správně a může docházet k nežádoucímu chování prostředí.



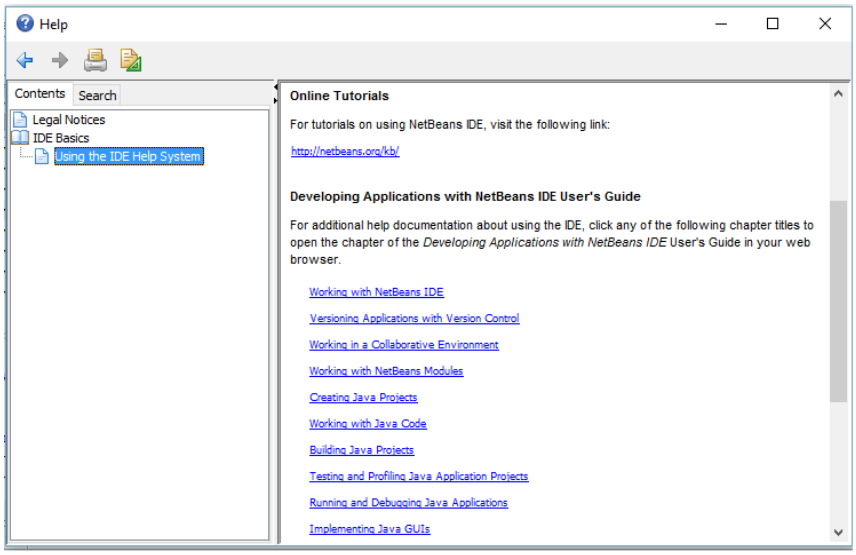

Obrázek 5.17: Netbeans: správa pluginů v prostředí

Výše zmíněný plugin portál je online databáze, kterou lze prohlížet na domovských stránkách Netbeans. Vyhledávání pluginů je možné filtrovat podle různých kategorií či verzí IDE, pro které jsou určené. V současné době (20. 3. 2016) databáze čítá 975 různých rozšíření [54].

### 5.2.10 Dokumentace a zdroje informací

Při prvotním pohledu na okno nápovědy může její obsah působit poněkud stroze. Obsahuje totiž pouze odkazy na uživatelskou příručku Oraclu dostupnou online. Ta velmi podrobně popisuje možnosti práce s vývojovým prostředím, ovšem nebude ji možné využít v případě nedostupnosti internetového připojení. K dispozici je i dokumentace offline, integrovaná přímo v prostředí.

Postrádám v ní ovšem rozdělení do nějakých logických sekcí, které by bylo možné procházet. Jediný způsob, jak ji využít, je tedy použití vyhledávače, jenž podle hledaného slova zobrazí relevantní odkazy do dokumentace. Nápověda v prostředí Netbeans je zobrazena na Obrázcích 5.18 a 5.19.

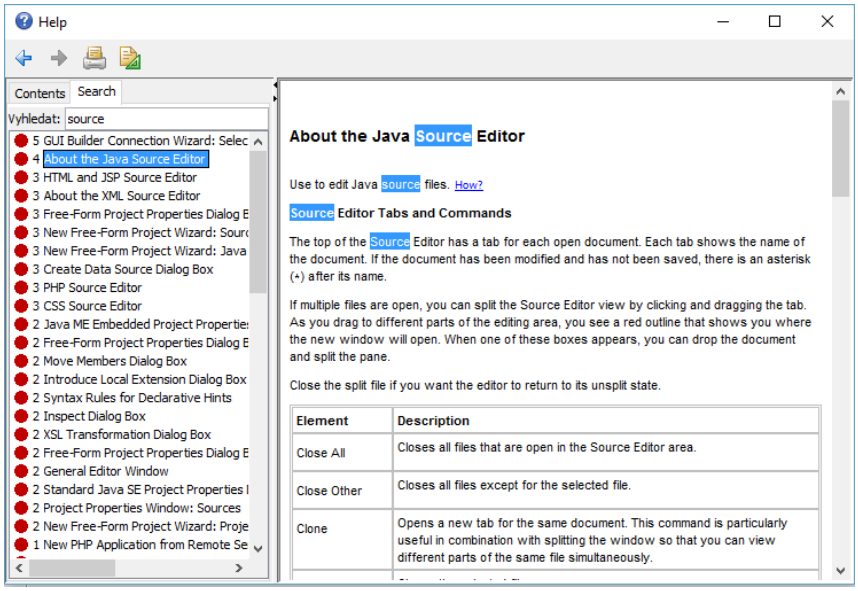



Obrázek 5.18: Netbeans: nápověda v prostředí

Prakticky jakékoliv informace lze nalézt na oficiálních stránkách[3] Netbeans. Ty obsahují odkazy na různé tutoriály, videa, dokumentace k jednotlivým verzím, komunitní fórum, *FAQ* a mnoho dalšího. Zajímavé jsou i tzv. „Learning trails“, které shromažďují veškeré materiály týkající se konkrétního vývoje, jako např. Java EE, Java SE apod.

Knižních publikací o Netbeans, ať již platformě nebo samotném vývojovém prostředí, existuje celá řada. Z těch popisujících vývojové prostředí lze zmínit publikaci [55], která provádí čtenáře začátkem používání tohoto IDE. Druhá publikace, o které se zmíním, se věnuje API platformy Netbeans a provází čtenáře vývojem *Rich Client Aplikací*. Tuto knihu uvádím hlavně proto, jelikož byla komunitou Netbeans přeložena do českého jazyka [56].

[3] https://netbeans.org




Obrázek 5.19: Netbeans: vyhledávání v nápovědě

## 5.3 BlueJ

Na konci 90. let vytvořil Michael Kölling jako součást své disertační práce na australské univerzitě Monash výukový, objektově orientovaný jazyk a zároveň vývojové prostředí zvané Blue. Svou koncepcí se jazyk Blue velmi podobal jazyku C++. Výukové prostředí bylo určeno především pro výuku studentů prvních ročníků. Vývoj ovšem dále nepokračoval, jelikož se vývojáři zaměřili na vytvoření stejného vývojové prostředí jako Blue, ovšem pro Javu — tedy BlueJ. V roce 1999 byla vydána jeho první verze a od té doby (hlavně díky velké podpoře Sun Microsystems a nyní Oraclu) jeho vývoj nadále pokračuje. Vývojový tým se rozšířil o další členy a v současnosti celý tým pracuje na univerzitě Kent ve Velké Británii [27, 57].

### 5.3.1 Instalace

BlueJ je dostupný pro různé operační systémy, ať již jako předpřipravený *installer* pro systémy Windows a Mac, balík `.deb` pro Ubuntu/Debian Linux či spustitelný `.jar` soubor pro ostatní operační systémy. Využít lze i možnost používat přenosnou verzi BlueJ spustitelnou z disků USB. Součástí instalačních balíků je i několik ukázkových projektů, které mohou usnadnit začátky

s objektově orientovaným programováním.

Poslední verzí, kterou lze nainstalovat, je BlueJ 3.1.7 (vydána 23. 2. 2016), která je v této práci rovněž popisována.

### 5.3.2 Spuštění a konfigurace

Prostředí se spustí prakticky okamžitě, jelikož se nenačítají žádné rozsáhlé moduly, jako je tomu u profesionálních prostředí. V jednom okně aplikace lze otevřít pouze jeden projekt. Existuje ovšem rozšíření, které umožňuje práci s více projekty najednou. Samotné projekty nedodržují nějakou striktní strukturu rozdělení souborů do jednotlivých adresářů. Adresáře jsou použity pouze pro dokumentaci, balíky či případně přidaná rozšíření, která budou vysvětlena v sekci 5.3.10.

V konfiguraci prostředí toho není příliš mnoho co nastavovat kromě základních nastavení editoru či klávesových zkratek. Důležitá je ovšem možnost volby použitého jazyka v prostředí, kdy je možné vybírat z 18 jazyků včetně češtiny.

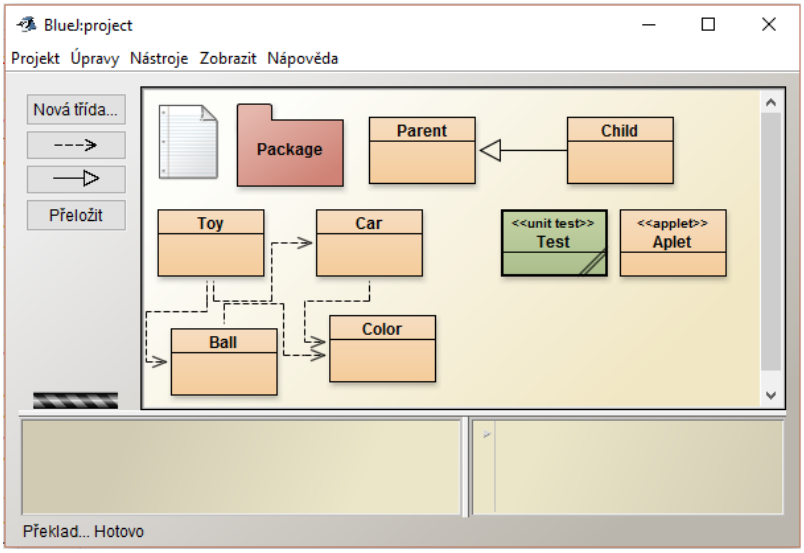
### 5.3.3 Popis prostředí

Že je BlueJ jednoduché prostředí, je vidět již při prvotním pohledu na uživatelské rozhraní. Nenachází se zde žádné rozsáhlé panely tlačítek ani seskupení pohledů. V něčem je ovšem jednoduchý až příliš. Nástroje jako je editor či debugger se spouští jako další okno aplikace. Editor navíc může obsahovat pouze jednu třídu, takže v případě potřeby editovat více tříd zároveň je nutné mít spuštěno několik oken, což velmi komplikuje práci s prostředím. Práci s prostředím lze prakticky rozdělit do několika částí, které jsou popsány níže.

### 5.3.4 Diagram tříd

Třídy jsou v prostředí reprezentovány jako ikony, a to různé pro klasickou třídu, rozhraní, testovací třídu apod. Jak je vidět na Obrázku 5.20, třídy mezi sebou mohou mít určité vazby. Plná čára mezi třídami reprezentuje dědičnost, přerušovaná čára s trojúhelníkovou šipkou označuje implementaci

rozhraní a přerušovaná čára s jednoduchou šipkou slouží pro zobrazení nějaké funkční závislosti. Celkově toto zobrazení působí jako diagram tříd daného projektu včetně závislostí mezi nimi.



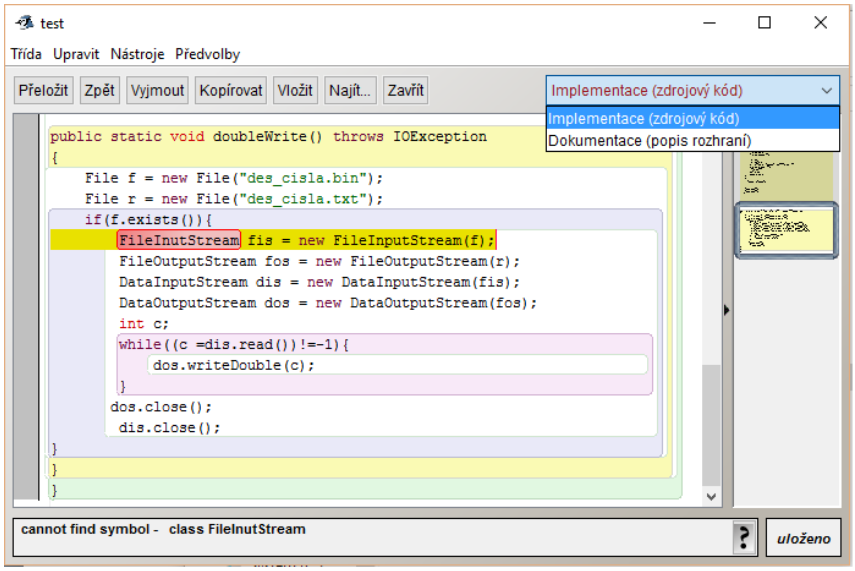

Obrázek 5.20: BlueJ: diagram tříd

### 5.3.5 Implementace kódu

BlueJ se snaží udržet v kódu přehlednost, proto barevně odlišuje bloky kódu jako jsou metody, cykly apod. Rovněž nabízí možnost automaticky zformátovat napsaný kód či doplnit dokumentační komentář. S tím souvisí možnost přepnout pohled právě na dokumentaci editované třídy. Obsažená je i jednoduchá kontextová nápověda. Chyby v kódu se odhalí až po překladu dané třídy. Kompilátor odhalí pouze první chybu, na kterou při překladu narazí. Řádek, ve kterém se chyba vyskytuje barevně označí a vypíše chybovou zprávu. Výše popsané vlastnosti editoru jsou vyobrazeny na Obrázku 5.21.

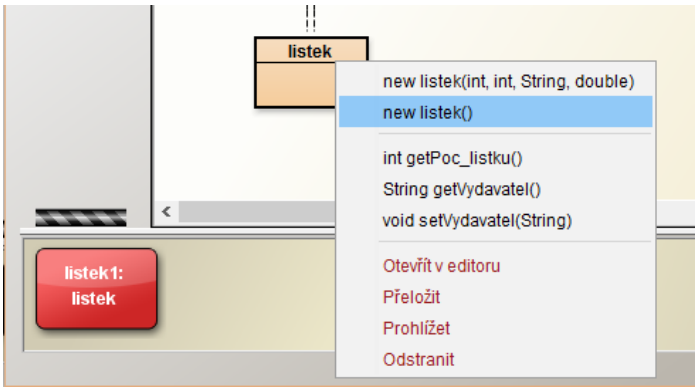
### 5.3.6 Práce s objekty

V prostředí je kladen důraz na interakci s uživatelem, proto lze spouštět statické metody třídy nebo vytvářet instance objektů bez nutnosti použití metody `main()`. Vytvořené instance objektů jsou zobrazeny jako vizuální




Obrázek 5.21: BlueJ: editor obsažený v prostředí

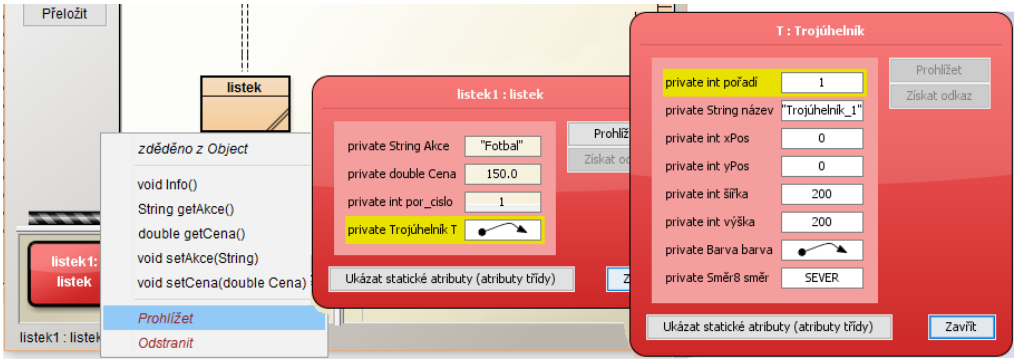
prvky, se kterými je možné dále pracovat, tedy spouštět jejich metody či prohlížet hodnoty proměnných. Práci s objekty shrnují Obrázky 5.22 a 5.23.




Obrázek 5.22: BlueJ: vytvoření instance objektu

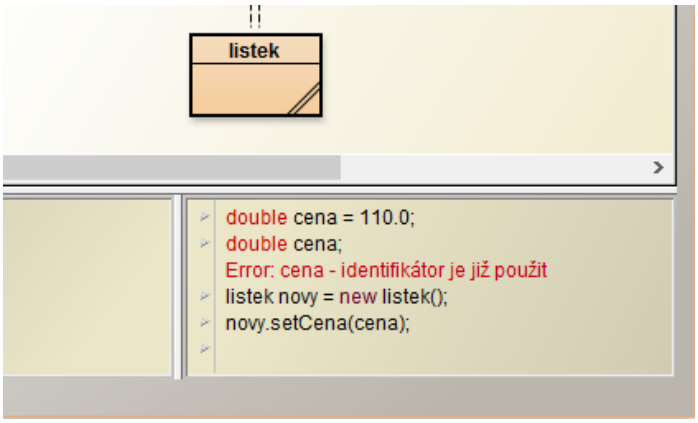
### 5.3.7 Příkazová řádka

Příkazová řádka umožňuje zaznamenávat příkazy uživatele a postupně je provádět. Je možné zde vytvářet proměnné, instance objektů a volat jejich




Obrázek 5.23: BlueJ: prohlížení proměnných instance

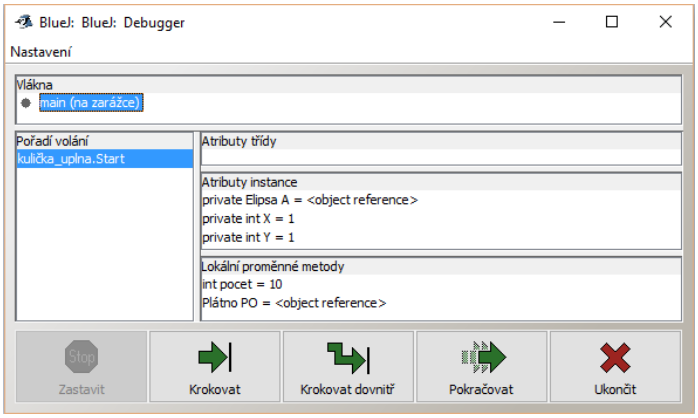
metody, vyvolávat statické metody třídy apod. V případě chybného zápisu je pouze zobrazeno varovné hlášení a lze dále pokračovat v zápisu. Příklad použití příkazové řádky ukazuje Obrázek 5.24. Ve standardním nastavení není příkazová řádka zobrazena, ale to je možné změnit v nastavení prostředí.



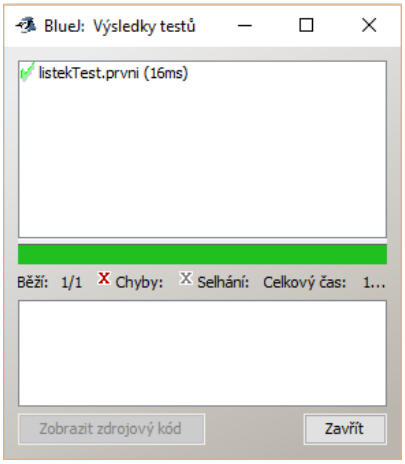

Obrázek 5.24: BlueJ: použití příkazové řádky

### 5.3.8 Ladění a testování

V prostředí pochopitelně nechybí debugger pro ladění programů. Ten umožňuje jednoduše krokovat program a při tom sledovat hodnoty lokálních proměnných, atributů třídy či atributů instance (viz Obrázek 5.25). Přidána je i podpora pro vytváření a spouštění JUnit testů (viz Obrázek 5.26). Otestovat lze buď konkrétní metoda či třída, případně lze spustit veškeré testy obsažené v projektu najednou.




Obrázek 5.25: BlueJ: ladění programů




Obrázek 5.26: BlueJ: výsledky JUnit testování

### 5.3.9 Další možnosti

BlueJ umožňuje i vývoj Java ME aplikací. K tomu je ovšem zapotřebí doinstalování potřebných nástrojů. Jak BlueJ rozšířit o tuto technologii a následně ji využívat podrobně popisuje tutoriál obsažený v dokumentaci pro BlueJ [58].

Obsažena je i možnost sdílení týmových projektů využitím centralizovaných systému CVS a Subversion. Nástroje pro týmovou práci nejsou ve standardním nastavení zobrazeny, to je ovšem možné změnit v nastavení prostředí.

Mezi další možnosti tohoto prostředí patří například možnost importovat

do projektu již existující třídy, generování dokumentace či vytváření spustitelných `.jar` souborů.

### 5.3.10 Možnosti rozšíření

I BlueJ je možné rozšířit o některé funkcionality. V současnosti (20. 3. 2016) je možné vybírat z 23 rozšíření. Rozšíření se stahují z webového prohlížeče ve formě `.jar` souborů, které se umístí do složky `extensions`. Pro tuto složku existují tři různé lokace, které definují možnosti používání daného rozšíření. Ve stručnosti může být umístěním složky `extensions`:

- instalační složka BlueJ — rozšíření dostupné pro všechny uživatele a všechny projekty,
- domovská složka uživatele — rozšíření dostupné pro všechny projekty konkrétního uživatele,
- složka projektu — rozšíření dostupné pro konkrétní projekt.

Přesné lokace jsou závislé na operačním systému počítače. Podrobné informace lze nalézt na webových stránkách týkajících se rozšíření prostředí BlueJ [59].

### 5.3.11 Dokumentace

Dokumentace k prostředí je dostupná pouze online, a to prostřednictvím odkazu obsaženým v prostředí. Rozdělena je do podrobných tutoriálů, které popisují základní práci s prostředím, dále testování či týmovou práci. K dispozici jsou i návody přeložené do dalších jazyků. Ty ovšem nejsou aktuální (například český návod se vztahuje k verzi 1.2.x a naposled byl aktualizován roku 2002) [58].

Jako kvalitní zdroj informací lze využít i knihu vydanou přímo autory prostředí [60].

## 5.4 Greenfoot

Vývojové prostředí Greenfoot je vyvíjeno týmem vývojářů pracujících na univerzitě Kent v Canterbury ve Velké Británii, tedy stejným týmem, který vyvíjí více známé IDE BlueJ. Z toho je patrné, že obě prostředí budou mít mnoho společného. Greenfoot se však zaměřuje především na tvorbu vizuálních 2D aplikací, jejichž vývoj je nejen snadný, ale i zábavný, díky čemuž lze lépe proniknout do objektového programování [36].

### 5.4.1 Instalace

Asi nikoho nepřekvapí, že instalační balíky jsou stejné, jako je tomu u prostředí BlueJ. Díky implementaci v Javě lze Greenfoot využívat na jakémkoliv operačním systému. Nechybí ani možnost stažení přenosné verze s vlastním JDK, kterou lze spouštět z přenosných médií. Součástí instalačních balíků jsou i ukázkové projekty, které lze pro začátek modifikovat pro ujasnění práce s prostředím.

Poslední verzí, kterou lze nainstalovat, je Greenfoot 3.0.2 (vydána 23. 12. 2015), která je v této práci rovněž popisována.

### 5.4.2 Spuštění a konfigurace

Start prostředí je o několik málo vteřin pomalejší než je tomu u IDE BlueJ. Jedno okno aplikace může obsahovat pouze jeden projekt. V projektech se rovněž „míchají“ jednotlivé typy souborů a není pro ně stanovena adresářová struktura. Adresáře jsou použity pouze pro obrázky a zvuky přidané do projektu.

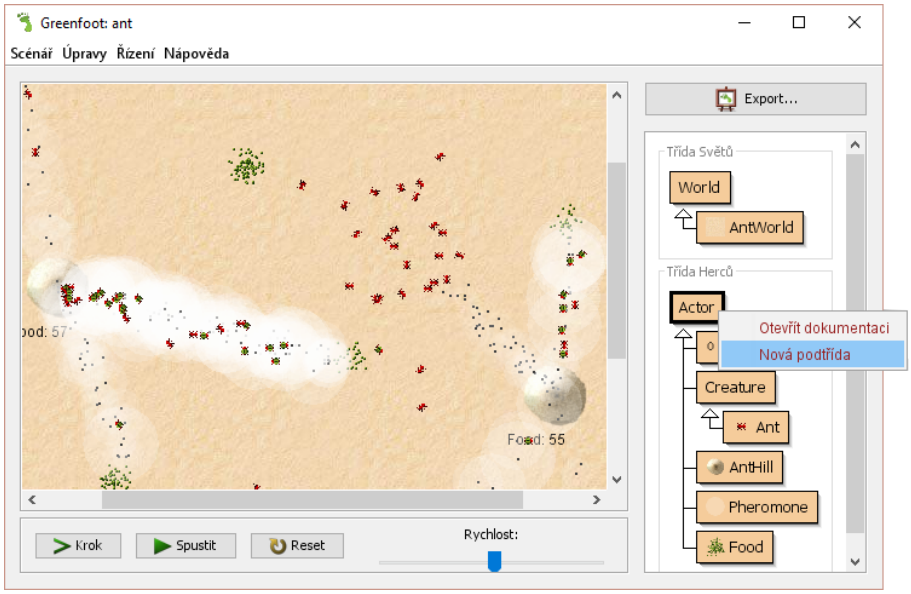
V konfiguraci prostředí toho není příliš mnoho co nastavovat kromě základních nastavení editoru či klávesových zkratek. Důležitá je ovšem možnost volby použitého jazyka v prostředí, kdy je možné vybírat z 18 jazyků včetně češtiny.

### 5.4.3 Popis prostředí

Greenfoot se zaměřuje především na vizuální a interaktivní práci s objekty. Proto je prostředí založené na Greenfoot API, tedy vlastních třídách, které nejsou ve standardních knihovnách Javy obsaženy. Jedná se o několik tříd, které umožňují snadno a rychle vytvářet jednoduché hry či simulace i úplným začátečníkům s objektovým programováním. Práci s prostředím lze rozdělit do několika sekcí, které jsou popsány níže.

### 5.4.4 Diagram tříd

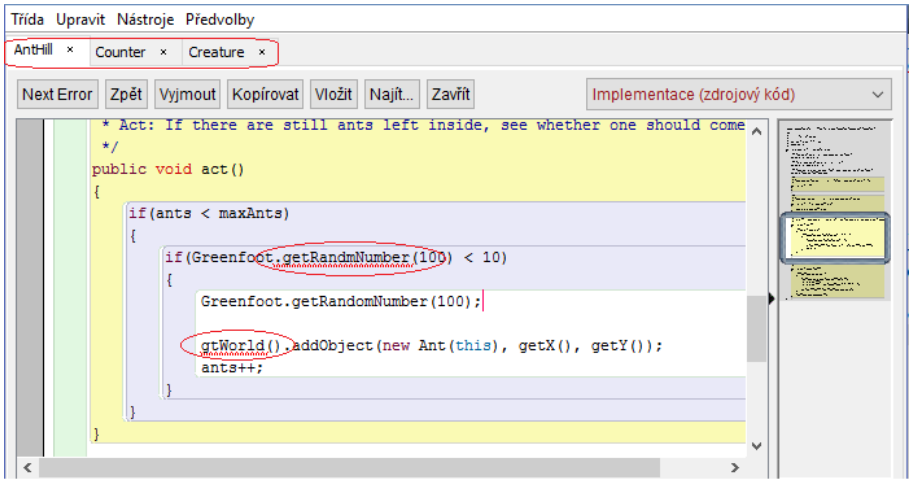
Práce s prostředím se odvíjí od dvou hlavních tříd. První z nich je třída `World` (Svět), která reprezentuje grafické plátno. Druhá třída, třída `Actor` (Aktér), reprezentuje objekty objevující se ve světě, tedy objekty přidávané na plátno. Ty mají typicky své vlastnosti a chování. Obě tyto třídy jsou součástí diagramu tříd v prostředí a dvojklikem lze zobrazit jejich dokumentaci. Jak ukazuje Obrázek 5.27, třídy vytvářené programátorem jsou potomci právě těchto dvou tříd. Možné je i importovat do projektu několik již vytvořených tříd, které jsou součástí instalačního balíku.




Obrázek 5.27: Greenfoot: hlavní okno aplikace

### 5.4.5 Implementace kódu

Při vytváření aplikací nelze generovat žádný kód, vše je tedy nutné implementovat přímo v editoru. Ten se na první pohled nijak neliší od editoru v prostředí BlueJ. Ve třech stěžejních věcech se ovšem odlišuje. První z nich je možnost otevřít více tříd v jednom editoru. Díky tomu odpadá nepříjemnost v podobě několika otevřených oken aplikace a přepínání mezi nimi. Druhým rozdílem je kompilace kódu, jelikož je kód kompilován automaticky již během jeho editace. S tím souvisí i poslední rozdíl a tím je zobrazování nalezených chyb. Greenfoot dokáže najít všechny chyby v kódu a ne tedy jen první jako IDE BlueJ. Tyto rozdíly je možné spatřit na Obrázku 5.28.




Obrázek 5.28: Greenfoot: editor

### 5.4.6 Práce s objekty

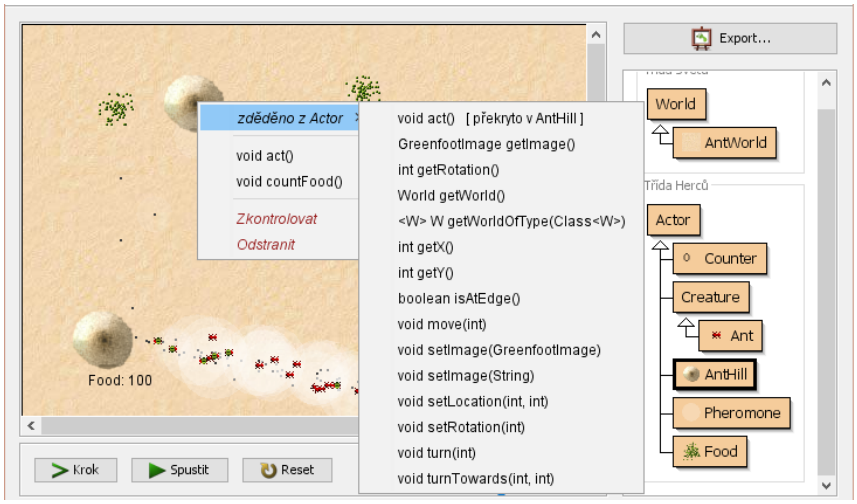
Při vytváření nových tříd je možné k nim rovnou přiřadit obrázek. Možné je vybírat z celé řady obrázků knihovny Greenfoot nebo importovat vlastní. Pro třídy světů se implementují především konstruktory, které definují velikost světa a rozvržení objektů ve světě. Mohou pochopitelně obsahovat i metody instance, ovšem řadu metod (například pro přidání/odebrání objektu ze světa) již dědí od nadřazené třídy `World`.

Pro objekty, nebo-li aktéry ve světě je nejpodstatnější metoda `act()`, která definuje chování objektu ve světě. Tato metoda je zděděna z třídy `Actor`, ale

sama o sobě nic nedělá. Až její překrytí v třídě potomka a implementování jejího těla umožňuje vykazovat objektu určité chování jako například pohyb.

Základem spuštění některé simulace je vytvoření instance konkrétního světa. Poté je možné svět ovládat tlačítky, které lze vidět na Obrázku 5.27. Tlačítko *Krok* vyvolá spuštění metody `Act()` u každého objektu obsaženém ve světě. Tlačítko *Spustit* spustí simulaci světa a to tím, že opakovaně volá metody `Act()` jednotlivých aktérů, přičemž lze i nastavovat rychlost simulace, tedy prodlevu mezi opakovaným voláním této metody. Během běhu simulace je možné ji stejným tlačítkem i pozastavit. Tlačítko *Reset* slouží pro obnovení světa do základního stavu.

Pokud je simulace pozastavena, je možné s objekty jednoduše manipulovat myší. To platí i pro přidávání dalších objektů do světa, kdy stačí kliknout na danou třídu, podržet klávesu *Shift* a jednoduše přidávat další aktéry do světa. Rovněž je možné interaktivně volat metody instance jednotlivých objektů, či prohlížet jejich atributy, což ukazuje Obrázek 5.29.




Obrázek 5.29: Greenfoot: interaktivní práce s objekty

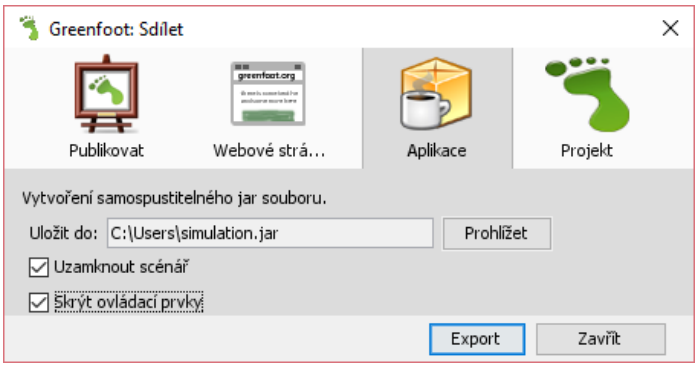
### 5.4.7 Ladění

V prostředí nechybí debugger pro ladění programů. Greenfoot využívá stejný debugger jako je obsažený v IDE BlueJ. Ten je možné vidět na Obrázku 5.25.

### 5.4.8 Další možnosti

Jelikož je možné přidávat k simulacím zvukové efekty, je zde obsažena i možnost zaznamenat zvukový výstup simulace. Celý záznam, případně pouze jeho vybranou část, lze poté uložit do souboru `.wav`.

Vytvořenou aplikaci lze exportovat do spustitelného `.jar` souboru, přičemž lze nastavit, zda bude možné simulaci ovládat tlačítky jako je tomu v prostředí či nikoli (viz Obrázek 5.30). Přidána je i možnost vytvoření webové stránky s appletem či publikování simulace v galerii scénářů organizace Greenfoot.




Obrázek 5.30: Greenfoot: export aplikace

V prostředí je možné vytvářet i jiné třídy, než pouze potomky tříd `World` a `Actor`. Lze tedy implementovat programy využívající terminál, jednoduché GUI formuláře apod. U některých aplikací však docházelo po jejich ukončení k restartování hlavního okna prostředí, což je dosti nepříjemné. To značí, že i když Greenfoot umožňuje vytvářet i jiné aplikace, primárně je určen právě pro 2D hry a simulace.

### 5.4.9 Dokumentace

Dokumentace k prostředí je dostupná pouze online, a to prostřednictvím odkazu obsaženým v aplikaci. Zde je možné najít několik tutoriálů či videí, které podrobně popisují práci s prostředím. Na domovských stránkách Greenfoot lze rovněž najít diskuzní fórum pro registrované uživatele či galerii uživatelských scénářů (projektů), které lze stáhnout a využít jako inspiraci pro vlastní aplikace. Přímo v prostředí je obsažena pouze dokumentace tříd

Greenfoot API [61].

K dispozici je i oficiální kniha, vydaná přímo autorem prostředí, kterou mohou využít učitelé i studenti [62].

# 6 Porovnání prostředí

Výše popsané aplikace se zaměřují na dvě různé oblasti — vývoj a výuku. Proto i kritéria, podle kterých se budou aplikace hodnotit, musí být odlišné vzhledem k oblasti jejich zaměření. Pro porovnávaná prostředí budou nejprve vymezena kritéria hodnocení, jež budou následně slovně okomentována. Na závěr bude přidána tabulka s bodovým ohodnocením zvolených kritérií (0-5 bodů), ze které budou patrné přednosti a nedostatky jednotlivých aplikací.

Velkou roli při výběru správného IDE hraje dojem samotného uživatele, proto je k profesionálním prostředím připojeno i dotazníkové šetření (viz příloha A). Odpovědi respondentů budou shrnuty v sekci 6.2.14. Pro výuková prostředí dotazník připojen není, vzhledem k obtížnosti samostatného seznámení začátečníků s možnostmi daných aplikací.

## 6.1 Výuková prostředí

Hodnotit výuková prostředí je mnohem jednodušší než hodnotit ty profesionální. Vzhledem k tomu, že obě popsané aplikace přistupují k výuce jiným způsobem lze snadno zhodnotit, který přístup je lepší. Pro výuková prostředí jsem zvolil následující kritéria hodnocení:

- jednoduchost prostředí,
- ukázkové projekty,
- diagram tříd,
- editor,
- vizualizace a práce s objekty,
- ladění a testování,
- dostupné materiály,
- další možnosti.

### 6.1.1 Jednoduchost prostředí

Jelikož jsou obě prostředí založena na stejném konceptu, tak jejich rozhraní jsou téměř stejná. Obě aplikace působí velmi přívětivě a nenabízí žádné složité funkcionality, které by začátečník nevyužil. Pro seznámení s prostředími postačí velmi krátký čas a je tedy možné prakticky ihned začít vytvářet první projekty. Obě aplikace rovněž nabízí lokalizaci několika světových jazyků včetně češtiny.

### 6.1.2 Ukázkové projekty

Instalační balíky obou aplikací obsahují i několik ukázkových projektů. Greenfoot má však navrch v tom, že na svých stránkách udržuje galerii uživatelských projektů, které je možné stáhnout a využít jako inspiraci pro vlastní aplikace.

### 6.1.3 Diagram tříd

Diagram v IDE Greenfoot zobrazuje pouze hierarchii tříd, jež jsou potomci tříd `World` a `Actor`. Pokud projekt obsahuje i jiné třídy, nejsou u nich zobrazeny vazby na jejich okolí. BlueJ naopak obsahuje velmi podrobný diagram tříd, který zobrazuje veškeré vazby v daném projektu.

### 6.1.4 Editor

Editory obou aplikací na první pohled vypadají shodně, ovšem velmi se liší. Greenfoot na rozdíl od svého protějšku umožňuje editovat více tříd v jednom okně editoru. Navíc je přidána funkce automatického překladu kódu, která odhalí všechny stávající chyby. BlueJ v tomto směru velmi pokulhává, jelikož je nutné kód překládat ručně a označí se vždy jen první nalezená chyba.

### 6.1.5 Vizualizace a práce s objekty

Výhodou IDE Greenfoot je možnost přidávat k objektům jednoduše obrázky a simulovat jejich činnost, což činí práci zábavnou. Objekty je navíc možné do simulace přidávat i způsobem *Drag and drop*. U objektů v simulaci je možné prohlížet jejich atributy či volat všechny jejich metody. Celý dojem ovšem kazí skutečnost, že u objektů mimo simulaci nelze volat jejich metody instance a je tedy nutné použít metodu `main()` či najít jiný způsob. BlueJ umožňuje snadno pracovat s objekty přes jejich grafickou reprezentaci, tzn. prohlížet jejich atributy a volat jejich metody. Pracovat s nimi lze i využitím příkazové řádky obsažené v prostředí.

### 6.1.6 Ladění a testování

Ladění aplikací umožňují obě prostředí využitím stejného debuggeru. BlueJ však oproti svému protějšku obsahuje i podporu JUnit testování, což je nezbytná činnost budoucích programátorů.

### 6.1.7 Dostupné materiály

K oběma prostředím je k dispozici oficiální kniha a řada manuálů. Organizace Greenfoot navíc spravuje vlastní kanál *Youtube*, který obsahuje téměř sedm desítek videí.

### 6.1.8 Další možnosti

Greenfoot je zaměřen především na tvorbu simulací, proto další možnosti neobsahuje. BlueJ je oproti tomu mnohem více univerzálnější. Lze jej doplnit o řadu rozšíření jako je například jednoduchý GUI builder. Možné je doplnit i podporu pro tvorbu Java ME aplikací. Navíc je již v prostředí obsažena podpora pro týmovou práci na projektech využitím systémů CVS a Subversion. Z toho plyne, že i když je BlueJ výukové prostředí, lze v něm teoreticky vytvořit jakoukoliv aplikaci.

### 6.1.9 Bodové hodnocení

Protože se aplikace porovnávají mezi sebou, obdrží vždy jedna z nich v dané kategorii maximální počet bodů. Druhá potom obdrží odpovídající podíl bodů vzhledem k tomu, jak moc za svým protějškem v dané kategorii zaostává.

Tabulka 6.1: Porovnání výukových prostředí

| **Kritérium** | **BlueJ** | **Greenfoot** |
|---|---|---|
| Jednoduchost prostředí | 5 | 5 |
| Ukázkové projekty | 3 | 5 |
| Diagram tříd | 5 | 2 |
| Editor | 2 | 5 |
| Vizualizace a práce s objekty | 5 | 4 |
| Ladění a testování | 5 | 3 |
| Dostupné materiály | 3 | 5 |
| Další možnosti | 5 | 0 |
| **Průměr** | **4,13** | **3,63** |

### 6.1.10 Shrnutí

Jak ukazuje Tabulka 6.1, lépe z provedeného porovnání dopadl BlueJ. To přikládám především jeho univerzálnosti. Aplikaci Greenfoot lze doporučit jako nástroj, jež zaujme mladé studenty a umožní jim učit se OOP zábavnou formou. Využít jej lze na základních školách či v prvním ročníku středních škol. Práce s IDE BlueJ sice není tak jednoduchá, ovšem nutí studenty od samého začátku využívat standardní třídy rozhraní API, což je přinutí mnohem více nad svým kódem přemýšlet. Navíc díky možnostem testování a rozšíření o další funkcionality BlueJ poskytuje více možností pro práci s prostředím, díky čemuž lze později snáze přejít na profesionální IDE. Využití lze doporučit na středních školách či případně v prvním ročníku vysokých škol.

## 6.2 Profesionální prostředí

Hodnotit profesionální vývojová prostředí není úplně jednoduché, jelikož jejich funkcionalitu lze neustále rozšiřovat dalšími pluginy. Proto bude porovnáno, co obě prostředí nabízí ve své „základní verzi“, tedy bez jakéhokoliv rozšíření. Pro vývojová prostředí jsem zvolil následující kritéria hodnocení:

- podpora Java technologií,
- reakce prostředí,
- možnosti přizpůsobení,
- generování kódu,
- kontextová nabídka,
- refaktoring,
- testování,
- ladění,
- verzování,
- nápověda v prostředí,
- dostupné materiály,
- možnosti rozšíření.

### 6.2.1 Podpora Java technologií

U obou prostředí lze vybírat z několika instalačních balíků. Java vývojáře budou zajímat především balíky Java SE a Java EE, které nabízejí obě prostředí. Netbeans, jako oficiální prostředí Oraclu, nabízí navíc balík *All*, který obsahuje podporu i pro technologie Java FX a Java ME.

### 6.2.2 Reakce prostředí

Start prostředí je závislý na počtu načítaných modulů, proto jej nelze brát jako směrodatný faktor. Důležitější je doba reakce prostředí na různé činnosti a v tomto ohledu působí Eclipse mnohem svižněji.

### 6.2.3 Možnosti přizpůsobení

V případě přizpůsobení prostředí zachází Eclipse do větších detailů než jeho protějšek. Navíc umožňuje přepínat mezi různými perspektivami, které zobrazují různé rozložení pohledů vzhledem k prováděné činnosti.

### 6.2.4 Generování kódu

Možnosti generování kódu se zdají na první pohled stejné. Výhoda Netbeans se však s krývá ve správci šablon, které lze využít při vytváření nových tříd. V prostředí je již definována celé řada šablon pro různé třídy, které je navíc možné editovat či vytvořit přímo vlastní. Další výhodou je integrovaný GUI builder pro interaktivní vytváření GUI aplikací.

### 6.2.5 Kontextová nápověda

Kontextové nápovědy obou prostředí se zobrazí prakticky okamžitě. Eclipse umožňuje nastavit si zpoždění zobrazení nápovědy, ovšem nepředpokládám, že by to někdo využil. Co je užitečné, je možnost omezovat návrhy nápovědy dle určitých kategorií, což Netbeans nenabízí. I díky tomu působí kontextová nápověda Eclipsu mnohem přehledněji.

### 6.2.6 Refaktoring

Možnosti refaktoringu jsou u obou prostředí prakticky stejné. Eclipse si navíc udržuje historii refaktoringu jednotlivých projektů ve workspace.

### 6.2.7 Ladění

Obě prostředí umožňují standardní práci s debuggerem jako je krokování programu, sledování proměnných či přidávání podmínek. Netbeans navíc obsahuje i možnost pořizovat při ladění snímky GUI aplikací a zobrazovat informace o jednotlivých komponentách na snímku.

### 6.2.8 Testování

Podpora jednotkového testování je standardem většiny profesionálních prostředí. Využívají se k tomu frameworky JUnit a TestNG, Eclipse však obsahuje pouze první zmiňovaný. Dále se k testování využívají nástroje pro výkonové testy či pro statickou analýzu kódu, jež jsou oba integrované v prostředí Netbeans. Eclipse umožňuje provádět pouze velmi omezenou statickou analýzu kódu.

### 6.2.9 Verzování

Kromě lokální historie souborů, kterou si udržují obě prostředí, jsou standardním vybavením profesionálních IDE i nástroje pro práci s verzovacími systémy. Součástí Eclipsu je pouze Git. Netbeans obsahuje i podporu pro verzovací systémy Mercurial a Subversion.

### 6.2.10 Nápověda v prostředí

Eclipse disponuje velmi podrobnou a přehlednou nápovědou integrovanou v prostředí. V porovnání s ní působí nápověda v prostředí Netbeans velmi stroze. Netbeans totiž udržuje podrobné tutoriály, na které nápověda odkazuje pouze online, což je velká nevýhoda při nedostupnosti internetového připojení.

### 6.2.11 Zdroje informací

Veškeré informace lze nalézt na webových stránkách samotných organizací, přičemž poněkud lepší orientace v datech je na stránkách Netbeans. K oběma prostředím se váže i řada knižních publikací.

### 6.2.12 Možnosti rozšíření

Pro obě prostředí je k dispozici obrovské množství pluginů, o které jdou rozšířit. Eclipse však v tomto směru Netbeans výrazně převyšuje. V porovnání je to zhruba 1600:1000 pluginům. Jediná výhoda Netbeans je pouze možnost omezovat se při hledání rozšíření na konkrétní verzi IDE, což Eclipse neumožňuje.

### 6.2.13 Bodové hodnocení

Protože se aplikace porovnávají mezi sebou, obdrží vždy jedna z nich v dané kategorii maximální počet bodů. Druhá potom obdrží odpovídající podíl bodů vzhledem k tomu, jak moc za svým protějškem v dané kategorii zaostává. Jde tedy o stejný způsob hodnocení jako u výukových prostředí.

Tabulka 6.2: Porovnání vývojových prostředí

| **Kritérium** | **Eclipse** | **Netbeans** |
|---|---|---|
| Podpora Java technologií | 3 | 5 |
| Reakce prostředí | 5 | 3 |
| Možnosti přizpůsobení | 5 | 3 |
| Generování kódu | 2 | 5 |
| Kontextová nápověda | 5 | 3 |
| Refaktoring | 5 | 4 |
| Ladění | 3 | 5 |
| Testování | 2 | 5 |
| Verzování | 2 | 5 |
| Nápověda v prostředí | 5 | 2 |
| Zdroje informací | 4 | 5 |
| Možnosti rozšíření | 5 | 3 |
| **Celkem** | **3,83** | **4,0** |

### 6.2.14 Dotazníkové šetření

Pro sběr dat byla oslovena skupina 10 respondentů, kteří mají zkušenosti s používáním profesionálních vývojových prostředí. Vzhledem k anonymitě jsou jednotliví respondenti označeni v tabulkách pouze písmenem. Tabulky obsahují pouze odpovědi respondentů (1 — nejhorší, 5 — nejlepší) na daná tvrzení. Dotazník byl inspirován semestrální prací na téma kvantitativní testování dvou vývojových prostředí [63]. Kompletní znění dotazníku je možné najít v příloze A.

Tabulka 6.3: Eclipse: výsledek dotazníku

| **Tvrzení/Respondent** | A | B | C | D | E | F | G | H | I | J | **Průměr** |
|---|---|---|---|---|---|---|---|---|---|---|---|
| T1 | 4 | 3 | 4 | 4 | 4 | 4 | 4 | 4 | 2 | 4 | 3,7 |
| T2 | 4 | 2 | 4 | 5 | 4 | 4 | 5 | 4 | 2 | 4 | 3,8 |
| T3 | 3 | 4 | 3 | 4 | 4 | 2 | 4 | 4 | 3 | 5 | 3,6 |
| T4 | 2 | 2 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 5 | 3,7 |
| T5 | 5 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4,1 |
| T6 | 5 | 2 | 5 | 4 | 2 | 4 | 3 | 5 | 1 | 5 | 3,6 |
| T7 | 3 | 2 | 3 | 4 | 3 | 4 | 4 | 4 | 2 | 5 | 3,4 |
| T8 | 2 | 4 | 4 | 5 | 4 | 2 | 4 | 4 | 4 | 5 | 3,8 |
| T9 | 5 | 3 | 5 | 5 | 4 | 4 | 4 | 5 | 1 | 5 | 4,1 |
| **Celkový průměr** | | | | | | | | | | | 3,76 |

Tabulka 6.4: Netbeans: výsledek dotazníku

| **Tvrzení/Respondent** | A | B | C | D | E | F | G | H | I | J | **Průměr** |
|---|---|---|---|---|---|---|---|---|---|---|---|
| T1 | 3 | 4 | 2 | 3 | 4 | 5 | 3 | 2 | 4 | 3 | 3,3 |
| T2 | 3 | 4 | 3 | 4 | 4 | 5 | 3 | 3 | 4 | 2 | 3,5 |
| T3 | 2 | 2 | 2 | 4 | 4 | 5 | 5 | 2 | 2 | 5 | 3,3 |
| T4 | 2 | 2 | 4 | 3 | 2 | 4 | 4 | 3 | 4 | 2 | 3,0 |
| T5 | 5 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 2 | 3,9 |
| T6 | 5 | 2 | 4 | 4 | 4 | 4 | 4 | 4 | 1 | 5 | 3,7 |
| T7 | 3 | 2 | 2 | 3 | 3 | 3 | 4 | 2 | 2 | 5 | 2,9 |
| T8 | 2 | 4 | 3 | 4 | 3 | 4 | 4 | 3 | 4 | 4 | 3,5 |
| T9 | 4 | 4 | 3 | 4 | 4 | 5 | 4 | 2 | 1 | 1 | 3,2 |
| **Celkový průměr** | | | | | | | | | | | 3,37 |

### 6.2.15 Shrnutí

Jak ukazuje Tabulka 6.2, o něco lépe z provedeného porovnání dopadl Netbeans, a to především díky množství funkcionalit, jež integruje. Eclipse měl navrch v oblasti intuitivnosti, jelikož působil propracovanějším dojmem (vzhled, možnosti přizpůsobení, reakce prostředí). To se ostatně projevilo i v dotazníkovém šetření, kde dopadl lépe právě Eclipse.

Označit jedno prostředí jako lepší prakticky není možné. Závisí především na dojmu samotného uživatele. Navíc díky množství pluginů, které pro obě prostředí existují, mohou disponovat prakticky totožnými možnostmi. Obě prostředí jsou zdarma, proto doporučuji si obě stáhnout, vyzkoušet na menších projektech a rozhodnout se dle vlastního názoru. Obdobné srovnání, jako bylo provedeno v této práci, lze pochopitelně provést i u dalších prostředí dostupných na trhu.

# 7 Závěr

Cílem této práce bylo objevit seznam existujících vývojových prostředí pro jazyk Java, ať již zamýšlených pro vývoj či výuku, a následné porovnání vhodných dvojic z oblasti výuky a vývoje.

Z oblasti výuky byla zvolena prostředí BlueJ a Greenfoot, z oblasti vývoje potom prostředí Eclipse a Netbeans. Všechna zvolená prostředí byla nejprve podrobně popsána a následně porovnána se svým protějškem dle definovaných kritérií. U vývojových (profesionálních) prostředí je kromě vlastního porovnání připojeno i dotazníkové šetření, které mapuje názory uživatelů na tyto aplikace.

Přínosem práce je především popis možností jednotlivých aplikací a definování bodů, ve kterých jsou aplikace porovnány se svým konkurentem. Díky tomu si může čtenář ujasnit, v čem jsou silné a slabé stránky konkrétních aplikací. Dalším přínosem práce je uvedený seznam různých prostředí pro jazyk Java, jelikož jsem žádný kompletní seznam nenalezl. Seznam v této práci čítá téměř dvě desítky aplikací, není ovšem vyloučené, že existují další.

Pro obdobné práce bych doporučil omezit se pouze na jednu oblast zaměření aplikací, tedy pouze na vývoj či výuku. To by umožnilo hlouběji prozkoumat jednotlivé aplikace či je porovnat s více konkurenty z dané oblasti. Rovněž by bylo možné vytvořit v dotaznících konkrétní scénář na konkrétním projektu, který by bylo nutné splnit. Díky tomu by bylo možné měřit i dobu vykonávání úkolů v jednotlivých aplikacích.

# Přehled zkratek

**API** (Application Programming Interface) — rozhraní pro programování aplikací

**IDE** (Integrated Development Enviroment) — pojem označující vývojové prostředí

**FAQ** (Frequently Asked Questions) — často kladené otázky

**GUI** (Graphical User Interface) — grafické uživatelské rozhraní aplikací

**Java EE** (Enterprise Edition) — umožňuje vývoj podnikových aplikací

**Java FX** (Effects) — umožňuje vývoj bohatých grafických aplikací

**Java ME** (Micro Editon) — umožňuje vývoj mobilních aplikací

**Java SE** (Standard Edition) — standardní edice jazyka Java

**JDK** (Java Development Kit) — nástroje umožňující vývoj aplikací

**JRE** (Java Runtime Enviroment) — prostředí Javy umožňující běh aplikací

**JVM** (Java Virtual Machine) — virtuální stroj vykonávající bajtkód

**OOP** (Object-oriented Programming) — objektově orientované programování

**RCP** (Rich Client Platform) — sada modulů, které lze využít jako základ klientských aplikací

# Seznam obrázků

# Seznam tabulek

# Literatura


[1] Herout, Pavel. *Učebnice jazyka Java*. 3., rozš. vyd. [i.e. 4. vyd.]. České Budějovice: Kopp, 2008, 381 s. ISBN 978-80-7232-355-5.

[2] Novotný, Luděk. *Historie a vývoj Jazyka Java* [online]. Praha: 2003 [cit. 2015-11-2]. Dostupné z: http://www.fi.muni.cz/usr/jkucera/pv109/2003p/xnovotn8.htm.

[3] *Difference between JDK, JRE and JVM*. JavaTpoint [online]. [cit. 2015-11-5]. Dostupné z: http://www.javatpoint.com/difference-between-jdk-jre-and-jvm.

[4] *Use Notepad and Command Prompt For Java programming*. Instructables [online]. [cit. 2015-11-5]. Dostupné z: http://www.instructables.com/id/Use-Notepad-and-Command-Prompt-For-Java-programmin/?ALLSTEPS.

[5] *Best Free Programming Editor* 2015-03-05 [online]. [cit. 2015-11-5]. Dostupné z: http://www.techsupportalert.com/best-free-programming-editor.htm.

[6] *Integrated development environment (IDE) definition* [online]. [cit. 2015-11-5]. Dostupné z: http://searchsoftwarequality.techtarget.com/definition/integrated-development-environment.

[7] *What is RCP and why should I care?* [online]. Modular Mind [cit. 2016-03-31]. Dostupné z: http://www.modumind.com/what-is-rcp/.

[8] *Rich Client Platform*. Eclipse [online]. 2015-10-14 [cit. 2016-03-31]. Dostupné z: https://wiki.eclipse.org/Rich_Client_Platform.

[9] *Adresářová struktura projektů v Javě* [online]. 2007-04-02 [cit. 2015-11-5]. Dostupné z: http://java.vse.cz/SVN/JavaAdresare.

[10] *Java-Packages*. Tutorialspoint [online]. [cit. 2015-11-17]. Dostupné z: http://www.tutorialspoint.com/java/java_packages.htm.

[11] *Přehled fontů s pevnou šířkou*. AbcLinuxu [online]. 2010-8-31 [cit. 2015-11-17]. Dostupné z: http://www.abclinuxu.cz/clanky/prehled-fontu-s-pevnou-sirkou.

[12] *Eclipse efektivně – Content Assist* [online]. 2009-09-09 [cit. 2015-11-17]. Dostupné z: http://www.aspectworks.com/2009/09/eclipse-efektivne-1-content-assist.

[13] FORD, Jerry Lee. *Microsoft Visual Basic 2008 express programming for the absolute beginner*. Australia: Course Technology, c2009. ISBN 978-1-59863-900-1.

[14] *Common syntax errors in Java* [online]. [cit. 2015-11-17]. Dostupné z: http://northern.lkdsb.net/Kedwell/ICS4U/Java/intro/Syntax

[15] *Javadoc tutorial* [online]. [cit. 2015-11-23]. Dostupné z: https:// students.cs.byu.edu/ cs240ta/fall2012/tutorials/javadoctutorial.html.

[16] *Debugging Your First Java Application*. JetBrains [online]. [cit. 2015-11-23]. Dostupné z: https://www.jetbrains.com/idea/help/debugging-your-first-java-application.html.

[17] KING, Tim. *Twelve Benefits of Writing Unit Tests First* [online]. [cit. 2015-11-23]. Dostupné z: http://sd.jtimothyking.com/2006/07/11/twelve-benefits-of-writing-unit-tests-first/.

[18] *Nástroje pro podporu vývoje na platformě Java* [online]. [cit. 2015-11-23]. Dostupné z: http://www.skoumal.net/cs/nastroje-pro-podporu-vyvoje-na-platforme-java/.

[19] *Distribuované verzovací systémy – úvod (1)*. AbcLinuxu [online]. 2011-1-25 [cit. 2016-03-31]. Dostupné z: http://www.abclinuxu.cz/clanky/distribuovane-verzovaci-systemy-uvod-1.

[20] *11 BEST JAVA IDES FOR JAVA PROGRAMMERS*. Codecall [online]. 2014-11-12 [cit. 2015-12-1]. Dostupné z: http://codecall.net/2014/11/12/11-best-desktop-web-ides-java-programmers/.

[21] *18 Best Free Java IDE Software*. List of Freeware [online]. [cit. 2015-12-1]. Dostupné z: http://listoffreeware.com/list-of-best-free-java-ide-software/.

[22] *Nejlepší nástroje na programování v Javě*. ComputerWorld [online]. 2011-03-14 [cit. 2015-12-1]. Dostupné z: http://computerworld.cz/vyvoj/nejlepsi-nastroje-na-programovani-v-jave-42939.

[23] *NetBeans IDE - The Smarter and Faster Way to Code*. Netbeans [online]. [cit. 2015-12-1]. Dostupné z: https://netbeans.org/features/index.html.

[24] *IntelliJ IDEA: The Most Intelligent Java IDE*. JetBrains [online]. [cit. 2016-02-23]. Dostupné z: https://www.jetbrains.com/idea/#chooseYourEdition.

[25] *Available Subscription Options*. JetBrains Store [online]. [cit. 2016-02-23]. Dostupné z: https://www.jetbrains.com/store-/terms/comparison.html#LicenseComparison.

[26] *What is the IntelliJ Platform?*. JetBrains [online]. [cit. 2016-03-02]. Dostupné z: http://www.jetbrains.org/pages/viewpage.action?pageId=983889.

[27] *About BlueJ*. BlueJ [online]. [cit. 2015-12-5]. Dostupné z: http://www.bluej.org/about.html.

[28] *Android Studio - nové vývojové prostředí* [online]. 2013-11-04 [cit. 2015-12-5]. Dostupné z: https://www.zdrojak.cz/clanky/android-studio-nove-vyvojove-prostredi/.

[29] JEMEROV, Dmitry. *IntelliJ IDEA and Android Studio FAQ*. In: IntelliJ Idea Blog [online]. 2013 [cit. 2016-02-23]. Dostupné z: http://blog.jetbrains.com/idea/2013/05/intellij-idea-and-android-studio-faq/.

[30] *JCreator* [online]. [cit. 2015-12-10]. Dostupné z: http://www.jcreator.com/.

[31] *JBuilder® 2008 R2 Product Editions* [online]. [cit. 2016-02-26]. Dostupné z: http://www.embarcadero.com/products/jbuilder/product-editions.

[32] *SnapCode - Building apps is a snap* [online]. [cit. 2015-12-10]. Dostupné z: http://www.reportmill.com/snap/.

[33] *NaviCoder IDE for Java Lite*. Softpedia [online]. [cit. 2016-02-11]. Dostupné z: http://www.softpedia.com/get/Programming/Other-Programming-Files/NaviCoder-IDE-for-Java-Lite.shtml.

[34] *About jGRASP* [online]. [cit. 2016-02-11]. Dostupné z: http://www.jgrasp.org/.

[35] *About DrJava* [online]. [cit. 2016-02-11]. Dostupné z: http://drjava.org/.

[36] *About Greenfoot* Greenfoot [online]. [cit. 2016-02-11]. Dostupné z: http://www.greenfoot.org/overview.

[37] *SkyIDE*. Softpedia [online]. [cit. 2016-02-11]. Dostupné z: http://www.softpedia.com/get/Programming/Coding-languages-Compilers/SkyIDE.shtml.

[38] *Asterix IDE* . Sourceforge [online]. [cit. 2016-02-11]. Dostupné z: https://sourceforge.net/projects/asterixide/.

[39] *tIDE: have tIDE on your sIDE !*. tIDE [online]. [cit. 2016-02-11]. Dostupné z: http://tide.olympe.in/.

[40] *Zeus IDE: A powerful, language neutral programmer's development environment* [online]. [cit. 2016-02-11]. Dostupné z: http://www.zeusedit.com/index.html.

[41] *Jenuity 2.4*. Software informer [online]. [cit. 2016-02-11]. Dostupné z: http://jenuity.software.informer.com/2.4/.

[42] *JotAzul: Object Oriented Java IDE* [online]. [cit. 2016-02-11]. Dostupné z: http://jotazul.sourceforge.net/.

[43] *About the Eclipse Foundation*. Eclipse [online]. [cit. 2016-03-19]. Dostupné z: https://eclipse.org/org/.

[44] *Commercial Rich client platform (RCP) applications*. Eclipse [online]. [cit. 2016-03-19]. Dostupné z: https://eclipse.org/community/rcpcp.php.

[45] *Open Source Rich client platform (RCP) applications*. Eclipse [online]. [cit. 2016-03-19]. Dostupné z: https://eclipse.org/community/rcpos.php.

[46] BURNETTE, Ed. *Eclipse IDE: pocket guide*. 1st ed. Sebastopol, CA: O'Reilly, c2005. ISBN 05-961-0065-5.

[47] VOGEL, Lars. *Eclipse IDE: Java programming, debugging, unit testing, task management and Git version control with Eclipse*. 3rd ed. Leipzig: Vogella, 2013. ISBN 39-437-4704-2.

[48] *A Brief History of NetBeans*. Netbeans [online]. [cit. 2016-03-19]. Dostupné z: https://netbeans.org/about/history.html.

[49] *The NetBeans Platform*. Netbeans [online]. [cit. 2016-03-19]. Dostupné z: https://netbeans.org/features/platform/index.html.

[50] *NetBeans Platform Showcase*. Netbeans [online]. 2016-03 [cit. 2016-03-19]. Dostupné z: https://platform.netbeans.org/screenshots.html.

[51] *NetBeans IDE Dual License Header and License Notice*. Netbeans [online]. [cit. 2016-03-19]. Dostupné z: https://netbeans.org/cddl-gplv2.html.

[52] *NetBeans IDE 8.1 Download*. Netbeans [online]. [cit. 2016-05-03]. Dostupné z: https://netbeans.org/downloads/.

[53] *NetBeans IDE Features: Versioning*. Netbeans [online]. [cit. 2016-03-26]. Dostupné z: https://netbeans.org/features/ide/versioning.html.

[54] *Welcome to the NetBeans Plugin Portal: Plugins catalogue*. Netbeans [online]. [cit. 2016-03-27]. Dostupné z: http://plugins.netbeans.org/PluginPortal/.

[55] WIELENGA, Geertjan. *Beginning Netbeans IDE: For Java Developers*. Apress, 2015. ISBN 978-1-4842-1258-5.

[56] BÖCK, Heiko. *Platforma NetBeans: podrobný průvodce programátora*. Vyd. 1. Brno: Computer Press, 2010. ISBN 978-80-251-3116-9.

[57] *Blue: What is it?* [online]. 1999-03-19 [cit. 2016-04-01]. Dostupné z: https://www.cs.kent.ac.uk/people/staff/mik/blue/.

[58] *BlueJ - The interactive Java environment: BLUEJ DOCUMENTATION* [online]. [cit. 2016-04-04]. Dostupné z: http://www.bluej.org/doc/documentation.html.

[59] *BlueJ Extensions: Installing extensions*. BlueJ [online]. [cit. 2016-04-04]. Dostupné z: http://www.bluej.org/extensions/extensions.html.

[60] BARNES, David J a Michael KÖLLING. *Objects first with Java: a practical introduction using BlueJ*. 5th ed. Boston: Pearson, c2012. ISBN 01-324-9266-0.

[61] *Greenfoot: Documentation*. Greenfoot [online]. [cit. 2016-04-4]. Dostupné z: http://www.greenfoot.org/doc.

[62] KÖLLING, Michael. *Introduction to programming with greenfoot object-oriented programming in java with games and simulations*. Second edition. Boston: Pearson, 2016. ISBN 01-340-5429-6.

[63] *Kvantitativní testování dvou vývojových prostředí* [online]. [cit. 2016-04-20]. Dostupné z: http://hcisemestralky.felk.cvut.cz/system/mems/7629/original/B1.pdf. Semestrální práce. České vysoké učení technické v Praze.

# A Dotazník

## Porovnání vývojových prostředí Eclipse a Netbeans

Následující dotazník slouží jako podklad pro bakalářkou práci porovnávající vývojová prostředí Eclipse a Netbeans. Před vyplněním dotazníku je nutné vyzkoušet si, ať již na vlastních nebo ukázkových projektech, několik základních operací v obou porovnávaných prostředích:

- vygenerování getterů/setterů, vygenerování metody toString(),
- vyzkoušení kontextové nápovědy a debuggeru,
- prozkoumání možností refaktoringu,
- prozkoumání nápovědy v prostředí,
- doinstalování libovolného pluginu.

A dále odpovědět na několik tvrzení:
T1: Prostředí působí velmi přívětivě.
T2: S prostředím se snadno pracuje.
T3: Prostředí reaguje velmi rychle.
T4: Potřebné funkcionality jsem snadno našel.
T5: Funkcionality v prostředí jsou na vysoké úrovni.
T6: Integrované funkcionality mi plně dostačují.
T7: Nápověda v prostředí je dobře zpracována.
T8: Pro ovládání prostředí bych nepotřeboval příliš času.
T9: Dovedu si představit používat toto prostředí často.

Pro odpovědi na tato tvrzení vyplňte přiloženou tabulku a to následujícím bodováním:
5 — Souhlasím
4 — Spíše souhlasím
3 — Nevím
2 — Spíše nesouhlasím
1 — Nesouhlasím

Tabulka A.1: Tabulka pro uživatelské odpovědi

| **IDE/Tvrzení** | **T1** | **T2** | **T3** | **T4** | **T5** | **T6** | **T7** | **T8** | **T9** |
|---|---|---|---|---|---|---|---|---|---|
| Eclipse | | | | | | | | | |
| Netbeans | | | | | | | | | |