

Část E

Přílohy

Tato část se vyskytuje pouze v elektronické verzi učebnice. Podrobněji vás seznámí s postupem při instalaci prostředí *JShell* v systému *Windows* včetně představení poněkud opomíjené možnosti definice substituovaných disků. Následující přílohy jsou pak určeny pro naprosté začátečníky, kteří občas zápasí s termíny a někteří i s angličtinou.

Příloha A

Příprava spuštění JShell pod Windows

A.1 Podkapitola přílohy

Problémy s klávesnicí

Už od svého prvního uvedení ve verzi 9 mělo prostředí *JShell* problém s některými jinými rozloženými kláves než US. V českém prostředí bylo možné až do verze 12 tyto problémy řešit používáním rozložení označovaného jako *České (QWERTY)*. To používá řada programátorů, protože znaky, které se nevyskytují na klávesnici psacího stroje, má toto rozložení umístěny na stejných klávesách jako rozložení US, jenom se aktivují s jiným nastavením přepínačů.

Bohužel se pak někdo rozhodl prostředí *JShell* „vylepšit“. Od té chvíle sice funguje rozložení označované jako *České*, ale pro změnu zase přestalo chodit rozložení *České (QWERTY)*, které nám nyní neumožňuje zadat některé důležité znaky – např. složené závorky (`{}`) a „svislítko“ (`|`), a to ani tak, že je do textu vložíte ze schránky.

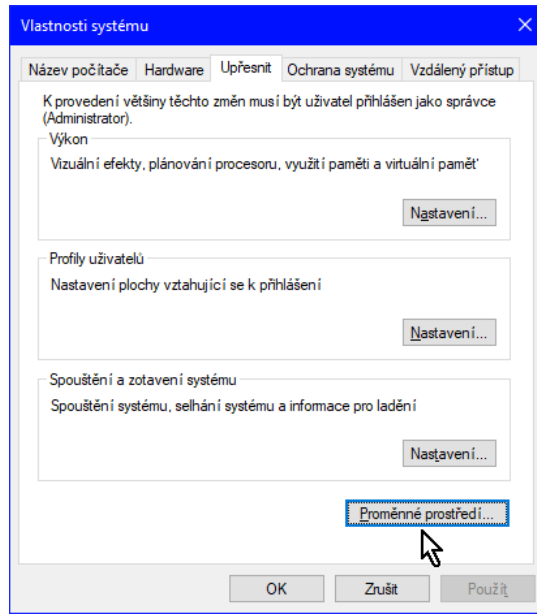
Vložit je můžete buďto přímým zadáním jejich kódu (`{=123, |=124, }=125`), anebo použitím jiného rozložení kláves.

Příprava spuštění pod Windows

Jak již bylo řečeno, jednou z velkých nevýhod *Windows* je, že mají pro různé skupinky států nastavená různá kódování. Všechny doprovodné programy k tomuto kurzu ale budou kódovány v celosvětově jednotném kódování UTF-8.

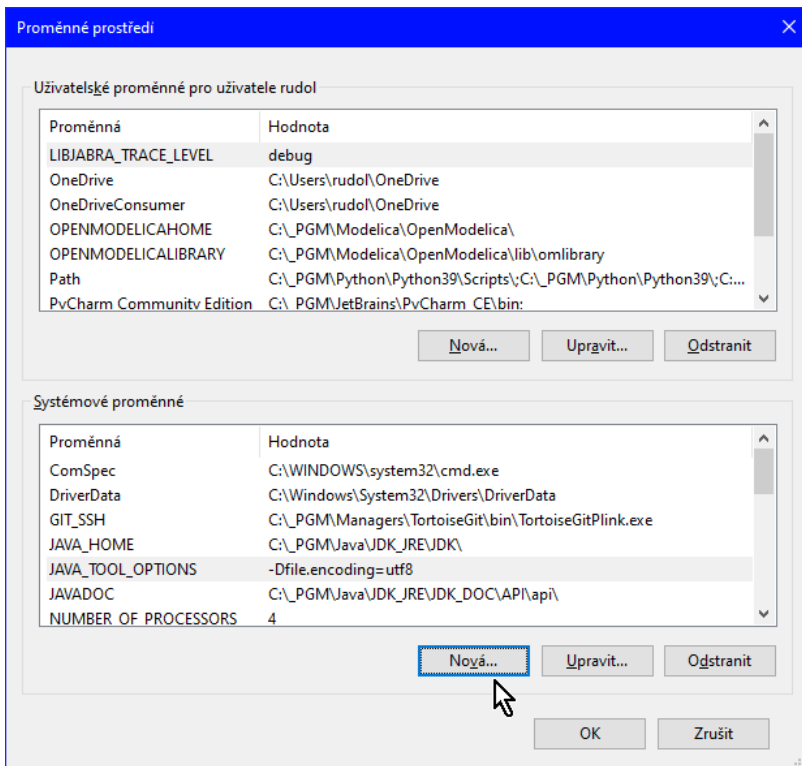
Budete-li chtít toto kódování používat ve všech svých programech v jazyku *Java*, bude nejlepší, když definujete proměnnou prostředí `JAVA_TOOL_OPTIONS`. Dosáhnete toho následovně:

1. V nastaveních systému požádáte o nastavení proměnných prostředí.
2. Otevře se okno **Vlastnosti systému**, v němž stisknete tlačítko **Proměnné prostředí** (viz obrázek [A.1](#)).



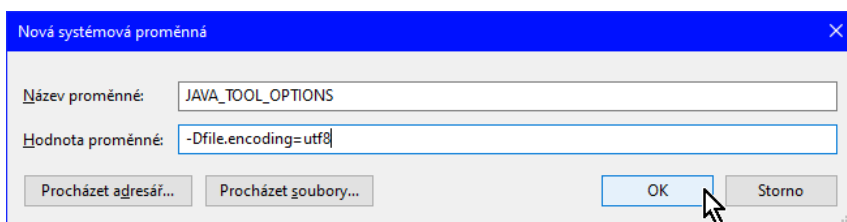
Obrázek A.1:
*Dialogové okno **Vlastnosti systému***

3. Tím otevřete stejnojmenné okno. V něm požádáte o vytvoření nové proměnné (viz obrázek [A.2](#)).



Obrázek A.2:
Dialogové okno *Proměnné prostředí*

4. V následně otevřeném dialogovém okně **Nová systémová proměnná** zadáte název `JAVA_TOOL_OPTIONS` a hodnotu `-Dfile.encoding=utf8`. (viz obrázek [A.3](#)).



Obrázek A.3:
Popisek obrázku

5. Zadání potvrdíte a okna postupně pozavíráte.

Pro vlastní spuštění pak doporučujeme definovat dávkový soubor, který nejprve nastaví kódovou stránku 65001 (číslo stránky s kódováním UTF-8) a teprve pak spustí *JShell*, jak naznačuje obrázek [2.1](#) na straně [41](#).

A.2 Definice substituovaných disků

V operačním systému *Windows* můžete používat 26 logických disků – pro každé písmeno abecedy jeden. Většina uživatelů však používá pouze zlomek tohoto počtu. *Windows* umožňují použít volná písmena pro tzv. **substituované disky**, což jsou složky, které se rozhodnete vydávat za logický disk. Protože o této možnosti většina uživatelů neví, a přitom je to funkce velice užitečná, ukážu vám, jak ji můžete využít. Substituované disky se definují pomocí příkazu:

```
SUBST název_disku substituovaná_složka
```

Nejjednodušší způsob, jak definovat ve *Windows* např. substituovaný disk **P:**, je vložit do složky, kterou budete chtít substituovat jako disk **P:**, dávkový soubor s příkazem k substituci. (Písmeno **P** se pro *Python* hodí nejlépe, ale můžete si vybrat jakékoliv jiné, které je na vašem počítači volné.)

Pokud jste ještě nepracovali s dávkovými soubory, tak vězte, že to jsou obyčejné textové soubory, do nichž zapisujete příkazy pro operační systém. Jejich název může být libovolný, ale musí mít příponu `.bat` (zkratka ze slova `batch` – dávka). V dávkovém souboru budou následující příkazy (na velikosti písmen nezáleží):

```
SUBST P: /d
```

```
SUBST P: .
```

První příkaz má za úkol zrušit případnou doposud nastavenou substituci disku **P:** (není-li v daném okamžiku označený disk substituován, systém vypíše chybovou zprávu, ale jinak se nic nestane), druhý příkaz pak substituuje aktuální složku jako disk **P:**.

Soubor umístíte do složky, z níž budete chtít udělat substituovaný disk. Kdykoliv pak tento dávkový soubor spustíte, dávka substituuje složku, v níž je umístěna, jako příslušný disk. Dávka se přitom spouští obdobně jako aplikace – např. poklepáním na ikonu jejího souboru v *Průzkumníku*.

Kdykoliv od této chvíle budete pracovat s diskem **P:**, budete ve skutečnosti pracovat s obsahem substituované složky. A naopak: cokoliv uděláte s obsahem substituované složky, uděláte zároveň s obsahem disku **P:**.

Budete-li chtít mít danou substituci nastavenou trvale, můžete umístit zástupce dávkového souboru do položky `Po spuštění` ve startovní nabídce. Protože je v každé verzi operačního systému jinde, bude nejlepší, když ji ve startovní nabídce najdete, klepnete na ni pravým tlačítkem a v následně otevřené místní nabídce zadáte **Otevřít**. Tím otevřete okno průzkumníka s touto složkou. Pak v druhém okně průzkumníka otevřete složku s příslušným dávkovým souborem, uchopíte jeho ikonu **PRAVÝM** tlačítkem myši, přesunete ji do složky nabídky a pustíte. Otevře se místní nabídka (ta se otevře, pouze pokud přesouváte soubor pravým tlačítkem myši), ve které zadáte, že zde chcete vytvořit zástupce, a tím celý proces končí.

Abyste mohli složku substituovat jako nějaký disk, nesmí váš operační systém používat disk označený tímto písmenem. Písmeno může být použito nejvýše pro jiný substituovaný disk, protože tuto substituci můžete před nastavením nové substituce zrušit (pro tento případ je v dávkovém souboru první příkaz s parametrem */d* – delete).

Používáte-li operační systém *Windows*, můžete urychlit budoucí vyhledání složky s projekty právě tím, že pro ni zřídíte zvláštní substituovaný disk. Kdykoliv se pak obrátíte na příslušný disk, obrátíte se ve skutečnosti k příslušné složce. Pomocí substituce si tak můžete zkrátit cestu k často používaným složkám.

Příloha B

Význam cizích slov

Jak jsme se zmínili v úvodu, knihu jsme se snažili psát tak, aby ji mohli používat i studenti středních škol a víceletých gymnázií, z nichž mnozí si ještě s řadou relativně běžně používaných cizích slov „netykají“. Snažili jsme se proto do tohoto slovníku zařadit všechna cizí slova, která byla v textu použita. Pokud jsme na nějaké zapomněli, napište, abychom je mohli do příštího vydání doplnit.

alokovat	přidělit zdroje – v OOP se používá nejčastěji v souvislosti s pamětí: alokovat paměť = vyhradit ji a přidělit danému objektu.
arita	počet operandů operátoru (operátory nulární, unární, binární, ternární)
aspekt	hledisko, stanovisko uplatňované při posuzování, zorný úhel
averze	odpor
architektura	uspořádání, skladba nějakého celku
definice	vytvoření
deklarace	veřejné prohlášení; část programu vymezující vlastnosti objektů; tím, že v programu něco deklaruji, oznamuji, jaké to bude, až to bude „živé“
ekvivalent	obdoba
entita	základní objekt zkoumání
explicitně	výslovně, zřetelně
hyperodkaz	zkratka z „ <i>hypertextový odkaz</i> “, což je aktivní odkaz do jiné části dokumentu nebo do jiného dokumentu, kam se dá přejít klepnutím na tento odkaz
hypotetický	založený na předpokladu, podmíněný, nejistý
implicitní	skrytý, předem daný (použije se, nebude-li řečeno jinak)
implementace	způsob provedení
interní	vnitřní, často také pro vnitřní potřebu, neukazovaný navenek
kompilace	překlad (programu)

metodika	1. pracovní postup; 2. nauka o metodě vyučování
mnemonický	např. název – vytvořený tak, aby se snadno zapamatoval
priorita	přednost
netriviální	opak triviální
realizace	provedení, uskutečnění
reference	dobrozdání, doporučení, posudek – např. v životopise se vás ptají na firmy, které by jim mohly na vás dát reference. Řada autorů však tento termín používá (špatně) i jako překlad anglického termínu <i>reference</i> , který se ale správně překládá jako odkaz.
robustní	důkladný (robustní program není citlivý na vadná vstupní data, přerušení síťového připojení ani další vlivy okolí)
seance	zasedání, sezení
separátní	oddělený, zvláštní
specifikace	bližší určení, vymezení něčeho s uvedením přesných rozlišujících údajů
specifikovat	určit, zadat
synonyma	slova, která mají stejný, nebo velmi podobný význam (např. běžet – utíkat)
syntaxe	pravidla pro vytváření přípustných programů; syntaxe se nezabývá tím, jestli bude program pracovat, ale pouze tím, jestli jej smíme takto napsat
triviální	jednoduché na pochopení nebo na realizaci (uskutečnění)
virtuální	zdánlivý

Příloha C

Česko-americký slovník

Možná byste na tomto místě čekali spíše česko-anglický slovník, ale snažil jsem se hned v nadpisu upozornit na to, že liší-li se u některých slov britský a americký překlad (nebo jen pravopis – např. colour × color), používám ten americký, protože počítačová angličtina je prakticky výhradně americká. Pro jistotu uvádím u anglických slovíček v hranatých závorkách i jejich výslovnost. V tomto přepisu označuje písmeno ə zvuk, který vydáváte po vyslovení samotné souhlásky – např. zkratku JDK můžete vyslovit buď dlouze [jédéká], anebo krátce [jədəkə].

balíček	package [pekidž]
barva	color [kalər]
budovatel	builder [bildər]
dědictví	inheritance [inhəritəns]
dědit	inherit [inhərit]
halda	heap [híp]
identifikátor	identifier [ajdentifaiər]
instance	instance [instəns]
jedináček	singleton [singltən]
jednoduchý název	simple name [simpl nejm]
mapa	map [mep]
metoda	method [methəd]
množina	set [set]
objekt	object [obdžikt]
odkaz	reference [refrəns]
parametr	parameter, argument [paramitr, ágjumənt]
pole	array [ərei]
proměnná	variable [variəbl]

přebít	override [ouvəraɪd]
přepravka	crate [kreɪt]
rozhraní	interface [ɪntəfeɪs]
seznam	list [lɪst]
služebník	servant [sɛvənt]
továrna, tovární	factory [fektəri]
třída	class [klás]
vlastnost	property [propəti]
zásobník	stack [stek]
zásobník návratových adres	return stack [rɪtəən stek]
závislost	dependence [dɪpendəns]
zpráva	message [mesɪdʒ]

Výslovnost některých slov v programu

boolean	[búlín]
break	[breik]
byte	[bait]
case	[keɪs]
class	[klás]
double	[dabl]
else	[els]
final	[fainl]
float	[flout]
for	[fór]
implements	[impliments]
instanceof	[ɪnstəns of]
interface	[ɪntəfeɪs]
NetBeans	[netbíns]
new	[njú]
package	[pekɪdʒ]
private	[praɪvɪt]
public	[pʌblɪk]
return	[rɪtəən]
short	[šót]
static	[stetɪk]
super	[súpə]
switch	[swič]
while	[wɪl]

Část F

SEZNAMY

Tato část se vyskytuje pouze v elektronické verzi učebnice. Najdete v ní seznamy odkazů na výpisy programů, obrázky, tabulky a také na odbočky vysazené jako podšeděné bloku

Seznam výpisů programů

Výpis 2.1:	Obsah konzolového okna otevřeného po spuštění dávky !_JShell.bat na mém počítači.....	41
Výpis 2.2:	První tři pokusné úryvky.....	42
Výpis 2.3:	Více úryvků na jednom řádku.....	45
Výpis 2.4:	Výpisy úryvků příkazem /list.....	46
Výpis 2.5:	Výpis všech úryvků příkazem /list -all.....	47
Výpis 3.1:	Komentáře.....	55
Výpis 3.2:	Zadávání celočíselných hodnot.....	55
Výpis 3.3:	Zadávání reálných hodnot.....	57
Výpis 3.4:	Zadávání znaků.....	58
Výpis 3.5:	Literály typu String.....	60
Výpis 3.6:	Textové bloky.....	63
Výpis 3.7:	Logické hodnoty.....	63
Výpis 4.1:	Deklarace proměnných.....	67
Výpis 4.2:	Tři druhy dělení.....	71
Výpis 4.3:	Implicitní a explicitní přetypování.....	72
Výpis 4.4:	Použití složeného přiřazovacího příkazu.....	74
Výpis 4.5:	Inkrementační a dekrementační operátory.....	75
Výpis 4.6:	Porovnávací operátory.....	76
Výpis 5.1:	Práce s atributy.....	86
Výpis 5.2:	Volání metod třídy Math.....	87
Výpis 5.3:	Vytváření nových objektů.....	88
Výpis 6.1:	Ovládání vytvořeného robota.....	92
Výpis 6.2:	Demonstrace vlivu skrývání.....	93
Výpis 6.3:	Opětná aktivace příkazem /reload -restore.....	95
Výpis 6.4:	Načtení knihovny a import potřebných tříd.....	100
Výpis 6.5:	Použití hvězdičkového importu.....	101
Výpis 7.1:	Definice metod kDoubleStep().....	105
Výpis 7.2:	Definice a použití metod doubleStep(Karel) a turnAbout(Karel).....	107
Výpis 7.3:	Definice a použití metody turnWithInfo(Karel, String).....	109
Výpis 7.4:	Definice a použití přetížené metody turnWithInfo(String, Karel).....	111
Výpis 7.5:	Použití přetížených verzí konstruktorů třídy Karel.....	114
Výpis 7.6:	Definice a použití metody wallBehind(Karel) vracející hodnotu.....	115
Výpis 7.7:	Přehled doposud definovaných metod.....	116
Výpis 7.8:	Definice a použití metod forwardClosed(Karel), forwardFree(Karel) a rightFree(Karel) používajících logické výrazy a vracejících hodnotu.....	118
Výpis 8.1:	Použití cyklu while na příkladu metod goToWall(Karel) a turnToNorth(Karel).....	122

Výpis 8.2:	Použití cyklu <code>do...while</code> na příkladu metod <code>goToEmpty</code> (Karel) a <code>markersToWall</code> (Karel)	124
Výpis 8.3:	Použití cyklu <code>for</code> na příkladu metod <code>putNMarkers</code> (Karel, int), <code>putIncreasingMarkers</code> (Karel) a <code>markers</code> (Karel)	126
Výpis 8.4:	Použití nekonečného cyklu na příkladu metody <code>around</code> (Karel)	128
Výpis 9.1:	Použití jednoduchého podmíněného příkazu	131
Výpis 9.2:	Použití úplného podmíněného příkazu	132
Výpis 9.3:	Použití podmíněného výrazu	133
Výpis 9.4:	Zanořeně zarovnaný složený podmíněný příkaz	134
Výpis 9.5:	Doporučené formátování při výběru z více možností	134
Výpis 9.6:	Využití složeného podmíněného příkazu při průchodu bludištěm	135
Výpis 9.7:	Koncepce přepínače (příkazu <code>switch</code>) převzatá z jazyka C	138
Výpis 9.8:	Novější koncepce přepínače (příkazu <code>switch</code>)	138
Výpis 9.9:	Použití přepínacího výrazu	139
Výpis 9.10:	Použití přepínacího příkazu s klasickou syntaxí	140
Výpis 9.11:	Definice metody <code>markersToWall</code> (Karel) používající cyklus s podmínkou uprostřed	141
Výpis 10.1:	Nejjednodušší definice třídy a vytvoření její instance	144
Výpis 10.2:	Definice a použití třídy <code>Light</code> se třemi konstruktory	148
Výpis 10.3:	Upravená definice třídy <code>Light</code> v souboru <code>s10a_Light.jsh</code>	156
Výpis 10.4:	Prověření funkce třídy <code>Light</code> ze souboru <code>s10a_Light.jsh</code>	157
Výpis 12.1:	Třída <code>Light</code> z výpisu 10.3 na straně 156 definovaná tentokrát jako řádná, samostatně definovaná veřejná třída v balíčku <code>eu.pedu.b67._12</code>	171
Výpis 12.2:	Definice třídy <code>ClassTemplate</code> v balíčku <code>eu.pedu.b67.templates</code> představující šablonu definic třídy	178
Výpis 13.1:	Obsah souboru <code>package-info.java</code> v balíčku <code>eu.pedu.b67._12</code>	184
Výpis 13.2:	Začátek definice třídy <code>Light</code> s doplněnými dokumentačními komentáři, ale bez komentářů oddělujících jednotlivé sekce	185
Výpis 14.1:	Možná definice metody <code>moveBy</code> (int, int, IMovable)	194
Výpis 14.2:	Definice interfejsu <code>IMovable</code>	195
Výpis 14.3:	Výňatek z definice třídy <code>Light</code> implementující interfejs <code>IMoveable</code>	197
Výpis 14.4:	Test přesovatelnosti instance třídy <code>Light</code>	198
Výpis 15.1:	Prověření plynulých změn pozice a velikosti instance třídy <code>Light</code>	202
Výpis 15.2:	Definice interfejsu <code>IChangeable</code> v balíčku <code>eu.pedu.b67.canvas_4</code>	207
Výpis 15.3:	Definice interfejsu <code>IVehicle1_1</code> v balíčku <code>eu.pedu.b67.vehicle</code>	209
Výpis 15.4:	Definice třídy <code>Robot1_1</code> implementující interfejs <code>IVehicle1_1</code> (po odebrání komentářů)	210
Výpis 15.5:	Test funkcionality třídy <code>Robot1_1</code>	212
Výpis 16.1:	Definice skriptu <code>importlib.jsh</code>	225
Výpis 16.2:	Definice interfejsu <code>IVehicle1_2</code> (po odebrání komentářů)	226
Výpis 16.3:	Provedené změny ve třídě <code>Robot1_2</code> (po odebrání komentářů)	227
Výpis 16.4:	Test upravené třídy <code>Robot1_2</code>	227
Výpis 17.1:	Ověření funkce příkazu <code>import static</code>	230
Výpis 17.2:	Definice interfejsu <code>eu.pedu.b67.canvasmanager.ICMPaintable</code>	231
Výpis 17.3:	Definice záznamu a práce s ním	235
Výpis 17.4:	Definice třídy (záznamu) <code>Area</code>	236
Výpis 17.5:	Použití přepravek (záznamů) typu <code>Position</code> , <code>Size</code> a <code>Area</code>	237
Výpis 17.6:	Definice abstraktní továrny – interfejsu <code>IVehicleFactory_3</code>	241

Výpis 18.1:	Výpis přidaných částí kódu ve třídě <code>Light</code> v balíčku <code>eu.pedu.b67._18</code>	246
Výpis 18.2:	Prověra nezávislosti blikání instancí třídy <code>Light</code>	247
Výpis 18.3:	Lambda-výraz definovaný prostřednictvím operátoru <code>::</code>	249
Výpis 18.4:	Definice třídy <code>Interval</code>	253
Výpis 18.5:	Test funkcionality třídy <code>Interval</code>	253
Výpis 18.6:	Názvy a abstraktní metody vybraných funkčních interfejsů	256
Výpis 18.7:	Definice a použití lambda-výrazů za použití operátoru <code>::</code>	257
Výpis 18.8:	Přetypování lambda-výrazů	259
Výpis 19.1:	Definice prázdné třídy a přehled jejích členů	265
Výpis 19.2:	Počáteční primitivní definice třídy <code>Square</code>	267
Výpis 19.3:	Definice třídy <code>Square</code> s doplněným konstruktorem	269
Výpis 19.4:	Demonstrace přebíjení metod	271
Výpis 19.5:	Definice metody <code>setSize(int, int)</code> pro třídu <code>Square</code>	272
Výpis 19.6:	Počátek chybové zprávy po žádosti o nastavení velikosti	273
Výpis 19.7:	Definice metody <code>setSize(int)</code> v interfejsu <code>IResizable</code>	273
Výpis 19.8:	Definice metody <code>setSize(int, int)</code> pro třídu <code>Square</code>	273
Výpis 19.9:	Demonstrace zakrývání statických metod	275
Výpis 20.1:	Experimenty s abstraktní třídou	279
Výpis 20.2:	Definice (hlavní) třídy <code>Robot4_4</code> definující otočná vozidla s odebranými komentáři, přetíženými konstruktory a příkazy <code>package</code> a <code>import</code>	283
Výpis 20.3:	Definice abstraktního rodiče jednosměrných podobjektů – třídy <code>ARobot1_4</code>	287
Výpis 20.4:	Definice třídy <code>Robot1E_4</code> definující jednosměrné podobjektoty otočené na východ	289
Výpis 20.5:	Příkazy a úryvky pro test otočných vozidel	290
Výpis 21.1:	Demonstrace chování instancí neměnného typu na příkladu stringů	298
Výpis 21.2:	Definice nešikvné verze metody <code>reverseInWords(int)</code>	302
Výpis 21.3:	Definice upravené verze metody <code>reverseInWords(int)</code>	303
Výpis 21.4:	Definice jednoduchého výčtového typu a jeho použití	304
Výpis 21.5:	Definice výčtového typu <code>Direction</code> robota Karla bez komentářů	305
Výpis 21.6:	<code>Interval</code> využívající komparátor	310
Výpis 22.1:	Ukázka vyhození výjimky	314
Výpis 22.2:	Reakce systému na vyhození výjimky	315
Výpis 22.3:	Definice metody <code>unreachable()</code> demonstrující nedosažitelnost kódu za vyhozením výjimky	316
Výpis 22.4:	Metoda <code>try_catch(int)</code> demonstrující zachycení vyhozené výjimky	318
Výpis 22.5:	Demonstrace funkce bloku <code>try...catch...finally</code>	320
Výpis 22.6:	Definice vlastní kontrolované a nekontrolované výjimky	321
Výpis 22.7:	Demonstrace odezvy překladače při kontrole zabezpečení reakce na vyhození kontrolované výjimky	323
Výpis 22.8:	Definice metody <code>conversion()</code> převádějící kontrolovanou výjimku na nekontrolovanou	324
Výpis 23.1:	Práce s množinou	332
Výpis 23.2:	Seznamy a práce s nimi	334
Výpis 23.3:	Dva způsoby seřazení prvků v seznamu	336
Výpis 23.4:	Základní operace s mapami	337
Výpis 24.1:	Demonstrace základních vlastností polí objektů	340
Výpis 24.2:	Základní vlastnosti polí hodnot primitivních typů	341
Výpis 24.3:	Inicializace pole a jeho tisk	343
Výpis 24.4:	Další způsoby inicializace	344
Výpis 24.5:	Dvojměrné pole	347

Výpis 24.6:	Definice a použití metody <code>average(doube...)</code> využívající proměnný počet argumentů	348
Výpis 24.7:	Převod čísla do tisíce na vyjádření slovy v češtině	350
Výpis 25.1:	Srovnání použití lambda-výrazu a anonymní funkce	358
Výpis 25.2:	Použití metody <code>forEach(Consumer<T>)</code>	360
Výpis 26.1:	Ukázka použití datovodů při generování čísel sportky	370
Výpis 27.1:	Několik příkladů práce s instancemi třídy <code>File</code>	377
Výpis 27.2:	Demonstrace použití výstupních datových proudů	383
Výpis 27.3:	Demonstrace použití vstupních datových proudů	386
Výpis 27.4:	Demonstrace použití třídy <code>Scanner</code>	387
Výpis 28.1:	Definice třídy <code>Guess</code> s jednoduchým programem	389
Výpis 28.2:	Definice třídy <code>Main</code> spouštějící vzorovou demonstrační verzi programu	390
Výpis 28.3:	Obsah souboru <code>MANIFEST.MF</code>	393

Seznam obrázků

Obrázek 1.1: Průběh popularity IDE z „velké trojky“	39
Obrázek 2.1: Konzolové okno otevřené po spuštění dávky !_JShell.bat na mém počítači	41
Obrázek 5.1: Vytvořený svět spolu s robotem.....	88
Obrázek 6.1: Svět Robota po provedení akcí z výpisu 6.1	92
Obrázek 6.2: Stromová struktura balíčků	98
Obrázek 7.1: Výsledná podoba dvorku se čtyřmi roboty.....	114
Obrázek 8.1: Vývojový diagram cyklu s počáteční podmínkou (cyklu while).....	121
Obrázek 8.2: Syntaktický diagram cyklu s počáteční podmínkou (cyklu while).....	122
Obrázek 8.3: Vývojový diagram cyklu s ukončovací podmínkou (cyklu do...while).....	123
Obrázek 8.4: Syntaktický diagram cyklu s ukončovací podmínkou (cyklu do...while)	123
Obrázek 8.5: Vývojový diagram cyklu s parametrem (cyklu for)	125
Obrázek 8.6: Syntaktický diagram cyklu s parametrem (cyklu for)	125
Obrázek 8.7: Vývojový diagram cyklu s podmínkou uprostřed	129
Obrázek 9.1: Vývojový diagram podmíněného příkazu.....	130
Obrázek 9.2: Vývojový diagram úplného podmíněného příkazu.....	132
Obrázek 9.3: Syntaktický diagram obecného podmíněného příkazu	132
Obrázek 9.4: Vývojový diagram složeného podmíněného příkazu	134
Obrázek 9.5: Bludiště s robotem.....	134
Obrázek 9.6: Vývojový diagram přepínače – příkazu switch.....	136
Obrázek 9.7: Syntaktický diagram přepínače – příkazu switch	137
Obrázek 10.1: Okno plátna se třemi vytvořenými světy	148
Obrázek 11.1: Otevření existujícího projektu	160
Obrázek 11.2: Aplikační okno BlueJ po otevření projektu 67_Java zobrazující diagram tříd balíčku eu.pedu.b67.....	161
Obrázek 11.3: Aplikační okno programu BlueJ s diagramem tříd balíčku eu.pedu.b67.canvas_1	162
Obrázek 11.4: Posunutí ikony třídy v diagramu	164
Obrázek 11.5: Změna velikosti ikony třídy v diagramu.....	165
Obrázek 11.6: Výběr skupiny tříd pomocí „výběrového obdélníku“.....	166
Obrázek 11.7: Přesun bloku tříd.....	167
Obrázek 11.8: Přesun bloku tříd.....	168
Obrázek 14.1: Diagram tříd balíčku eu.pedu.b67.canvas_2	193
Obrázek 15.1: Diagram tříd balíčku eu.pedu.b67.canvas_3	201
Obrázek 15.2: Diagram tříd balíčku eu.pedu.b67.canvas_4	205
Obrázek 15.3: Diagram tříd balíčku eu.pedu.b67.canvas_4 po odstranění šipek závislostí a přesunu ikon tříd a interfejsů do přehlednějšího uspořádání	206
Obrázek 16.1: Zmenšení počtu závislostí po aplikaci vzoru Prostředník.....	221
Obrázek 16.2: Podoba diagramu tříd po zavedení abstrakce, které se musejí všechny komunikující objekty přizpůsobit	222

Obrázek 16.3: Diagram tříd balíčku eu.pedu.b67.canvasmanager	225
Obrázek 17.1: Diagram tříd balíčku eu.pedu.b67.geom.....	233
Obrázek 18.1	254
Obrázek 19.1 Rodičovský podobjekt	268
Obrázek 20.1: Diagram tříd balíčku eu.pedu.b67.geom.....	281
Obrázek 20.2: Principiální diagram uspořádání tříd podle návrhového vzoru Stav	282
Obrázek 22.1: Hierarchie dědění výjimek	317
Obrázek 23.1: Hlavní datové typy patřící do knihovny kolekcí.....	327
Obrázek 23.2: Syntaktický diagram „dvojtečkového“ cyklu for (cyklu „for each“).....	331
Obrázek 25.1: Rozdělení interních datových typů.....	353
Obrázek 26.1: Schéma postupu zpracování dat předávaných datovody	366
Obrázek 27.1: Exponenciální nárůst počtu tříd při přidávání funkcností	379
Obrázek 27.2: Třídy po aplikaci dekorátorů	379
Obrázek 28.1: Dialogové okno BlueJ: Vytvoření souboru JAR	391

Seznam tabulek

Tabulka 4.1: Inkrementační a dekrementační operátory	75
Tabulka 27.1: Rodičovské třídy jednotlivých typů datových proudů	381

Seznam odboček – podšeděných bloků

Odbočka – podšeděný blok	24
Historie robota Karla	91
Tisk na standardní výstup	109
Bílé znaky a uspořádání programu	145
Zásobník návratových adres – ZNA	248
Problémy s kódováním znaků	301
Prohlížení obsahu JAR-souborů	388