

edice

začínáme s ...

PROGRAMOVÁNÍ V JAZYKU JAVA

Učebnice pro budoucí profesionální programátory

- » Přehled základních jazykových konstrukcí a principů OOP
- » Návrh objektové architektury pro škálovatelné aplikace
- » Tvorba základního rámce projektu a jeho API pro efektivní práci týmu
- » Zefektivnění vývoje pomocí testovacích scénářů

📄 soubory ke stažení na www.grada.cz



edice
začínáme s ...

Programování v jazyku JAVA

**Učebnice pro budoucí
profesionální programátory**

RUDOLF PECINOVSKÝ

Rudolf Pecinovský

Programování v jazyku Java

Učebnice pro budoucí profesionální programátory

Vydala Grada Publishing, a.s.

U Průhonu 22, Praha 7

obchod@grada.cz, www.grada.cz

tel.: +420 234 264 401

jako svou 10132. publikaci

Odpovědný redaktor Petr Somogyi

Fotografie na obálce Depositphotos/KostyaKlimenko

Grafická úprava a sazba Rudolf Pecinovský

Počet stran 576

První vydání, Praha 2025

Vytiskly Tiskárny Havlíčkův Brod, a. s.

© Grada Publishing, a.s., 2025

Cover Design © Grada Publishing, a. s., 2025

Cover Photo © Depositphotos/KostyaKlimenko

Upozornění pro čtenáře a uživatele této knihy

Všechna práva vyhrazena. Žádná část této tištěné či elektronické knihy nesmí být reprodukována a šířena v papírové, elektronické či jiné podobě bez předchozího písemného souhlasu nakladatele.

Neoprávněné užití této knihy bude trestně stíháno.

Názvy produktů, firem apod. použité v knize mohou být ochrannými známkami nebo registrovanými ochrannými známkami příslušných vlastníků.

Automatizovaná analýza textů nebo dat ve smyslu čl. 4 směrnice 2019/790/EU a použití této knihy k trénování AI jsou bez souhlasu nositele práv zakázány.

ISBN 978-80-271-8054-7 (ePub)

ISBN 978-80-271-8053-0 (pdf)

ISBN 978-80-271-5751-8 (print)

Věnováno mé ženě Jarušce

Stručný obsah

Poděkování	22
Úvod	23
O autorovi	30
Část A Superzáklady	31
Kapitola 1 Předehra	32
Kapitola 2 Prostředí JShell	45
Kapitola 3 Zadávání hodnot	60
Kapitola 4 Proměnné, výrazy a příkazy	74
Část B Začínáme programovat	91
Kapitola 5 Práce s objekty	92
Kapitola 6 Balíčky, knihovny, robot Karel	103
Kapitola 7 Definice metod	119
Kapitola 8 Opakování kódu – cykly	140
Kapitola 9 Rozhodování	152
Část C Objektově orientované programování	171
Kapitola 10 Základy definice třídy	172
Kapitola 11 Uspořádání kódu	188
Kapitola 12 Programátorská dokumentace	202
Kapitola 13 Rozhraní a interfejsy	211
Kapitola 14 Dědění interfejsů	222
Kapitola 15 Návrhové vzory	236
Kapitola 16 Prohlubujeme znalosti	252
Kapitola 17 Lambda-výrazy, funkční interfejsy a generické typy a metody	274
Kapitola 18 Dědění implementace	292
Kapitola 19 Abstraktní třídy	312

Část D Standardní knihovna 329

Kapitola 20 Speciální datové typy.....	330
Kapitola 21 Výjimky.....	348
Kapitola 22 Kontejnery	362
Kapitola 23 Pole	377
Kapitola 24 Interní datové typy	391
Kapitola 25 Datovody.....	405
Kapitola 26 Čtení a ukládání dat	416
Kapitola 27 Vytváření aplikací.....	434

Část E Vývoj aplikace 445

Kapitola 28 Zásady objektové architektury.....	446
Kapitola 29 Návrh základní architektury	455
Kapitola 30 Testovací scénáře – dokončení API	471
Kapitola 31 První a druhá etapa: scénáře HAPPY a BASIC	482
Kapitola 32 Třetí etapa: Implementace navržené architektury	494
Kapitola 33 Čtvrtá etapa: spuštění hry	505
Kapitola 34 Pátá etapa: tvorba světa	514
Kapitola 35 Šestá etapa: definice standardních akcí	523
Kapitola 36 Sedmá a osmá etapa: děláme aplikaci robustní	530
Kapitola 37 Etapy 9–11: přidáváme nestandardní akce	542
Kapitola 38 Závěr: doplnění uživatelského rozhraní	555
Literatura.....	568
Rejstřík.....	571

Část F Přílohy 577

Příloha A Příprava spuštění JShell pod Windows.....	578
Příloha B Význam cizích slov	583
Příloha C Česko-americký slovník.....	585

Část G Seznamy 588

Seznam S1 Výpisy programů	589
Seznam S2 Seznam obrázků	595
Seznam S3 Seznam tabulek	597
Seznam S4 Seznam odboček – podšeděných bloků.....	598

Podrobný obsah

Poděkování	22
Úvod	23
Komu je kniha určena	23
Koncepce výkladu	24
Potřebné vybavení	25
Doprovodné programy	26
Použité typografické konvence	27
Odbočka – podšeděný blok	29
Zpětná vazba	29
O autorovi	30
 Část A Superzáklady	 31
Kapitola 1 Přehled	32
1.1 Trocha historie	32
1.1.1 Co je to program	33
1.1.2 Postupná změna priorit	33
1.1.3 Vývoj metodik programování	35
1.1.4 OOP a OFP	36
1.2 Překladače, interprety, platformy	36
1.2.1 Operační systém a platforma	37
1.2.2 Programovací jazyky	37
1.2.3 Způsoby zpracování programu	38
1.3 Java a její zvláštnosti	39
1.3.1 Java je jazyk i platforma	40
1.4 Vývojové nástroje	41
1.4.1 Základní vývojářská sada	41
1.4.2 IDE	41
1.5 Shrnutí a soubory pro opakování	44
Kapitola 2 Prostředí JShell	45
2.1 Prostředí JShell	45
2.1.1 Spuštění programu JShell	45
2.1.2 JShell v IDE	47
2.2 Úryvky (snippets)	48
2.2.1 Použití proměnných	49
2.2.2 Identifikace úryvků	49
2.2.3 Terminologie: výrazy, příkazy, deklarace, definice	50
2.2.4 Středník	50
2.2.5 Více objektů na řádku, zavlečené chyby	50
2.3 Příkazy prostředí JShell	52
2.3.1 Vyloučení úryvku: /drop	52
2.3.2 Přehled aktivních úryvků: /list	52

2.3.3 Přehled všech úryvků: <code>/list -all</code>	53
2.3.4 Uložení aktivních úryvků: <code>/save <file></code>	54
2.3.5 Uložení všech zadaných úryvků: <code>/save -all <file></code>	55
2.3.6 Uložení dosavadního průběhu seance: <code>/save -history <file></code>	55
2.3.7 Načtení skriptu: <code>/open <file></code>	55
2.3.8 Ukončení seance: <code>/exit</code>	56
2.3.9 Restart: <code>/reset</code>	56
2.3.10 Znovuzavedení: <code>/reload -restore</code>	57
2.3.11 Natavení startovního skriptu: <code>/set -start <file></code>	57
2.3.12 Náповěda: <code>/?</code>	58
2.3.13 Spuštění editoru: <code>/edit</code>	58
2.3.14 Nastavení běhového prostředí: <code>/env</code>	59
2.4 Shrnutí a soubory pro opakování	59
Kapitola 3 Zadávání hodnot	60
3.1 Datové typy	60
3.1.1 Primitivní datové typy	61
3.1.2 Objektové datové typy	62
3.2 Komentáře	63
3.3 Zadávání celočíselných hodnot	64
3.4 Zadávání reálných hodnot	64
3.5 Zadávání znaků	66
3.6 Prázdný odkaz <code>null</code>	68
3.7 Hodnoty typu <code>String</code>	68
3.7.1 Textové bloky	71
3.8 Zadávání logických hodnot	72
3.9 Literály	73
3.10 Shrnutí a soubory pro opakování	73
Kapitola 4 Proměnné, výrazy a příkazy	74
4.1 Pravidla pro tvorbu identifikátorů	74
4.1.1 Používání znaku <code>\$</code>	75
4.2 Deklarace proměnných	76
4.3 Proměnné versus konstanty	77
4.4 Výrazy a jejich terminologie	78
4.5 Operace přiřazení =	79
4.5.1 Asociativita	80
4.6 Aritmetické operace	80
4.6.1 Sčítání	80
4.6.2 Odčítání	80
4.6.3 Násobení	81
4.6.4 Dělení	81
4.7 Přetypování	81
4.8 Volání metod	84
4.9 Závorky	84
4.10 Složené přiřazení	85
4.11 Inkrementační a dekrementační operátory	86
4.12 Porovnávací operátory	88
4.13 Příkazy versus výrazy	89
4.14 Shrnutí a soubory pro opakování	89

Část B Začínáme programovat **91**

Kapitola 5 Práce s objekty	92
---	-----------

5.1 Nejprve trocha teorie.....	92
5.1.1 Principy OOP	92
5.1.2 Objekty	93
5.1.3 Atributy	93
5.1.4 Zprávy	94
5.1.5 Metody	94
5.1.6 Třídy a jejich instance	95
5.1.7 Interní datové typy	96
5.1.8 Třída jako datový typ	96
5.1.9 Třída jako objekt	96
5.1.10 Kontejner	97
5.1.11 Terminologie objektových datových typů	97
5.2 Práce s atributy	98
5.3 Volání metody	98
5.4 Konstruktory a tovární metody	100
5.4.1 Tovární metoda	100
5.4.2 Konstruktory	101
5.4.3 Správa paměti	101
5.5 Shrnutí a soubory pro opakování	102
Kapitola 6 Balíčky, knihovny, robot Karel	103
6.1 Velké programy a jejich problémy	103
6.2 Balíčky	104
6.2.1 Kořenový balíček a stromy balíčků	104
Datová struktura strom	105
6.3 Import objektových typů	105
6.3.1 Import všech typů z daného balíčku – hvězdičkový import	107
6.4 Knihovny a JAR-soubory	108
6.4.1 Začlenění knihovny robota Karla	108
6.4.2 Začlenění knihovny při startu	110
6.5 Robot Karel a jeho svět	110
Historie robota Karla	111
6.5.1 Vytvoření světa	111
6.5.2 Vytvoření robota	113
6.5.3 Akce	113
6.5.4 Testy	114
6.5.5 Zrychlování a skrývání	114
6.5.6 Reakce na zavření okna	116
6.6 Zakázaný balíček java	118
6.7 Shrnutí a soubory pro opakování	118
Kapitola 7 Definice metod	119
7.1 Start JShell na počátcích dalších kapitol	119
7.2 Zásada DRY	120
7.3 Definice a volání metody	120
7.3.1 Povinný středník	121
7.3.2 Příklad jednoduché metody pro robota	121
7.4 Blok příkazů	122
7.5 Metody s parametry	123
7.5.1 Parametry	123
7.5.2 Argumenty	123
7.5.3 Metody robota versus metody interaktivního prostředí	124
7.6 Metody s více parametry	125
Tisk na standardní výstup	126
7.7 Lokální proměnné a konstanty	126
7.7.1 Proměnné lokální v bloku	127
7.7.2 Demonstrace použití konstant	127

7.7.3 Dočasná existence proměnných v blocích	128
Zásobník návratových adres – ZNA	128
7.8 Přetěžování metod	129
7.9 Ještě jednou robot Karel a jeho svět	130
7.9.1 Třídy RobotWorld a RobotWindow	130
7.9.2 Přetížené tovární metody tříd RobotWorld a RobotWindow	131
7.9.3 Přetížené konstruktory třídy Karel	132
7.10 Metody vracející hodnotu	133
7.11 Přehled aktuálně definovaných metod v JShell	134
7.12 Logické výrazy	135
7.12.1 Příklad	136
7.13 Předávání hodnotou a odkazem	137
7.14 Shrnutí a soubory pro opakování	139
Kapitola 8 Opakování kódu – cykly	140
8.1 Cykly	140
8.2 Cyklus s počáteční podmínkou – cyklus while	141
8.3 Cyklus s koncovou podmínkou – cyklus do...while	143
8.4 Cyklus s parametrem – cyklus for	145
8.4.1 Příklad	146
8.5 Nekonečný cyklus	149
8.6 Cyklus s podmínkou uprostřed	149
8.7 Vnořování cyklů	150
8.8 Shrnutí a soubory pro opakování	151
Kapitola 9 Rozhodování	152
9.1 Jednoduchý podmíněný příkaz	153
9.2 Úplný podmíněný příkaz	154
9.3 Podmíněný výraz	155
9.4 Složený podmíněný příkaz	156
9.5 Přepínač – příkaz/výraz switch	158
9.5.1 Pravidla	160
9.5.2 Přepínací výraz – výraz switch	162
9.5.3 Využití klasické verze příkazu switch	163
9.6 Cyklus s podmínkou uprostřed – příkaz break	164
9.7 Příkaz continue	166
9.8 Rekurze	167
9.9 Shrnutí a soubory pro opakování	169

Část C Objektově orientované programování **171**

Kapitola 10 Základy definice třídy	172
10.1 Vytváříme vlastní třídu	172
Bílé znaky a uspořádání programu	173
10.2 Viditelnost tříd a jejich členů: public , private	174
10.3 Výměna knihoven – knihovna tvarů	174
10.4 Diagramy tříd	176
10.5 Konstruktory	177
10.5.1 Podrobnosti o konstruktorech	177
10.6 Atributy	179
10.6.1 Definice atributů	179
10.6.2 Modifikátor final	179
10.7 Konstruktory a parametr this	180
10.7.1 Kvalifikace parametrem this	181

10.8	Magické hodnoty	182
10.9	Modifikátor static	183
10.9.1	Definice třídy počítající své instance	183
10.10	Zděděné metody, metoda toString()	184
10.11	Výsledná definice a test.....	184
10.11.1	Upravená definice	185
10.11.2	Test upravené definice	185
10.12	Shrnutí a soubory pro opakování	187
Kapitola 11	Uspořádání kódu	188
11.1	Uspořádání programů ve složce 77_JAVA	188
11.1.1	Vývojové prostředí	189
11.1.2	Problémy s překladem	189
11.1.3	Knihovna Java77_NZ2	189
11.1.4	Úprava prostředí pro JShell	190
11.2	Konvence pro názvy balíčků v Javě	190
11.3	Samostatně definované třídy.....	190
11.4	Příkazy package a import	192
11.5	Zapouzdření a skrývání implementace	193
11.6	Entity programu	194
11.7	Rozhraní versus implementace.....	194
11.7.1	Signatura versus kontrakt	195
11.7.2	API.....	195
11.8	Atributy versus vlastnosti; přístupové metody	196
11.8.1	Možné kombinace a výhody	196
11.8.2	Pravidla pro názvy.....	197
11.9	Uspořádání jednotlivých prvků v těle třídy	197
11.9.1	Motivace pro zavedení některých zásad	198
11.10	Prázdná standardní třída.....	199
11.11	Shrnutí a soubory pro opakování	201
Kapitola 12	Programátorská dokumentace.....	202
12.1	Komentáře a dokumentace.....	202
12.1.1	Proč psát srozumitelné programy	202
12.1.2	Odsazování	204
12.1.3	Dokumentační komentáře	204
12.2	Pomocné značky pro tvorbu dokumentace	205
12.3	Dokumentace balíčku	206
12.4	Ukázka dokumentované třídy	206
12.5	Zobrazení dokumentace.....	208
12.5.1	Obsah a uspořádání dokumentace.....	208
12.6	Zakomentování a odkomentování části programu	210
12.7	Shrnutí a soubory pro opakování	210
Kapitola 13	Rozhraní a interfejs	211
13.1	Motivace	211
13.2	Rozhraní versus interfejs versus interface	212
13.2.1	Interfejs a jeho instance	212
13.3	Použití v programu	213
13.3.1	Knihovní balíček shapes77.canvas_2	214
13.4	Použití interfejsu na příkladu.....	215
13.4.1	Počáteční úvahy	215
13.5	Definice interfejsu.....	217
13.6	Implementace interfejsu třídou	218
13.6.1	Anotace @Override	218
13.6.2	Ověření funkcionality	220

13.7 Shrnutí a soubory pro opakování	221
Kapitola 14 Dědění interfejsů	222
14.1 Problém více rolí	222
14.2 Současná implementace více interfejsů	223
14.2.1 Realizace v kódu	224
14.3 Dědění interfejsů	225
14.3.1 Trocha teorie o dědění	225
14.3.2 Dědění a přetypování	226
14.3.3 Aplikace dědění interfejsů – balíček canvas_4	227
14.4 Deklarace rodičů interfejsu v kódu	229
14.5 Vlastnosti interfejsů na příkladu Multishape	229
14.5.1 Podrobnosti o třídě Multishape	230
14.5.2 Mnohotvar se skládá z kopií	230
14.6 Příklad: definice vozidla	230
14.6.1 Kontrakt	231
14.6.2 Test vytvořeného vozidla	234
14.6.3 Problém plátna	235
14.7 Shrnutí a soubory pro opakování	235
Kapitola 15 Návrhové vzory	236
15.1 Úvod do návrhových vzorů	236
15.2 Přehled vzorů, s nimiž jsme se již setkali	238
15.2.1 Knihovni třída (Utility class)	238
15.2.2 Statická tovární metoda (Static factory method)	238
15.2.3 Jedináček (Singleton)	239
15.2.4 Náhradník (Substitute)	239
15.2.5 Výčtový typ (Enumerated type)	239
15.2.6 Multiton, Originál	240
15.2.7 Služebník (Servant)	240
15.2.8 Prázdný objekt (Null Object)	241
15.2.9 Prototyp (Prototype)	241
15.3 Teoretické pozadí nové koncepce kreslení	242
15.3.1 Návrhový vzor Prostředník (Mediator)	242
15.3.2 Inverze závislostí	243
15.3.3 Návrhový vzor Pozorovatel (Observer), hollywoodský princip	245
15.4 Správce plátna – CanvasManager	246
15.4.1 Balíček canvasmanager	247
15.4.2 Import klíčových tříd knihovny	247
15.5 Nová verze třídy Robot	249
15.6 Shrnutí a soubory pro opakování	251
Kapitola 16 Prohlubujeme znalosti	252
16.1 Statický import	252
16.2 Další členy interfejsů	253
16.2.1 Statické konstanty	253
16.2.2 Statické metody	255
16.2.3 Implicitní metody	255
16.2.4 Soukromé metody	255
16.3 Návrhový vzor Adaptér (Adapter)	256
16.4 Návrhový vzor Přepřavka – třídy typu record	257
16.4.1 Třídy typu záznam (record)	257
16.4.2 Záznamy v používané knihovně	258
16.5 Návrhový vzor Šablonová metoda	260
16.5.1 Příklad	261
16.6 Návrhový vzor Abstraktní továrna	262
16.6.1 Motivace	262

16.6.2	Návrhový vzor Tovární metoda.....	263
16.6.3	Co je abstraktní továrna	263
16.6.4	Použití.....	263
16.6.5	Tovární třída převádí konstruktory a statické členy na instanční.....	263
16.7	Podrobnosti o konstruktorech	265
16.7.1	Dvě části těla konstrukturu instancí – inicializační bloky.....	266
16.7.2	Inicializace konstant	268
16.7.3	Konstruktor třídy	268
16.7.4	Konstanty vyhodnotitelné v době překladu	271
16.8	Shrnutí a soubory pro opakování	273
Kapitola 17	Lambda-výrazy, funkční interfejsy a generické typy a metody	274
17.1	Nezávislé opakování zadané akce	274
17.1.1	Možná řešení.....	275
17.2	Syntaxe lambda-výrazů.....	275
17.2.1	Aplikace na světlo	276
17.3	Lambda-výrazy a lokální proměnné.....	278
17.4	Lambda-výrazy zastupující metody	279
17.5	Funkční interfejsy	280
17.6	Generické datové typy a metody.....	281
17.6.1	Motivace	281
17.6.2	Syntaxe zadávání a používání generických typů	281
17.6.3	Specifika generických typů Javy	282
17.6.4	Omezení typových parametrů	282
17.6.5	Příklad: Interval	282
17.6.6	Typové parametry s více předky.....	284
17.6.7	Potomci a předci generických typů	284
17.6.8	Generické metody	285
17.7	Funkční interfejsy – pokračování.....	285
17.7.1	Demonstrace použití	287
17.8	Lambda-výrazy nelze přetypovat na Object přímo	289
17.9	Shrnutí a soubory pro opakování	291
Kapitola 18	Dědění implementace	292
18.1	Tři druhy dědění.....	292
18.1.1	Přirozené (nativní) dědění	292
18.1.2	Dědění rozhraní	293
18.1.3	Dědění implementace	293
18.1.4	LSP – Liskov Substitution Principle.....	294
18.1.5	Shrnutí	294
18.2	Základy dědění tříd	294
18.2.1	Univerzální (pra)rodič Object.....	295
18.3	Co dědíme od třídy Object.....	295
18.3.1	Class-objekt	296
18.3.2	Úplný název tříd v JShell	297
18.3.3	Přehled veřejných členů zděděných od třídy Object	297
18.4	Definice třídy s předkem – Square	298
18.4.1	Rodičovský podobjekt	298
18.4.2	Volání rodičovského konstrukturu	299
18.5	Přebíjení metod.....	301
18.5.1	Implementace přebíjení	303
18.6	Skrytá nebezpečí: úprava třídy Square	305
18.7	Modifikátor final v procesu dědění	307
18.7.1	Virtuální versus konečné metody	307
18.7.2	Obyčejné versus konečné třídy	308
18.8	Zakrývání statických metod.....	308
18.8.1	Metody interfejsů se nezakrývají	309

18.9	Jediný implementační předek	309
18.10	Přístupová práva	310
18.11	Akceptovatelné modifikace signatur potomků	310
18.12	Shrnutí a soubory pro opakování	311
Kapitola 19	Abstraktní třídy	312
19.1	Abstraktní třídy a jejich role v dědické hierarchii	312
19.1.1	Definice abstraktních tříd a metod	314
19.1.2	Experimenty s abstraktní třídou	314
19.2	Účel abstraktních tříd	316
19.3	Zavedení abstraktních tříd do projektu	316
19.4	Návrhový vzor Stav	317
19.5	Aplikace na příklad s vozidlem	318
19.5.1	Hlavní třída vozidla – třída Robot4_4	319
19.5.2	Stavově nezávislé metody	322
19.5.3	Stavově závislé metody	322
19.5.4	Společný rodič jednosměrných tříd – třída ARobot1_4	322
19.5.5	Jednostavové (jednosměrné) třídy	324
19.5.6	Test	326
19.6	Shrnutí a soubory pro opakování	327

Část D Standardní knihovna

329

Kapitola 20	Speciální datové typy	330
20.1	Třída Object	330
20.2	Objekty reprezentující hodnotu (ORV) a objekty reprezentující entitu (ORE)	331
20.2.1	Metody equals(Object)	332
20.2.2	Metoda hashCode()	333
20.2.3	Proměnné a neměnné ORV	333
20.3	Primitivní a obalové typy	335
20.4	Třída String	336
20.4.1	Možné problémy při práci s některými znaky	336
20.4.2	Interfejs java.lang.CharSequence	336
	Problémy s kódováním znaků	337
20.4.3	Třídy StringBuilder a StringBuffer	338
20.5	Výčtové typy – třídy typu enum	339
20.5.1	Složitější příklad: Direction	341
20.6	Třída Throwable	342
20.7	Třída Thread	342
20.8	Třída java.util.Optional<T> & spol.	343
20.8.1	Motivace	343
20.8.2	Alternativní řešení	343
20.9	Třída java.util.Random	344
20.10	Interfejs java.lang.Comparable<T>	345
20.11	Interfejs java.util.Comparator<T>	346
20.12	Shrnutí a soubory pro opakování	347
Kapitola 21	Výjimky	348
21.1	Co to jsou výjimky	348
21.2	Nejdůležitější výjimky	349
21.3	Vyhození výjimky	350
21.3.1	Reakce systému na vyhození výjimky	351
21.4	Výjimky a nedosažitelný kód	353
21.5	Hierarchie dědění výjimek	353

21.6 Zachycení vyhozené výjimky.....	355
21.7 Společný úklid – blok finally	356
21.8 Definice vlastních výjimek	358
21.9 Kontrolované výjimky	359
21.10 Převedení kontrolované výjimky na nekontrolovanou	360
21.11 Shrnutí a soubory pro opakování	361
Kapitola 22 Kontejnery	362
22.1 Co je to kontejner v Javě	362
22.2 Kategorizace kontejnerů	363
22.2.2 Kontejnery objektů	363
22.3 Knihovna kolekcí	364
22.4 Deklarujte typy co nejobecněji	366
22.5 Collection<E> – kolekce	367
22.6 Dvojtečkový cyklus for(:)	368
22.7 Množiny – Set<E>	369
22.8 Seznamy – List<E>	371
22.8.1 Pořadí prvků	371
22.8.2 Příklad: seřazení prvků v seznamu	373
22.9 Slovníky a mapy – Map<K, V>	374
22.9.1 Pohledy	376
22.10 Shrnutí a soubory pro opakování	376
Kapitola 23 Pole.....	377
23.1 Představení	377
23.1.1 Deklarace polí	378
23.2 Atributy a metody polí	378
23.3 Pole jako kontejner	378
23.3.1 Pole odkazů na objekty	379
23.3.2 Pole hodnot primitivních typů	379
23.3.3 Hlídnání mezi polí	381
23.3.4 Inicializace polí v deklaraci	382
23.3.5 Inicializace vytvářeného pole	383
23.3.6 Neinicializovaná pole objektových typů	384
23.4 Vícerozměrná pole	385
23.4.1 Obdélníková pole	385
23.4.2 Neobdélníková pole	386
23.4.3 Inicializace vícerozměrného pole	387
23.5 Metody s proměnným počtem argumentů	387
23.6 Pole, kolekce a moderní programování	388
23.7 Závěrečný příklad	388
23.8 Shrnutí a soubory pro opakování	390
Kapitola 24 Interní datové typy	391
24.1 Přehled	391
24.1.1 Terminologie	391
24.1.2 Společné charakteristiky	392
24.1.3 Použití	392
24.1.4 Jedna hladina soukromí	393
24.2 Globální interní (členské) datové typy	393
24.3 Vnořené datové typy	394
24.4 Vnitřní třídy	395
24.5 Lokální třídy	395
24.5.1 Pojmenované lokální třídy	396
24.5.2 Anonymní třídy	396
24.6 Návrhový vzor Iterátor	397

24.6.1	Interfejsy <code>java.util.Iterator<E></code> a <code>java.lang.Iterable<E></code>	398
24.6.2	Vnější (sekvencní) a vnitřní (dávkové) iterátory	399
24.6.3	Iterátory a dvojtečkový cyklus <code>for(:)</code>	400
24.7	Definice vlastní iterovatelné třídy	401
24.7.1	Generátor náhodných stringů	402
24.7.2	Použití v cyklu <code>for(:)</code>	403
24.7.3	Použití interního iterátoru <code>forEach()</code>	404
24.8	Shrnutí a soubory pro opakování	404
Kapitola 25	Datovody	405
25.1	Analogie	405
25.2	Druhy operací	406
25.3	Princip práce datovodu	406
25.4	Specifické vlastnosti	407
25.5	Datové typy související s datovody	408
25.6	Vytváření datovodů	408
25.6.1	Kolekce	408
25.6.2	Pole	409
25.6.3	Interfejs <code>java.util.stream.Stream</code>	409
25.7	Konceptní příklad	409
25.8	Operace s daty v datovodu	410
25.8.1	Koncové operace	410
25.8.2	Průběžné operace	411
25.8.3	Částečně průběžné operace	412
25.9	Kolektory	412
25.9.1	Příklad	413
25.10	Shrnutí a soubory pro opakování	415
Kapitola 26	Čtení a ukládání dat	416
26.1	Koncepce čtení a ukládání dat	416
26.2	Soubory: bleskové opakování	417
26.2.1	Soubor, souborový systém, cesta	417
26.2.2	Relativní, absolutní a kanonická cesta	418
26.2.3	Substituované disky ve Windows	418
26.3	Třída <code>java.io.File</code>	419
26.3.1	Instanční metody	419
26.4	Návrhový vzor Dekorátor	422
26.4.1	Motivace	422
26.4.2	Princip funkce	423
26.5	Rozdělení datových proudů	424
26.6	Zápis textů	425
26.6.1	Splachování a zavírání proudů	426
26.6.2	Přidávání dat na konec existujícího souboru	427
26.6.3	Příklad	427
26.7	Čtení textů	429
26.7.1	Příklad	430
26.8	Třída <code>java.util.Scanner</code>	431
26.9	Shrnutí a soubory pro opakování	433
Kapitola 27	Vytváření aplikací	434
27.1	Superjednoduchá demonstrační aplikace	434
27.2	Základní pravidla pro větší aplikace	437
27.3	Hlavní třída aplikace	437
27.3.1	Argumenty příkazového řádku	438
27.3.2	Definice třídy <code>Main</code>	438
27.3.3	Definice třídy <code>GuessIO</code>	440

27.4 JAR-soubor	441
Prohlížení obsahu JAR-souborů	441
27.4.1 Soubor MANIFEST.MF	442
27.4.2 Vytvoření JAR-souboru s aplikací	442
27.5 Spuštění aplikace	443
27.6 Shrnutí a soubory pro opakování	444

Část E Vývoj aplikace 445

Kapitola 28 Zásady objektové architektury	446
28.1 Předmluva	446
28.2 Architektura	447
28.3 Hlavní zásady návrhu	447
28.3.1 Připravenost na změny	448
28.3.2 CRIDP – maximální přehlednost	448
28.3.3 KISS – maximální jednoduchost	448
28.3.4 YAGNI – žádné zbytečnosti	448
28.3.5 SoC – jediný zodpovědný	449
28.3.6 SRP – jediná zodpovědnost	449
28.4 Antivzory	449
28.5 Návrh programu	450
28.6 Druhy vytvářených sólo-objektů	451
28.7 Dva způsoby návrhu	452
28.7.1 Návrh shora dolů	452
28.7.2 Návrh zdola nahoru	452
28.7.3 Porovnání	453
28.8 UML – diagram tříd	454
28.9 Důležitost čitelnosti programu	454
28.10 Shrnutí a soubory pro opakování	454
Kapitola 29 Návrh základní architektury	455
29.1 Proč právě textová konverzační hra	455
29.1.1 Proč hra vyvíjená pod frameworkem	456
29.2 Organizace balíčků a záznamy průběhu vývoje	456
29.2.1 Umístění studentských úloh	457
29.2.2 Umístění tříd frameworku	457
29.3. Koncepce vyvíjené aplikace	457
Co to je h-objekt	458
29.3.1 Nestandardní akce	459
29.4 Zadání	459
29.4.1 Vyjmenované požadavky	460
29.5 Účastníci – objekty vystupující ve hře	461
29.6 Správci skupin objektů	465
29.6.1 Správci v naší aplikaci	465
29.7 Specifikace jednotlivých účastníků	466
29.8 Společné API	466
29.9 Schopnosti jednotlivých účastníků	467
29.9.1 IGame	467
29.9.2 IWorld	468
29.9.3 INamed	468
29.9.4 IItemContainer	469
29.9.5 IPlace	469
29.9.6 IItem	469
29.9.7 IBag	469
29.9.8 IAction	470

29.10 Shrnutí a soubory pro opakování	470
Kapitola 30 Testovací scénáře – dokončení API	471
30.1 Jak testovat	471
30.1.1 Programování řízené testy	471
30.1.2 Jednotkové, integrační a regresní testy	472
30.1.3 Možnosti testování naší hry	473
30.2 Scénáře	473
30.3 Kroky scénáře – třída ScenarioStep	474
30.4 Výčtový typ TypeOfStep	476
30.5 Výčtový typ TypeOfScenario	476
30.6 Třída Scenario	477
30.7 Postup vývoje a požadavky na scénáře	477
30.7.1 Jednotlivé etapy vývoje	478
30.8 Doplnění API	479
30.8.1 Interfejs IAuthor	479
30.8.2 Interfejs IPortal	479
30.8.3 Třída BasicActions	480
30.8.4 Balíček game77.testers	481
30.8.5 Interfejs IGame	481
30.9 Shrnutí a soubory pro opakování	481
Kapitola 31 První a druhá etapa: scénáře HAPPY a BASIC	482
31.1 Balíček etapy	482
31.2 Návrh portálu	482
31.3 Správce scénářů – ScenarioManager	485
31.4 Test na hladině Level.HAPPY	488
31.5 Druhá etapa – Scénář BASIC	490
31.5.1 Balíček etapy	490
31.5.2 Návrh portálu	490
31.5.3 Úpravy třídy game77.ck1b_duplet.ScenarioManager	490
31.5.4 Test na hladině Level.DUPLET	492
31.6 Shrnutí a soubory pro opakování	493
Kapitola 32 Třetí etapa: Implementace navržené architektury	494
32.1 Tvorba kostry vyhovující API	494
32.2 Návrh tříd v balíčku ck1c_architecture	495
32.2.1 Třída Portal	495
32.2.2 Třída ANamed	495
32.2.3 Třída Action	496
32.2.4 Třída Item	497
32.2.5 Třída AItemContainer	497
32.2.6 Třída Bag	498
32.2.7 Třída Place	499
32.2.8 Třída World	499
32.2.9 Třída Game	501
32.3 Test na hladině Level.ARCHITECTURE	503
32.4 Shrnutí a soubory pro opakování	504
Kapitola 33 Čtvrtá etapa: spuštění hry	505
33.1 Návrh tříd v balíčku ck1d_start	505
33.2 Tři druhy objektů	505
33.3 Delegování zodpovědnosti	506
33.4 Statické metody třídy Action	507
33.4.1 Metody isActive() a stop()	507

33.4.2	Statická metoda <code>executeCommand(String)</code>	507
33.4.3	Definice má být krátká	508
33.4.4	Pomocné statické metody	509
33.5	Úpravy třídy <code>SceneManager</code>	509
33.5.1	Problematika statických konstant	510
33.5.2	Vlastní úprava	510
33.6	Spuštění testu na hladině <code>Level.START</code>	512
33.7	Shrnutí a soubory pro opakování	513
Kapitola 34	Pátá etapa: tvorba světa	514
34.1	Balíček <code>ck1e_world</code> a třída <code>Portal</code>	514
34.2	Vytváříme prostory	514
34.2.1	Inicializace prostoru	515
34.3	Modifikace třídy <code>ItemContainer</code>	516
34.4	Další úpravy třídy <code>SceneManager</code>	517
34.4.1	Simulace chodu hry podle scénáře	519
34.5	Modifikace třídy <code>World</code>	519
34.5.1	Hledání chyby	520
34.6	Test na hladině <code>Level.WORLD</code>	521
34.7	Shrnutí a soubory pro opakování	522
Kapitola 35	Šestá etapa: definice standardních akcí	523
35.1	Balíček <code>ck1f_basic</code> a třída <code>Portal</code>	523
35.2	Definice akcí	523
35.3	Funkce <code>executeStandardCommand(String)</code>	524
35.4	Definice výkonného kódu jednotlivých akcí	525
35.4.1	Akce <code>Jdi</code> a metoda <code>move(String[])</code>	526
35.4.2	Akce <code>Vezmi</code> a metoda <code>take(String[])</code>	527
35.4.3	Akce <code>Polož</code> a metoda <code>put(String[])</code>	528
35.4.4	Akce <code>?</code> a metoda <code>help(String[])</code>	528
35.4.5	Kontrolní test	529
35.5	Shrnutí a soubory pro opakování	529
Kapitola 36	Sedmá a osmá etapa: děláme aplikaci robustní	530
36.1	Nesplněné body zadání	530
36.2	Balíček pro hladinu <code>Level.TRIPLET</code>	531
36.2.1	Proč samostatná etapa	531
36.3	Chybový scénář	531
36.3.1	Co vše se má zkontrolovat	532
36.3.2	Reakce na pokus o nekorektní spuštění	533
36.3.3	Vlastní definice chybového scénáře	533
36.3.4	Test chybového scénáře	533
36.4	Přechod na hladinu <code>Level.MISTAKES</code>	534
36.4.1	Chybné spuštění hry	534
36.5	Problémy s argumenty standardních akcí	535
36.5.1	Nezadané argumenty	535
36.5.2	Finalizace změny prostoru	536
36.5.3	Problémy se zvedáním h-objektu	536
36.6	Změny v definici h-objektů a práce s nimi	537
36.6.1	Nová definice třídy <code>Item</code>	537
36.6.2	Úprava definice prostorů	538
36.6.3	Úprava batohu	539
36.6.4	Dokončení metody <code>take(String[])</code>	539
36.6.5	Dotažení metody <code>put(String[])</code>	541
36.7	Shrnutí a soubory pro opakování	541

Kapitola 37 Etapy 9–11: přidáváme nestandardní akce	542
37.1 Předehra	542
37.1.1 Plánovaný postup –etapy dalšího vývoje.....	542
37.2 RUNNING: Rozchození scénáře HAPPY	543
37.2.1 Úprava scénáře HAPPY.....	543
37.2.2 Úprava konstruktoru akcí	543
37.2.3 Výchozí podoba definic nestandardních akcí.....	545
37.3 QUADRUPLER: Příprava testu robustnosti	545
37.3.1 Nutné podmínky pro korektní zadání nestandardních akcí.....	546
37.3.2 Zanesení nutných podmínek	546
37.3.3 Zanesení nutných podmínek do scénářů	547
37.3.4 Průběžný test	550
37.3.5 Jaká chybná zadání se budou testovat	550
37.3.6 Přidání scénáře MISTAKE_NS	550
37.4 WHOLE: Rozchození výsledného kódu hry	550
37.4.1 Definice metod <code>conditions()</code> a <code>tests()</code> ve třídě <code>Game</code>	552
37.4.2 Definice atributů <code>CONDITIONS</code> a <code>TESTS</code> ve třídě <code>Action</code>	552
37.4.3 Definice kódu nestandardních akcí	553
37.5 Shrnutí a soubory pro opakování	554
Kapitola 38 Závěr: doplnění uživatelského rozhraní	555
38.1 Hlavní třída aplikace	555
38.2 Jednoduché textové uživatelské rozhraní.....	555
38.2.1 Možnost opakovaného spouštění	557
38.2.2 Přidání informací o aktuálním stavu	558
38.3 Opět trocha teorie	558
38.3.1 Vzor Model-View-Controller (MVC)	558
38.3.2 Kompaktnější varianta: M(V+C).....	559
38.4 Volitelné uživatelské rozhraní	559
38.4.1 Interfejs <code>IUI</code>	559
38.4.2 Definice třídy <code>Main2</code>	560
38.4.3 Úprava metod <code>run(...)</code> a <code>multirun(...)</code>	561
38.4.4 Definice třídy <code>Console</code>	563
38.5 Vytvoření primitivního GUI	563
38.5.1 Modalita dialogových oken.....	563
38.5.2 Třída <code>javax.swing.JOptionPane</code>	564
Návrhový vzor Fasáda.....	564
38.5.3 Třída <code>PrimitiveGUI</code>	565
38.6 Několik dalších námětů	566
38.7 Shrnutí a soubory pro opakování	567
Literatura.....	568
Rejstřík	571

Výpis 38.4: Definice interfejsu **IUI**

```

1 package game77.ck1l_user_io;
2
3 /* Interfejs {@code IUI} definuje požadované rozhraní objektů,
4  * jejichž prostřednictvím bude aplikace kounikovat s uživatelem. */
5 public interface IUI
6 {
7     /* Zobrazí uživateli odpověď hry a požádá jej o zadání dalšího příkazu. */
8     String getCommand(String answer);
9
10    /* Zobrazí zadaný string -- většinou odpověď hry. */
11    void showAnswer(String answer);
12
13    /* Zeptá se uživatele a očekává odpověď ANO/NE, nebo její ekvivalent. */
14    boolean askQuestion(String question);
15 }

```

38.4.2 Definice třídy **Main2**

Abyste mohli porovnávat změny, nebudu třídu **Main** dále upravovat, ale vytvořím její kopii nazvanou **Main2** a tu upravím tak, aby umožňovala metodám **run** a **multirun** komunikovat s uživatelem prostřednictvím zvoleného komunikačního objektu a aby bylo možné zadat, zda se součástí odpovědi stane i informace o aktuálním stavu hry.

Začneme úpravou metody **main(String[])**. V pozdějších dobách ji můžeme upravit tak, že potřebná nastavení bude možné nastavit jako argumenty příkazového řádku, jak jsme si vysvětlovali v pasáži [27.3.1 Argumenty příkazového řádku](#) na straně [438](#) a následně ukazovali ve výpisu [27.3](#).

V metodě definujeme následující pomocné proměnné:

- Proměnná **ui** bude uchovávat odkaz na objekt, jehož prostřednictvím bude aplikace komunikovat s uživatelem.
- Proměnná **game** bude uchovávat odkaz na hru, která se hraje.
- Hodnota logické proměnné **state** bude určovat, zda se má odpověď hry doplňovat o informace o aktuálním stavu hry.
- Proměnná **details** bude uchovávat odkaz na kód, který bude doplňovat odpověď hry o dodatečné informace o jejím stavu. Protože se tyto informace po zpracování každého příkazu mění, potřebujeme něco, co je bude na požádání operativně zjišťovat.

Zdrojový kód takto upravené metody si můžete prohlédnout ve výpisu [38.5](#).

V metodě by vás mohla zaujmout inicializace proměnné **details**. V závislosti na hodnotě proměnné **state** se do ní bude ukládat buď odkaz na lambda-výraz spouštějící metodu **currentState(IGame)**, anebo odkaz na lambda-výraz vracující prázdný string.

Jak vidíte na řádce [7](#), metodě **multirun(...)** od minula přibyly dva parametry: první obsahuje odkaz na kód, který může předat k odpovědi hry další informace, druhý obsahuje odkaz na komunikační objekt, jehož prostřednictvím bude aplikace komunikovat s uživatelem.

Výpis 38.5: Upravená definice metody `main(String[])` ve třídě `Main2`

```

1 public static void main(String[] args) {
2     IUI      ui = Console.get();
3     IGame    game = new game77.ck1k_whole.Portal().game();
4     boolean  state = true; //Budeme-li chtít vypisování aktuálního stavu
5     Supplier<String> details = (state ? () -> currentState(game)
6                               : () -> "");
7     multirun(game, details, ui);
8 }

```

38.4.3 Úprava metod `run(...)` a `multirun(...)`

Pojďme se tedy podívat, jak bychom měli po těchto změnách upravit definice metod `run(...)` a `multirun(...)`, kterým na jednu stranu přibudou parametry, ale na druhou stranu se v nich usnadní rozhodování.

Jak se můžete přesvědčit ve výpisu 38.6, definice obou metod se zavedením komunikačního objektu opravdu zjednoduší. Zájemci je mohou porovnat s definicemi ve výpisech 38.1 na straně 556 a 38.2 na straně 557.

Výpis 38.6: Upravené definice metod `run(...)` a `multirun(...)` ve třídě `Main2`

```

1 private static void run(IGame game, Supplier<String> details, IUI ui) {
2     String command = "";
3     for(;;) {
4         String answer = game.executeCommand(command);
5         if (! game.isActive()) {
6             ui.showAnswer(answer); //Už bez přidáných informací
7             return;
8         }
9         command = ui.getCommand(answer + details.get());
10    }
11 }
12
13 public static void multirun(IGame game, Supplier<String> details, IUI ui) {
14     for(;;) {
15         run(game, details, ui); //Spuštění jednoho běhu hry
16         boolean answer = ui.askQuestion("Chcete si zahrát ještě jednou?");
17         if (! answer) {
18             ui.showAnswer("Ještě jednou děkuji za hru.\nNa shledanou.\n");
19             return;
20         }
21         ui.showAnswer("Dobře, zahrajeme si ještě jednou.");
22     }
23 }

```

V definici metody `run(...)` bych chtěl upozornit na použití lambda-výrazu `details` na řádce 9, kde se k odpovědi hry přičte string, jenž je návratovou hodnotou lambda-výrazu definovaného v metodě `main(String[])` a obsahuje případné informace o aktuálním stavu hry, které mohou sloužit jako částečná nápověda.

Příkaz na řádce 6 vypisující odpovědi hry už po závěrečném příkazu tyto informace o aktuálním stavu hry k odpovědi hry nepřidává, protože předpokládáme, že po skončení hry je uživatel již nepotřebuje.

Výpis 38.7: Definice třídy *Console* v modulu *interfaces*

```

1  package game77.ck1l_user_io;
2
3  import java.util.Scanner;
4
5  public class Console
6      implements IUI
7  {
8      private static final Scanner SCANNER = new Scanner(System.in);
9
10     private static final Console CONSOLE = new Console();
11     public static Console get() { return CONSOLE; }
12     private Console() {}
13
14     @Override
15     public String getCommand(String answer) {
16         System.out.println(answer);
17         System.out.print("=====\nZadáte: ");
18         String command = SCANNER.nextLine();
19         System.out.print("-----\n");
20         return command;
21     }
22
23     public void showAnswer(String answer) {
24         System.out.println(answer + "=====\n");
25     }
26
27     public boolean askQuestion(String question) {
28         char answer;
29         for(;;) {
30             System.out.println("=====\n"
31                 + "Chcete si zahrát ještě jednou (A/N): ");
32             String line = SCANNER.nextLine().trim();
33             if (! line.isEmpty()
34                 && "01ANan".indexOf(line.charAt(0)) != -1) {
35                 answer = line.charAt(0);
36                 break; //Výskok z cyklu ----->
37             }
38             System.out.println("-----\n"
39                 + "Odpověď musí začínat některým ze znaků \"01ANan\"\n"
40                 + "Zkuste odpovědět znovu.");
41         }
42         return ("1Aa".indexOf(answer) != -1);
43     }
44 }
```


38.4.4 Definice třídy **Console**

Abychom mohli vše rozběhnout, potřebujeme alespoň jeden komunikační objekt. Definujeme třídu **Console**, která bude implementovat interfejs **IUI** a její instance bude zprostředkovávat komunikaci s uživatelem prostřednictvím standardního vstupu a výstupu.

Vzhledem k tomu, že budeme potřebovat pouze jedinou instanci této třídy, definujeme ji podle návrhového vzoru *Jedináček – Singleton*. Její definici si můžete prohlédnout ve výpisu [38.7](#). Doufám, že vám nebude vadit, že jsem v ní srazil k sobě definici atributu s jedináčkem, tovární metody a soukromého konstruktora.

38.5 Vytvoření primitivního GUI

Aplikace je tedy připravena na výběr z několika verzí uživatelského rozhraní. Pojďme se nyní podívat na to, jaké nástroje nám *Java* nabízí pro návrh alespoň primitivního GUI.

Java definuje ve standardní knihovně balíček **javax.swing** obsahující sadu modulů definujících základní nástroje pro vytváření GUI. Protože se jedná o relativně ucelenou sadu, bývá často označován jako knihovna nebo framework. Tuto knihovnu nyní použijeme.

38.5.1 Modalita dialogových oken

Než se ale pustíme do návrhu našeho GUI, měli bychom si nejprve ujasnit pojem modalita. Dialogová okna, jejichž prostřednictvím uživatel komunikuje s aplikací, můžeme rozdělit do dvou skupin:

- **Modální okna** zastaví jakoukoliv další komunikaci s aplikací do doby, než je uživatel řádně ukončí. Modální bývají např. okna pro otevření či uložení souboru. Dokud nezádáte, jaký soubor chcete otevřít, resp. uložit, aplikace se s vámi „nebaví“.

To, že se zastaví komunikace s uživatelem, ale neznamená, že se zastaví běh celé aplikace. Ta může dále běžet včetně případného zobrazování průběžných výsledků. Můžete např. spustit nějakou animaci a otevřít dialogové okno s výzvou: „Až se pokocháš, stiskni OK!“ Uživatel si může například prohlížet běžící animaci a stiskem OK pak celou aplikaci zavřít.

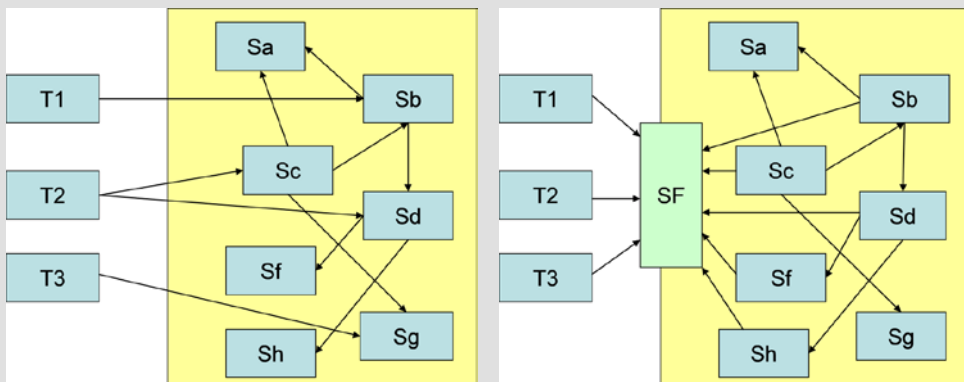
- **Nemodální okna** sice vyskočí, ale aplikace s uživatelem dále komunikuje. Nemodální bývají v řadě aplikací okna pro vyhledávání a nahrazování textu. Poté, co zadáte příkaz, který okno otevře, můžete odskočit do aplikace, tam např. vybrat do schránky text, který chcete vyhledat, vrátit se do okna a tam jej zadat. Nemodální okno může zůstat otevřené a vy můžete dále s aplikací pracovat.

38.5.2 Třída `javax.swing.JOptionPane`

Knihovna `swing` představuje poměrně rozsáhlý komplex vzájemně provázaných tříd, který umožňuje vytvářet komplexní grafické uživatelské rozhraní. Uživatelé však v řadě případů nepotřebují využívat veškeré její schopnosti a stačí jim vytvářet jednoduchá dialogová okna. Pro tento účel je v ní definována třída `JOptionPane`, kterou bychom mohli označit za implementaci návrhového vzoru *Fasáda* (viz podšeděný blok).

Návrhový vzor *Fasáda*

Návrhový vzor *Fasáda* použijeme ve chvíli, kdy nějaký systém začíná být pro své uživatele příliš složitý vzhledem k oblasti úloh, které chtějí s jeho pomocí řešit. Doporučuje nahradit sadu rozhraní jednotlivých subsystémů sjednoceným rozhraním zastupujícím celý systém. Definuje tak rozhraní vyšší úrovně, které usnadní využívání podsystémů. Cílem je zjednodušit rozhraní celého systému a snížit počet tříd, s nimiž musí uživatel přímo či nepřímo komunikovat.



Používá se např. u grafických knihoven, kdy uživatelům, kteří nepotřebují vytvářet složité strukturovaná okna, nabídne sadu funkcí vyvolávajících předpřipravená jednoúčelová okna.

Třída `JOptionPane` definuje 65 metod, z nichž nás budou zajímat následující tři, které otvírají modální okna:

`showMessageDialog(Component parentComponent, Object message)`

Otevře dialogové okno se zobrazeným textovým podpisem parametru `message` a tlačítkem `OK`. Parametr `parentComponent` určuje rámec, v němž se dialog zobrazí, ale může se zde zadat prázdný odkaz `null`.

`int showConfirmDialog(Component parentComponent, Object message)`

Otevře dialogové okno se zobrazeným textovým podpisem parametru `message` a tlačítky `Yes`, `No` a `Cancel`. Parametr `parentComponent` má stejný význam jako u předchozí

metody. Podle toho, které tlačítko bylo stisknuto, vrací jednu ze čtyř hodnot definovaných jako konstantní atributy třídy `JOptionPane`: `YES_OPTION`, `NO_OPTION` a při zavření okna hodnotu `CANCEL_OPTION`.

String showInputDialog(Object message)

Otevře dialogové okno s vypsanou výzvou a vstupním polem pro zadání údajů. Tato metoda má na rozdíl od předchozích přetíženou verzi, která nepožaduje zadání rámce, což obvykle znamená, že je okno vycentrováno na primární obrazovce.

38.5.3 Třída `PrimitiveGUI`

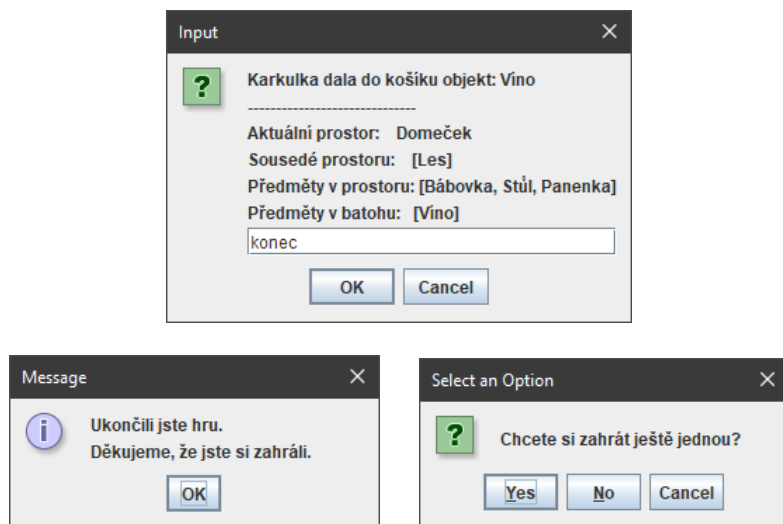
S pomocí uvedených metod třídy `JOptionPane` nyní můžeme snadno definovat třídu `PrimitiveGUI`, které bude realizovat primitivní grafické rozhraní pro naši hru. Její definici najdete ve výpisu [38.8](#).

Výpis 38.8: Definice třídy `PrimitiveGUI`

```

1  package game77.ck1l_user_io;
2
3  import javax.swing.JOptionPane;
4  import static javax.swing.JOptionPane.showInputDialog;
5  import static javax.swing.JOptionPane.showMessageDialog;
6  import static javax.swing.JOptionPane.showConfirmDialog;
7
8  public class PrimitiveGUI
9      implements IUI
10 {
11     private static final PrimitiveGUI SINGLETON = new PrimitiveGUI();
12     public static PrimitiveGUI get() { return SINGLETON; }
13     private PrimitiveGUI() {}
14
15     @Override
16     public String getCommand(String answer) {
17         String command = showInputDialog(answer);
18         if (command == null) { return ""; }
19         return command;
20     }
21
22     public void showAnswer(String answer) {
23         showMessageDialog(null, answer);
24     }
25
26     public boolean askQuestion(String question) {
27         int answer = showConfirmDialog(null, question);
28         return answer == JOptionPane.YES_OPTION;
29     }
30 }
```

Podobu zobrazovaných dialogových oken v průběhu hry si můžete prohlédnout na obrázku [38.1](#).



Obrázek 38.1:

Dialogové okno s výzvou k zadání údajů

38.6 Několik dalších námětů

Hra, kterou jsme doposud vytvořili, je velmi prostoduchá. Nicméně vytvořený program by mohl být základem dokonalejších programů, na nichž by bylo možné vysvětlovat další pravidla návrhu a programátorské obraty. Pojďme si projít některé náměty.

Převod pod kvalitní grafické uživatelské rozhraní

Po primitivním GUI je to jistě to první, co vás napadne. Nicméně popis tohoto tématu by spolu s výkladem potřebných programátorských obrátů vydal sám o sobě na samostatnou publikaci.

Zdokonalení h-objektů

Doposud jsme v prefixu názvu h-objektů pouze oznamovali, zda daný objekt je či není přenositelný. Mohli bychom ale zadat i jemnější dělení. Budou-li např. náš batoh představovat ruce, mohli bychom rozdělit předměty na ty, k jejichž přenesení stačí jedna ruka, a na ty těžší, k jejichž přenesení jsou potřeba obě ruce.

Vhodným nastavením prefixu můžete zadat i další charakteristiky, např. že se jedná o alkoholický nápoj, který proto smějí konzumovat pouze dospělé osoby.

H-objekty – prostory

Mezi h-objekty mohou být i takové, které jsou současně prostory, do nichž se musíte dostat speciálním příkazem. V aktuálním prostoru může být např. truhla, do níž se dostanete zadáním příkazu **otevři**. Pak se truhla stane aktuálním prostorem, z něž můžete h-objekty odebírat, nebo je do něj naopak ukládat.

Hra může také vyžadovat, abyste k otevření truhly použili klíč, který musíte nejprve najít a případně se jej proti nějakému odporu zmocnit.

Rozhovor

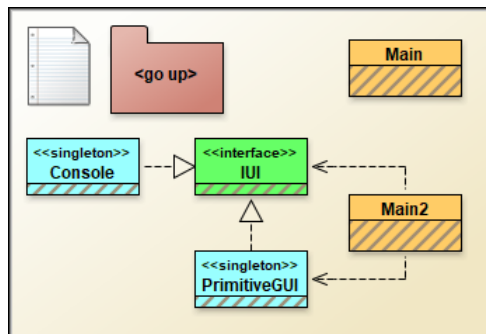
Speciálním druhem akce je taková, při níž zapředeme rozhovor s nějakým h-objektem – osobou či věcí, od níž něco potřebujeme. Můžeme např. oslovit kouzelného dědečka, od něj se dozvíme, že má hlad, my mu dáme buchtu, on nám dá píšťalku, kterou uspíme psa, jenž hlídá princeznu, apod.

Jednotlivé věty rozhovoru nejsou příkazy, ale na druhou stranu rozhovor může být prokládán příkazy. Program tak střídavě přepíná mezi dvěma režimy: rozhovorem a zadáváním příkazů.

38.7 Shrnutí a soubory pro opakování

Zadávané příkazy i obsah výpisů se záznamem komunikace s interpretem probíhající v této kapitole najdete v souborech nazvaných `s38__Uživatelské_rozhraní`. Aktuální podoba zdrojových kódů je zkopírována do balíčku `game77.ck1l_user_io`.

UML diagram aktuálního stavu tohoto balíčku si můžete prohlédnout na obrázku [38.2](#).



Obrázek 38.2:

Diagram tříd aktuálního stavu balíčku `game77.ck1l_user_io`

Literatura

- [1] *Akademický slovník cizích slov*. Praha: Academia, 1995. ISBN 80-200-0497-1.
- [2] BERGIN, Joseph, STEHLIK, Mark, ROBERTS, Jim, PATTIS, Richard: *Karel J Robot: A Gentle Introduction to the Art of Object-Oriented Programming in Java*. Dreamsongs Press, 2013, ISBN 978-0-9705795-1-5
- [3] BERGIN, Joseph: *Beyond Karel J Robot: A Gentle Introduction to the Art of Object-Oriented Programming*. Slant Flying Press, 2023, ISBN 978-1-9401130-9-8
- [4] BERGIN, Joseph: *Beyond Karel J Robot: A Gentle Introduction to the Art of Object-Oriented Programming, Volume 2*. Software Tools, 2013, ISBN 978-0-9851543-0-1
- [5] FOOTE Brian, YODER Joseph, *Big Ball of Mud Fourth Conference on Patterns Languages of Programs (PLoP '97/EuroPLoP '97)* Monticello, Illinois, září 1997. Dostupné na adrese <http://www.laputan.org/mud/>.
- [6] FOWLER, Martin. *UML Distilled: A Brief Guide to the Standard Object Modeling Language, 3rd Edition*. Addison-Wesley Professional, 2003. 192 s. (Přeloženo v [7]) ISBN 0-321-19368-7.
- [7] FOWLER Martin. *Destilované UML*. Praha: Grada Publishing, 2009. 174 s. (Překlad [6]). ISBN 978-80-247-2062-3.
- [8] FOWLER, Martin. *Refactoring – Improving the Design of Existing Code*. Addison-Wesley, 2000. 430 s. (Přeloženo v [9]) ISBN 0-201-48567-2.
- [9] FOWLER, Martin. *Refaktoring – Zlepšení existujícího kódu*. Praha: Grada Publishing, 2003. 394 s. (Překlad knihy [8]) ISBN 80-247-0299-1.
- [10] GAMMA E., HELM R., JOHNSON R., VLISSIDES J.: *Design Patterns: Elements of Reusable Object-oriented Software*. Boston: Addison-Wesley, 2001. ISBN 0-201-63361-2.
- [11] GOSLING J., JOY B., STEELE G., BRACHA G., BUCKLEY A., SMITH D., BIERMAN G.: *The Java® Language Specification – Java SE 14 Edition*. Dostupné na adrese <https://docs.oracle.com/javase/specs>
- [12] HAVELKA Antonín, PECINOVSKÝ Rudolf: *JUnit 5 – Jednotkové testování na platformě Java*. Praha: Grada Publishing, 2018. ISBN 978-80-271-0733-9.

- [13] LEXA Ivan: *Shora dolů nebo zdola nahoru?* Programování Ostrava, 1987. Dostupné na adrese <http://prog-story.technicalmuseum.cz/index.php/m-virtualni-sbirky-tm-v-brne/programovani-a-tvorba-sw-ostrava/1985-1994/1987-programovani-ostrava/2985-1987-shora-dolu-nebo-zdola-nahoru>.
- [14] McLAUGHLIN Brett D., POLLICE Gary WEST Dave: *Head First Object-Oriented Analysis and Design*. O'Reilly Media 2006. ISBN 978-0-596-00867-3.
- [15] PATTIS Richard E.: *Karel The Robot: A Gentle Introduction to the Art of Programming*. John Wiley & Sons, 1981. ISBN 0-471-59725-2.
- [16] PECINOVSKÝ R.: *Java 5.0 – Novinky jazyka a upgrade aplikací*, Brno: Computer Press, 2005, ISBN 80-251-0615-2. Dostupné na adrese <http://knihy.pecinovsky.cz/java5novinky/>
- [17] PECINOVSKÝ Rudolf: *Návrhové vzory – 33 vzorových postupů pro objektové programování*. Brno: Computer Press, 2007. ISBN 978-80-251-1582-4.
- [18] PECINOVSKÝ R.: *Myslíme objektově v jazyku Java*. Praha: Grada Publishing, 2009. ISBN 978-80-247-2653-3.
- [19] PECINOVSKÝ R.: *Java 7 – Učebnice objektové architektury pro začátečníky*. Praha: Grada Publishing, 2012. ISBN 978-80-247-3665-5 (tištěná), ISBN 978-80-247-8325-3 (PDF), ISBN 978-80-247-8326-0 (epub).
- [20] PECINOVSKÝ R.: *Java 8 – Učebnice objektové architektury pro mírně pokročilé*. Praha: Grada Publishing, 2014. ISBN 978-80-247-4638-8 (tištěná).
- [21] PECINOVSKÝ Rudolf: *OOP a Java 8: Návrh a vývoj složitějšího projektu vyhovujícího zadanému rámci*. Praha: Nakladatelství Evy a Tomáše Brucknerů, 2015. ISBN 978-80-87924-03-7.
- [22] PECINOVSKÝ Rudolf: *Java 9 – JShell*. Praha: Grada Publishing, 2017. ISBN 978-80-271-9785-9 (pdf), ISBN 978-80-271-9786-6 (epub).
- [23] PECINOVSKÝ Rudolf: *Java 14 – Kompletní příručka jazyka*. Praha: Grada Publishing, 2020. ISBN 978-80-271-1369-9.
- [24] PECINOVSKÝ R.: *Java 21 – Kompletní příručka jazyka*. Grada Publishing 2023. ISBN 978-80-271-7041-8 (pdf), ISBN 978-80-271-7042-5 (ePub), ISBN 978-80-247-0599-6 (print).
- [25] POST E.: *Real Programmers Don't Use PASCAL*. Datamation 1983. Dostupné na adrese <https://www.ee.ryerson.ca/~elf/hack/realmen.html>, český překlad *Opravdoví programátoři nepoužívají Pascal* dostupný na adrese <http://www.logix.cz/michal/humornik/Pojidaci.Kolacu.xp>.
- [26] PRAVDOVÁ Markéta, SVOBODOVÁ Ivana: *Akademická příručka českého jazyka, 2., rozšířené vydání*. Praha: Academia, 2019. ISBN 978-80-200-2947-8.

- [27] VIRIUS Miroslav: *Java – programování podprocesů (vláken)*. Praha: Grada Publishing, 2021. ISBN 978-80-271-3266-9 (print), 978-80-271-4253-8 (pdf), 978-80-271-4254-5 (ePub)
- [28] VERMEER Brian: *IntelliJ IDEA Dominates the IDE Market with 62% Adoption among JVM Developers*. publikováno 5. 2. 2020 na <https://snyk.io/blog/intellij-idea-dominates-the-ide-market-with-62-adoption-among-jvm-developers/>.

Rejstřík

\$

\$
používání, 75

/

/?, 58
/drop, 52
/edit, 58
/env, 59, 108
/exit, 56
/list, 52
/open, 55
/reload, 57
/reset, 56
/save, 54
/set, 57

@

@author, 205
@FunctionalInterface, 285
@Override, 218
@param, 205
@returns, 205
@throws, 205
@version, 205

A

Abstraktní továrna, 262
Adaptér, 256
adventura
koncepce, 457
agilní metodika: viz metodika:
agilní
agilní programování: viz
programování: agilní
AHA-příklad, 25
akce, 458
nestandardní, 459, 461

standardní, 459, 461
alternativa, 135
zkrácená, 135
AND, 135
anonymní třída, 392
anotace
@FunctionalInterface, 285
aplikace
hlavní třída, 434
soubor JAR, 442
spouštějící třída, 437
vytvoření, 441
argument, 84, 123
proměnný počet, 387
arita, 78
asociativita, 78
atribut, 93, 196
definice, 179
instanční, 95
statický, 95
viditelnost, 174

B

Babbage, 32
bajtkód, 38
balíček, 104
java, 118
kořenový, 104
básnička, 472
BBM, 446
bílý znak, 173
binární operátor, 78
blok
inicializační
statický, 269
příkazů
lokalita proměnných, 127
textový, 71
blok příkazů, 122, 141

BlueJ, 42
boolean, 61
literál, 72
byte, 62

C

cesta, 417
class-objekt, 96
classpath, 108
class-path, 108
class-soubor, 40
CRIDP, 202, 448
CTC, 271
cyklus, 140
do...while, 143
for, 145
hlavička, 141
nekonečný, 149
s parametrem, 145
s podmínkou uprostřed, 149
tělo, 141
while, 141

Č

číslo
exponentový tvar, 65
člen
statický, 97
viditelnost, 174
člen instanční, 97
člen třídy, 96
členy, 393
čtyřtečka, 279

D

datové členy, 393
datovod, 405
druhy operací, 406

vytvoření, 408
 z kolekce, 408
 z pole, 409
 datový typ: *viz* typ: datový
 interní, 391
 vnořený, 394
 dědění, 292
 implementace, 293
 přetypování, 226
 přirozené, 292
 rozhraní, 293
 definice, 50
 deklarace, 50
 dekompozice, 452
 Dekorátor, 422
 dekrementace --, 86
 dělení, 81
 Design Pattern: *viz* návrhový
 vzor
 diamantový operátor: *viz*
 operátor: diamantový
 disjunkce, 135
 exkluzivní, 136
 zkrácená, 135
 disk
 substituovaný, 418, 581
 dokumentace, 202
 tagy: *viz* dokumentace –
 značky
 značky, 205
 zobrazení, 208
 double, 61
 literál, 64
 DRY, 120
 dvojrozměrné pole, 385

E

Eclipse, 43
 entita, 194
 escape sekvence, 66
 exception, 348
 exponent, 65
 exponentový tvar čísla, 65

F

Fasáda, 564
 final, 77, 179
 float, 62
 funkce, 98

vnější, 148
 vnitřní, 148
 funkční členy, 393

G

garbage collector, 102
 generická metoda, 285
 globální typ, 391
 GoF, 237

H

halda, 63, 101
 historie, 32
 hladina soukromí, 393
 Hoare, 343
 hodnota
 magická, 182
 přípustná, 60
 hra
 akce, 458
 koncepce, 457
 příkaz, 458

Ch

char, 62
 literál, 66
 CharSequence, 336
 chyba
 syntaktická, 76
 zavlečená, 51

I

ID úryvku, 49
 IDE, 41
 identifikátor
 pravidla, 74
 if, 153
 implementace
 dědění implementace, 293
 skrývání, 193
 versus rozhraní, 194
 import, 105, 192
 statický, 252
 typu, 106
 initor, 265
 inkrementace ++, 86
 instance, 95
 int, 61

literál, 64
 integrační test, 472
 IntelliJ IDEA, 43
 interface, 212
 interfejs, 212
 @FunctionalInterface, 285
 dědění, 222, 225
 definice, 217
 deklarace rodičů, 229
 funkční, 280
 tabulka, 287
 CharSequence, 336
 Iterator, 398
 interní datový typ, 391
 interpret, 36, 38
 inverze závislosti, 243
 Iterator, 398

J

JAR, 108
 JAR-soubor, 337, 441, 442
 Java, 39
 jazyk
 hybridní, 38
 programovací, 37
 jazyky
 vyšší, 35
 JDK, 41
 jediná hladina soukromí, 393
 Jedináček, 95, 239
 Jednoduchá tovární metoda,
 238
 jednotkový test: *viz* test:
 jednotkový
 JRE, 41
 JShell, 42, 45
 příkaz, 52
 spuštění, 45
 středník, 50

K

Karel, 110
 historie, 111
 skrývání, 114
 zrychlování, 114
 karelcz
 Karel, 110
 KISS, 448
 klávesnice, 578

knihovna

 tvarů, 174

Knihovná třída, 238

kód

 uspořádání, 190

kolekce: *viz* kontejner

 převod na datovod, 408

komentář, 202

 dokumentační, 204

kompilátor: *viz* překladač

konjunkce, 135

 zkrácená, 135

konstanta, 77

 vyhodnotitelná v době

 překladu, 160

 vyhodnotitelná v době

 překladu, 271

konstruktor, 101, 177, 178

 instancí, 265

 this, 180

 třídy, 268

kontejner, 97, 362, 363

 mapa, 366

 množina, 364

 seznam, 365

 slovník, 366

kontrakt, 195

kvalifikace, 394

L

lambda-výraz

 syntaxe, 275

literál

 boolean, 72

 double, 64

 char, 66

 int, 64

 long, 64

 null, 68

 String, 68

lokalita

 proměnných

 v bloku příkazů, 127

lokální třída, 391

long, 62

 literál, 64

LSP, 294

M

M(V+C), 559

mantisa, 65

mapa, 366

metatřída, 97

metoda, 84, 94

 abstraktní, 313

 argument, 84

 defaultní, 255

 definice, 120

 equals(Object), 332

 generická, 285

 implicitní, 255

 instanční, 95

 konečná, 308

 nastavovací, 196

 návratová hodnota, 98

 parametr, 84

 parametry, 125

 println, 125

 přetěžování, 129

 přetížená, 99

 přístupová, 196

 s parametrem, 123

 statická, 95

 tovární, 101

 viditelnost, 174

 virtuální, 307

 volání, 69, 84

 volání metody, 84, 98

 vrácení hodnoty, 133

 zjišťovací, 196

metodika

 agilní, 447

mezera

 sémantická, 35

mnohotvar, 229

množina, 364

modalita, 563

modifikátor

 final, 179

 private, 174

 public, 174

 static, 183

modulární programování, 35

Multiton, 240

MVC, 558

N

násobení, 81

návrh

 postupný, 452

návrhový vzor, 236, *viz* vzor

 návrhový

 Dekorátor, 422

 Přpravka, 362, 363, 393

název

 jednoduchý, 104

 úplný, 104

negace, 135

NetBeans, 42

nonekvivalence, 136

nulární operátor, 78

null, 62

 literál, 68

null type, 62, 68

O

obalový typ, 335

obdélníkové pole, 385

Object, 62

objekt, 93

 class-objekt, 96

objektově orientované

 programování: *viz* OOP

objektový typ, 61

objekty

 datové, 505

 řídící, 506

 výkonné, 506

odčítání, 80

odkaz

 prázdný, 343

odkomentování části

 programu, 210

odsazování, 204

okno

 konzolové, 46

 modální, 563

 nemodální, 563

OOP, 35, 36

 principy, 92

operace, 78

 dělení, 81

 násobení, 81

 odčítání, 80

 sčítání, 80

- operační systém, 37
 - operand, 78
 - operátor, 78
 - (čtyřtečka), 279
 - binární, 78
 - čtyřtečka, 279
 - dekrementační --, 86
 - dělení, 81
 - diamantový, 282
 - infixový, 78
 - inkrementační ++, 86
 - kulaté závorky (), 84
 - násobení, 81
 - nulární, 78
 - odčítání, 80
 - postdekrementační, 86
 - postfixový, 78
 - postinkrementační, 86
 - predekrementační, 86
 - prefixový, 78
 - preinkrementační, 86
 - přetypování, 81
 - přiřazení, 79
 - složený, 85
 - sčítání, 80
 - ternární, 78
 - unární, 78
 - Optional, 343
 - OR, 135
 - ORE, 331
 - ORV, 331
- P**
- package, 192, viz balíček
 - paměť
 - správa, 101
 - správce, 102
 - parametr, 84, 123
 - typový
 - omezení, 282
 - s více předky, 284
 - parta čtyř, 237
 - platforma, 40
 - HWOS, 37
 - platné číslice, 62
 - POBLIOCHA, 203
 - podmíněný příkaz: viz příkaz:
 - podmíněný
 - složený, 156
 - podpis
 - textový, 184, 331
 - podprogram, 84
 - pole, 377
 - dvo rozměrné, 385
 - hlídání mezí, 381
 - hodnot primitivních typů, 379
 - inicializace
 - v definici, 383
 - v deklaraci, 382
 - obdélníkové, 385
 - odkazů, 379
 - převod na datovod, 409
 - vícerozměrné, 385
 - inicializace, 387
 - polymorfismus, 212
 - Posluchač, 245
 - Pozorovatel, 245
 - pravidlo
 - suché, 120
 - Prázdný objekt, 241
 - prázdný odkaz, 343
 - primitivní typ, 61, 335
 - primitivum, 61
 - print, 69
 - println, 69
 - priorita, 78
 - private, 174
 - procedura, 98
 - program
 - definice, 33, 94
 - interpretovaný, 38
 - objektově orientovaný
 - definice, 94
 - odkomentování, 210
 - překládání, 38
 - zakomentování, 210
 - programovací jazyk, 37
 - programování
 - agilní, 448
 - funkcionální, 36
 - modulární, 35
 - objektově orientované, 36, viz OOP
 - strukturované, 35
 - proměnná, 49
 - deklarace, 76
 - pomocná, 48
 - proměnný počet argumentů, 387
 - Prostředník, 242
 - Prototyp, 241
 - překladač, 36, 38
 - přepínač, 158
 - Přepravka, 257, 362, 363, 393
 - přetypování, 81
 - při dědění, 226
 - příkaz, 50, 458
 - if, 153
 - if...then, 153
 - if-then-else, 155
 - import, 192
 - package, 192
 - podmíněný
 - jednoduchý, 153
 - tělo, 153
 - úplný, 154
 - switch, 158
 - přiřazení
 - složené, 85
 - public, 174
- R**
- rekurze, 167
 - robot, 249
 - RobotWindow, 130
 - RobotWorld, 130
 - rozhraní, 212
 - versus implementace, 194
- Ř**
- řetězec
 - textový, 63
- S**
- samostatná aplikace, 441
 - scénář, 473
 - HAPPY, 474
 - šťastný, 474
 - sčítání, 80
 - SDK, 41
 - sémantická mezera, 35
 - seznam, 365
 - short, 62
 - signatura, 195
 - metody, 242
 - skrývání implementace: viz implementace: skrývání
 - slovník, 366

slovo
 klíčové, 74
 Služebník, 240
 Smalltalk, 96
 SoC, 449
 soubor
 JAR, 442, *viz* JAR-soubor
 zdrojový, 40
 souborový systém: *viz* systém:
 souborový
 SRP, 449, 506, 558
 static, 97, 183
 Statická tovární metoda, 238
 Stav, 317
 stereotyp, 177
 String, 62
 literál, 68
 sčítání stringů, 80
 StringBuffer, 338
 StringBuilder, 338
 stroj
 virtuální, 38, 60
 strukturované programování,
 35
 substituovaný disk: *viz* disk:
 substituovaný
 System, 125
 systém
 operační, 37
 souborový, 417

Š

Šablonová metoda, 260

T

TDD, 471
 tělo podmíněného příkazu, 153
 terminologie, 28
 arita, 78
 členy, 96
 entita programu, 194
 GoF, 237
 operace-operátor-operand, 78
 přepínač, 160
 rozměrnost polí, 385
 vektor, 385
 ternární operátor, 78
 test
 integrační, 472

jednotkový, 472
 vývoj řízený testy, 471
 Test Driven Development, 471
 textový blok: *viz* blok: textový
 textový řetězec: *viz* řetězec:
 textový
 this, 180
 konstruktor, 180
 throws, 359
 třída, 95
 abstraktní, 314
 anonymní, 392
 class-objekt, 96
 člen, 96, 174
 definice, 172
 hlavní třída aplikace, 434
 interní, 174
 knihovni
 Object, 295
 konečná, 308
 konstruktor třídy, 268
 lokální, 391
 Optional, 343
 řídící, 176
 samostatně definovaná, 174
 StringBuffer, 338
 StringBuilder, 338
 System, 125
 uspořádání prvků v těle, 197
 viditelnost, 174
 závislá, 176
 třída vnitřní, 392
 typ
 boolean, 61
 byte, 62
 datový, 60
 interní, 391
 vnořený, 394
 dědění rozhraní, 293
 double, 61
 float, 62
 generický, 281
 globální, 391
 hodnotový, 331
 char, 62
 int, 61
 interní, 96
 long, 62
 neměnný, 333
 obalový, 335

Object, 62
 objektový, 61
 odkazový, 331
 Optional, 343
 primitivní, 61, 335
 proměnný, 333
 short, 62
 String, 62
 vnořený, 392
 void, 60
 výčtový, 239
 typové členy, 393
 typový parametr: *viz* parametr:
 typový

U

učebnice
 programování ve verších, 472
 úloha, 28
 10.1, 187
 13.1, 221
 14.1, 235
 15.1, 251
 15.2, 251
 16.1, 265
 7.1, 125
 7.2, 134
 8.1, 143
 8.2, 145
 8.3, 148
 9.1, 154
 9.2, 155
 9.3, 158
 9.4, 161
 9.5, 165
 unární operátor, 78
 úryvek, 48
 ID, 49
 identifikace, 49

V

vazba
 časná, 308
 pozdní, 308
 vektor, 385
 verše, 472
 vícerozměrné pole, 385
 virtuální stroj, 38, *viz* stroj:
 virtuální

- vlastnost, 196
 - VMT, 303
 - vnitřní třída, 392
 - vnořený datový typ, 394
 - vnořený typ, 392
 - void, 60
 - Výčtový typ, 239
 - Vydavatel-Předplatitel, 245
 - výjimka, 348
 - catch, 355
 - definice vlastní, 358
 - kontrolovaná, 354, 359
 - převod na
 - nekontrolovanou, 360
 - nedosažitelný kód, 353
 - nekontrolovaná, 354
 - přehled důležitých výjimek, 349
 - throws, 359
 - try, 355
 - vyhození, 350
 - zachycení, 355
 - výjimky
 - hierarchie, 353
 - výpis
 - ExceptionCatching
 - rekursion1(int), 355
 - ExceptionChecking
 - conversion(), 361
 - throwsChecked(), 360
 - throwsUnchecked(), 360
 - ExceptionThrowing
 - throwing(), 350
 - výraz, 50
 - logický, 135
 - podmíněný, 155
 - přepínací, 162
 - switch, 162
 - výstup
 - standardní, 125
 - vývoj řízený testy, 471
 - výzva, 47
 - vzor
 - návrhový: viz návrhový vzor
 - Fasáda, 564
 - vzor návrhový
 - Abstraktní továrna, 262
 - Adaptér, 256
 - Iterátor, 397
 - Jedináček, 239
 - Jednoduchá tovární metoda, 238
 - Knihovni třída, 238
 - Multiton, 240
 - Posluchač, 245
 - Pozorovatel, 245
 - Prázdný objekt, 241
 - Prostředník, 242
 - Prototyp, 241
 - Přepřavka, 257
 - Služebník, 240
 - Statická tovární metoda, 238
 - Stav, 317
 - Šablonová metoda, 260
 - Výčtový typ, 239
 - Vydavatel-Předplatitel, 245
- X**
- XOR, 136
- Y**
- YAGNI, 448
- Z**
- zakomentování části programu, 210
 - zapouzdření, 193
 - zásada
 - CRIDP, 448
 - KISS, 448
 - SoC, 449
 - SRP, 449
 - YAGNI, 448
 - zásobník
 - návratových adres, 128
 - závorky
 - kulaté, 84
 - ZNA: viz zásobník:
 - návratových adres
 - znak
 - bílý, 173
 - zpráva, 94

Část F

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	578
A.1 Problémy s klávesnicí	578
A.2 Příprava spuštění pod Windows	578
A.3 Definice substituovaných disků	581
Příloha B Význam cizích slov	583
Příloha C Česko-americký slovník	585

Příloha A

Příprava spuštění JShell pod Windows

A.1 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 překladačů.

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 „svíslítko“ (`|`), a to ani tak, že je do textu vložíte ze schránky.

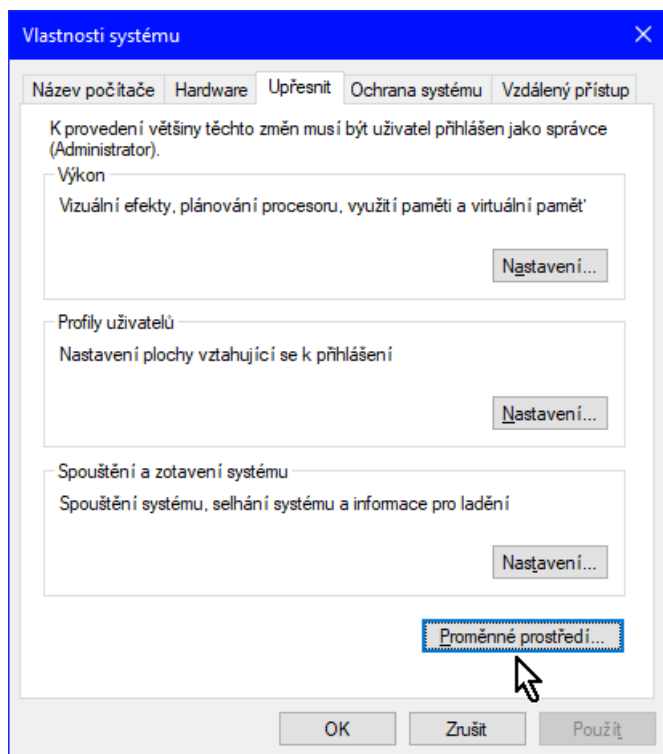
Vložit je můžete buďto přímým zadáním jejich kódu (`{=123, |=124, |=125`), anebo použitím jiného rozložení kláves.

A.2 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.

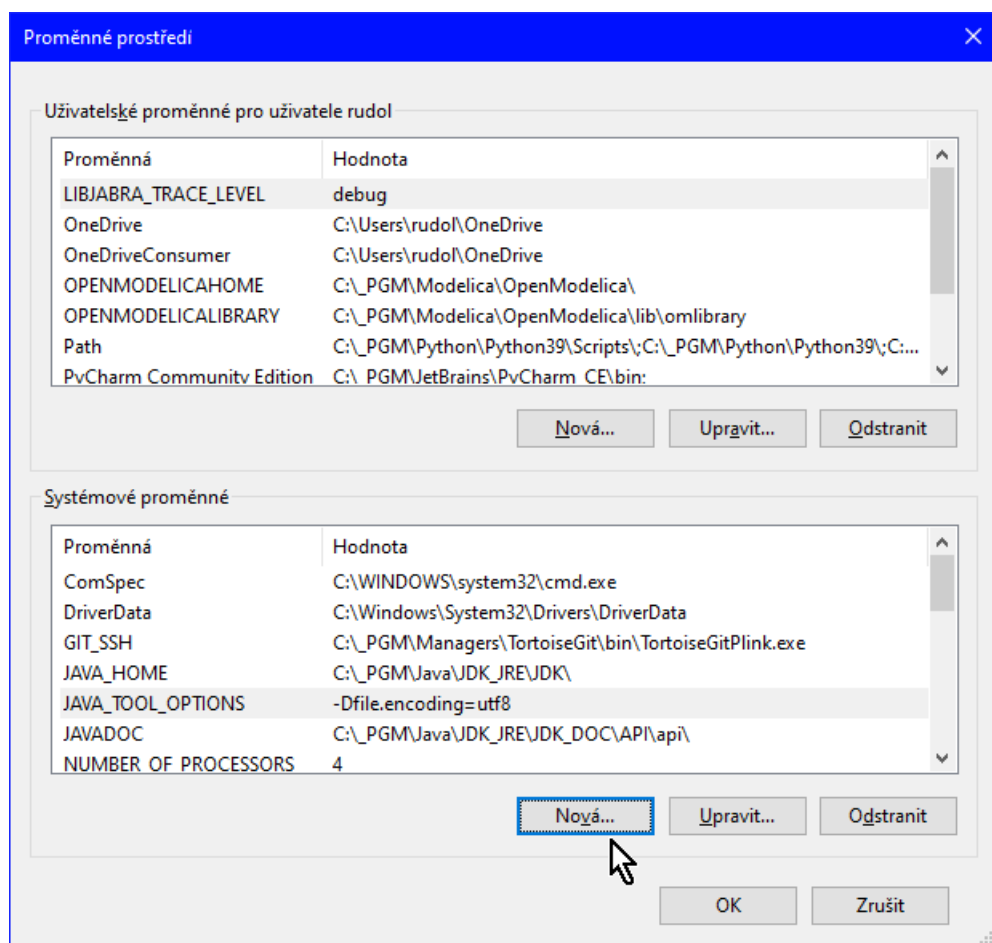
Budete-li chtít toto kódování používat ve všech svých programech v jazyku *Java*, bude nejlepší, když definujete proměnnou prostředí `JAVA_TOOL_OPTIONS`. Dosáhnete toho následovně:

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](#)).



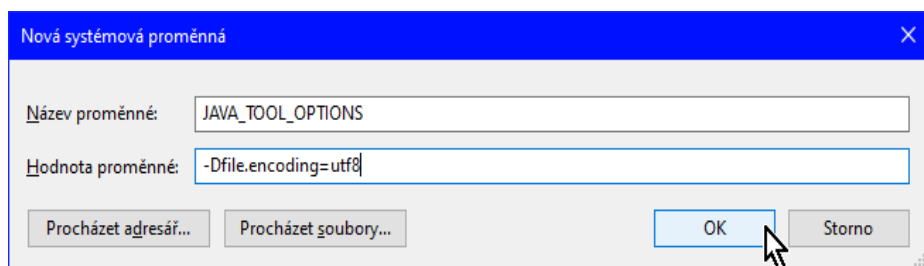
Obrázek A.1:
*Dialogové okno **Vlastnosti systému***

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 [46](#).

A.3 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 substituuovat 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 deklarují, oznamují, 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 [ouvəraid]
přepravka	crate [kreit]
rozhraní	interface [intəfeis]
seznam	list [list]
služebník	servant [sérvənt]
továrna, tovární	factory [fektəri]
třída	class [klás]
vlastnost	property [propəty]
zásobník	stack [stek]
zásobník návratových adres	return stack [ritəən stek]
závislost	dependence [dipendəns]
zpráva	message [mesidž]

Výslovnost některých slov v programu

boolean	[búlín]
break	[breik]
byte	[bait]
case	[keis]
class	[klás]
double	[dabl]
else	[els]
final	[fainl]
float	[flout]
for	[fór]
implements	[impliments]
instanceof	[instəns of]
interface	[intəfeis]
NetBeans	[netbíns]
new	[njú]
package	[pekidž]
private	[praivit]
public	[pablik]
return	[ritəən]
short	[šót]
static	[stetik]
super	[súpə]
switch	[swič]
while	[wail]

Část G

Seznamy

Seznam S1	Výpisy programů	589
Seznam S2	Seznam obrázků	595
Seznam S3	Seznam tabulek	597
Seznam S4	Seznam odboček – podšeděných bloků	598

Seznam S1

Výpisy programů

Výpis 2.1:	Přepsaný obsah konzolového okna otevřeného po spuštění dávky !_JShell.bat na mém počítači.....	47
Výpis 2.2:	První tři pokusné úryvky	48
Výpis 2.3:	Více úryvků na jednom řádku.....	51
Výpis 2.4:	Výpisy úryvků příkazem /list.....	53
Výpis 2.5:	Výpis všech úryvků příkazem /list -all.....	54
Výpis 2.6:	Různé způsoby uložení dosavadní práce příkazem /save	55
Výpis 2.7:	Vliv otevření skriptu PRINTING	56
Výpis 2.8:	Opětovné načtení předchozího stavu příkazem /reload -restore	57
Výpis 3.1:	Komentáře.....	64
Výpis 3.2:	Zadávání celočíselných hodnot.....	65
Výpis 3.3:	Zadávání reálných hodnot.....	66
Výpis 3.4:	Zadávání znaků.....	67
Výpis 3.5:	Literály typu String	69
Výpis 3.6:	Textové bloky	72
Výpis 3.7:	Logické hodnoty	72
Výpis 4.1:	Deklarace proměnných	77
Výpis 4.2:	Tři druhy dělení	81
Výpis 4.3:	Implicitní a explicitní přetypování	83
Výpis 4.4:	Použití složeného přiřazovacího příkazu	86
Výpis 4.5:	Inkrementační a dekrementační operátory	87
Výpis 4.6:	Porovnávací operátory	88
Výpis 5.1:	Práce s atributy	98
Výpis 5.2:	Volání metod třídy Math.....	99
Výpis 5.3:	Vytváření nových objektů	100
Výpis 6.1:	Import datového typu java.util.Date() a jeho následné použití.....	106
Výpis 6.2:	Zobrazení aktuálně importovaných tříd	107
Výpis 6.3:	Zpřístupnění knihovny robota Karla	109
Výpis 6.4:	Přidání prohledávané složky při spuštění JShell.....	110
Výpis 6.5:	Vytvoření světa, přidání robota a provedení příkazů	112
Výpis 6.6:	Ovládání vytvořeného robota	114
Výpis 6.7:	Demonstrace vlivu skrývání	115
Výpis 6.8:	Opětná aktivace příkazem /reload -restore.....	117
Výpis 7.1:	Definice metod kDoubleStep()	122
Výpis 7.2:	Definice a použití metod doubleStep(Karel) a turnAbout(Karel)	124

Výpis 7.3:	Definice a použití metody <code>turnWithInfo(Karel, String)</code>	126
Výpis 7.4:	Demonstrace použití lokálních konstant	127
Výpis 7.5:	Definice a použití přetížené metody <code>turnWithInfo(String, Karel)</code>	130
Výpis 7.6:	Použití přetížených verzí konstruktorů třídy <code>Karel</code>	132
Výpis 7.7:	Definice a použití metody <code>wallBehind(Karel)</code> vracející hodnotu	134
Výpis 7.8:	Přehled doposud definovaných metod	135
Výpis 7.9:	Definice a použití metod <code>forwardClosed(Karel)</code> , <code>forwardFree(Karel)</code> a <code>rightFree(Karel)</code> používajících logické výrazy a vracejících hodnotu	136
Výpis 7.10:	Demonstrace předávání argumentu odkazem v jazyku <code>C#</code>	138
Výpis 7.11:	Obdobná metoda v jazyku <code>Java</code>	139
Výpis 8.1:	Použití cyklu <code>while</code> na příkladu metod <code>goToWall(Karel)</code> a <code>turnToNorth(Karel)</code>	142
Výpis 8.2:	Použití cyklu <code>do...while</code> na příkladu metod <code>goToEmpty(Karel)</code> a <code>markersToWall(Karel)</code>	144
Výpis 8.3:	Použití cyklu <code>for</code> na příkladu metod <code>putNMarkers(Karel, int)</code> , <code>putIncreasingMarkers(Karel)</code> a <code>markers(Karel)</code>	147
Výpis 8.4:	Použití nekonečného cyklu na příkladu metody <code>around(Karel)</code>	149
Výpis 8.5:	Definice metody <code>clearToWall(Karel)</code>	150
Výpis 9.1:	Použití jednoduchého podmíněného příkazu	153
Výpis 9.2:	Použití úplného podmíněného příkazu	155
Výpis 9.3:	Použití podmíněného výrazu	156
Výpis 9.4:	Zanořené zarovnaný složený podmíněný příkaz	157
Výpis 9.5:	Doporučené formátování při výběru z více možností	157
Výpis 9.6:	Využití složeného podmíněného příkazu při průchodu bludištěm	157
Výpis 9.7:	Koncepce přepínače (příkazu <code>switch</code>) převzatá z jazyka <code>C</code>	159
Výpis 9.8:	Novější koncepce přepínače (příkazu <code>switch</code>)	160
Výpis 9.9:	Použití klasické podoby přepínače	162
Výpis 9.10:	Použití přepínacího výrazu	163
Výpis 9.11:	Definice metody <code>turnTo(Karel, Direction)</code> s využitím klasické verze přepínače	164
Výpis 9.12:	Definice metody <code>markersToWall(Karel)</code> používající cyklus s podmínkou uprostřed	165
Výpis 9.13:	Definice funkce <code>sum3xC()</code> demonstrující použití příkazu <code>continue</code>	166
Výpis 9.14:	Definice funkce <code>factorial()</code> demonstrující použití rekurze	167
Výpis 9.15:	Definice funkce <code>put_markers_at_wall()</code> řešené pomocí rekurze	168
Výpis 10.1:	Nejjednodušší definice třídy a vytvoření její instance	173
Výpis 10.2:	Definice a použití třídy <code>Light</code> se třemi konstruktory	178
Výpis 10.3:	Upravená definice třídy <code>Light</code> v souboru <code>sl0a_Light.jsh</code>	186
Výpis 10.4:	Prověření funkce třídy <code>Light</code> ze souboru <code>sl0a_Light.jsh</code>	187
Výpis 11.1:	Ekvivalent třídy <code>Light</code> , která je řešením úlohy 10.1 na straně 187, definovaný jako řádná, samostatně definovaná veřejná třída v balíčku <code>b77_java_nz2._11_arrangement</code>	191
Výpis 11.2:	Prověření funkce třídy <code>Light</code> z výpisu 11.1	192
Výpis 11.3:	Definice třídy <code>ClassTemplate</code> v balíčku <code>b77_java_nz2._11_arrangement</code> představující šablonu definic třídy	200
Výpis 12.1:	Obsah souboru <code>package-info.java</code> v balíčku <code>b77_java_nz2._12_documentation</code>	206

Výpis 12.2:	Začátek definice třídy <code>Light</code> s doplněnými dokumentačními komentáři, ale bez komentářů oddělujících jednotlivé sekce	207
Výpis 13.1:	Možná definice metody <code>moveBy(int, int, IMovable)</code>	216
Výpis 13.2:	Definice interfejsu <code>IMovable</code>	217
Výpis 13.3:	Výňatek z definice třídy <code>Light</code> implementující interfejs <code>IMoveable</code>	219
Výpis 13.4:	Test přesouvateľnosti instance třídy <code>Light</code>	220
Výpis 14.1:	Prověření plynulých změn pozice a velikosti instance třídy <code>Light</code>	224
Výpis 14.2:	Definice interfejsu <code>IChangeable</code> v balíčku <code>shapes77.canvas_4</code>	229
Výpis 14.3:	Definice interfejsu <code>IVehicle1_1</code> v balíčku <code>b77_java_nz2.vehicle</code>	231
Výpis 14.4:	Definice třídy <code>Robot1_1</code> implementující interfejs <code>IVehicle1_1</code> (po odebrání komentářů)	232
Výpis 14.5:	Test funkcionality třídy <code>Robot1_1</code>	234
Výpis 15.1:	Definice skriptu <code>importlib.jsh</code>	248
Výpis 15.2:	Definice interfejsu <code>IVehicle1_2</code> (po odebrání komentářů)	249
Výpis 15.3:	Provedené změny ve třídě <code>Robot1_2</code> (po odebrání komentářů)	249
Výpis 15.4:	Test upravené třídy <code>Robot1_2</code>	250
Výpis 16.1:	Ověření funkce příkazu <code>import static</code>	253
Výpis 16.2:	Definice interfejsu <code>shapes77.canvasmanager.ICMPaintable</code>	254
Výpis 16.3:	Definice záznamu a práce s ním	258
Výpis 16.4:	Definice třídy (záznamu) <code>Area</code>	259
Výpis 16.5:	Použití přepravek (záznamů) typu <code>Position</code> , <code>Size</code> a <code>Area</code>	260
Výpis 16.6:	Definice abstraktní továrny – interfejsu <code>IVehicleFactory1_3</code>	264
Výpis 16.7:	Definice instančních konstant a použití inicializačních bloků	266
Výpis 16.8:	Demonstrace vlivu použití inicializačních bloků	267
Výpis 16.9:	Definice statických konstant a inicializačních bloků	269
Výpis 16.10:	Aktivace třídy z výpisu 16.9	270
Výpis 16.11:	Třída <code>SwitchConst</code> používající přepínač	271
Výpis 16.12:	Spuštění třídy <code>SwitchConst</code> používající přepínač v podobě z výpisu 16.11	272
Výpis 16.13:	Třída <code>StaticArray</code> definující pole pevné velikosti jako statické atributy	272
Výpis 17.1:	Výpis přidaných částí kódu ve třídě <code>Light</code> v balíčku <code>b77_java_nz2._17_lambda</code>	277
Výpis 17.2:	Prověrka nezávislosti blikání instancí třídy <code>Light</code>	278
Výpis 17.3:	Lambda-výraz definovaný prostřednictvím operátoru <code>::</code>	280
Výpis 17.4:	Definice třídy <code>Interval</code>	283
Výpis 17.5:	Test funkcionality třídy <code>Interval</code>	283
Výpis 17.6:	Názvy a abstraktní metody vybraných funkčních interfejsů	287
Výpis 17.7:	Definice a použití lambda-výrazů za použití operátoru <code>::</code>	288
Výpis 17.8:	Přetypování lambda-výrazů	289
Výpis 18.1:	Definice prázdné třídy a přehled jejích členů	296
Výpis 18.2:	Počáteční primitivní definice třídy <code>Square</code>	298
Výpis 18.3:	Definice třídy <code>Square</code> s doplněným konstruktorem	300
Výpis 18.4:	Demonstrace přebíjení metod	302
Výpis 18.5:	Definice tříd <code>Matka</code> , <code>Dcera</code> , <code>Vnučka</code>	304
Výpis 18.6:	Definice metody <code>setSize(int, int)</code> pro třídu <code>Square</code>	305
Výpis 18.7:	Počátek chybové zprávy po žádosti o nastavení velikosti	306
Výpis 18.8:	Definice metody <code>setSize(int)</code> v interfejsu <code>IResizable</code>	306
Výpis 18.9:	Definice metody <code>setSize(int, int)</code> pro třídu <code>Square</code>	307

Výpis 18.10:	Demonstrace zakrývání statických metod.....	309
Výpis 19.1:	Experimenty s abstraktní třídou	315
Výpis 19.2:	Definice (hlavní) třídy Robot4_4 definující otočná vozidla s odebranými komentáři, přetíženými konstruktory a příkazy package a import	319
Výpis 19.3:	Definice abstraktního rodiče jednosměrných podobjektů – třídy ARobot1_4.....	323
Výpis 19.4:	Definice třídy Robot1E_4 definující jednosměrné podobjektoty otočené na východ.....	325
Výpis 19.5:	Příkazy a úryvky pro test otočných vozidel	327
Výpis 20.1:	Demonstrace chování instancí neměnného typu na příkladu stringů	334
Výpis 20.2:	Definice nešikovné verze metody reverseInWords(int)	338
Výpis 20.3:	Definice upravené verze metody reverseInWords(long).....	339
Výpis 20.4:	Definice jednoduchého výčtového typu a jeho použití	340
Výpis 20.5:	Definice výčtového typu Direction robota Karla bez komentářů	341
Výpis 20.6:	Interval využívající komparátor	347
Výpis 21.1:	Ukázka vyhození výjimky a reakce systému na něj	350
Výpis 21.2:	Začátek definice třídy ExceptionThrowing	352
Výpis 21.3:	Reakce systému na vyhození výjimky ve třídě se zdrojovým kódem	352
Výpis 21.4:	Definice metody unreachable() demonstrující nedosažitelnost kódu za vyhozením výjimky	353
Výpis 21.5:	Metoda try_catch(int) demonstrující zachycení vyhozené výjimky.....	355
Výpis 21.6:	Demonstrace funkce bloku try...catch...finally	357
Výpis 21.7:	Definice vlastní kontrolované a nekontrolované výjimky.....	358
Výpis 21.8:	Demonstrace odezvy překladače při kontrole zabezpečení reakce na vyhození výjimky	360
Výpis 21.9:	Definice metody conversion() převádějící kontrolovanou výjimku na nekontrolovanou.....	361
Výpis 22.1:	Práce s množinou	369
Výpis 22.2:	Seznamy a práce s nimi	372
Výpis 22.3:	Dva způsoby seřazení prvků v seznamu	374
Výpis 22.4:	Základní operace s mapami.....	375
Výpis 23.1:	Demonstrace základních vlastností polí objektů	379
Výpis 23.2:	Základní vlastnosti polí hodnot primitivních typů.....	380
Výpis 23.3:	Inicializace pole a jeho tisk.....	382
Výpis 23.4:	Další způsoby inicializace	383
Výpis 23.5:	Dvojezměrné pole	386
Výpis 23.6:	Definice a použití metody average(double...) využívající proměnný počet argumentů.....	387
Výpis 23.7:	Převod čísla do tisíce na vyjádření slovy v češtině	389
Výpis 24.1:	Srovnání použití lambda-výrazu a anonymní třídy	397
Výpis 24.2:	Použití metody forEach(Consumer<T>)	399
Výpis 24.3:	Omezení dvojtečkového cyklu for(:)	401
Výpis 24.4:	Třída RandomStrings – generátor pseudonáhodných textů	402
Výpis 24.5:	Test použitelnosti instancí RandomStrings v cyklu	403
Výpis 24.6:	Test použitelnosti instancí RandomStrings s interním iterátorem forEach().....	404
Výpis 25.1:	Ukázka použití datovodů při generování čísel sportky.....	413
Výpis 26.1:	Několik příkladů práce s instancemi třídy File	421
Výpis 26.2:	Demonstrace použití výstupních datových proudů	428
Výpis 26.3:	Demonstrace použití vstupních datových proudů	431
Výpis 26.4:	Demonstrace použití třídy Scanner	432
Výpis 27.1:	Definice třídy GuessC představující jednoduchou aplikaci.....	435

Výpis 27.2:	Překlad a spuštění aplikace definované v jednom zdrojovém souboru	437
Výpis 27.3:	Definice třídy Main spouštějící demonstrační aplikaci	439
Výpis 27.4:	Definice třídy GuessIO umožňující vybrat způsob komunikace	440
Výpis 27.5:	Obsah souboru MANIFEST.MF ve složce 77_LIB.META-INF	442
Výpis 30.1:	Definice alternativního konstruktoru třídy ScenarioStep	476
Výpis 31.1:	Třída Portal v balíčku game77.ck1a_happy	483
Výpis 31.2:	Etapy vývoje aplikace a jejich identifikátory	484
Výpis 31.3:	Definice třídy ScenarioManager definující (mimo jiné) šťastný scénář	485
Výpis 31.4:	Zpráva obdržená po spuštění třídy game77.ck1a_happy.Portal	489
Výpis 31.5:	Definice počátku třídy game77.ck1b_duplet.ScenarioManager	491
Výpis 31.6:	Výsek zprávy o testu třídy game77.ck1b_duplet.Portal	492
Výpis 32.1:	Konstruktor instancí třídy Action	496
Výpis 32.2:	Začátek definice třídy AItemContainer s výchozí podobou konstrukturu a metody items()	498
Výpis 32.3:	Prozatímní definice třídy Bag	499
Výpis 32.4:	Definice konstrukturu instancí třídy Place	499
Výpis 32.5:	Prozatímní definice třídy World	500
Výpis 32.6:	Prozatímní definice třídy game77.ck1c_architecture.Game	502
Výpis 32.7:	Konec zprávy o testu třídy game77.ck1c_architecture.Portal	503
Výpis 33.1:	Definice upravených metod ve třídě Game	506
Výpis 33.2:	Definice statických metod isActive() a stop() ve třídě Action	507
Výpis 33.3:	Definice statické metody executeCommand(String) ve třídě Action	508
Výpis 33.4:	Definice statických metod executeEmptyCommand(), executeStandardCommand(String) a initialize() ve třídě Action	509
Výpis 33.5:	Úprava počátku definice třídy ScenarioManager v balíčku ck1d_start	511
Výpis 33.6:	Výsek zprávy po spuštění třídy ck1d_start.Portal	513
Výpis 34.1:	Upravený počátek definice třídy Place v balíčku game77.ck1e_world	515
Výpis 34.2:	Návrh definice metody pro inicializaci prostoru	516
Výpis 34.3:	Upravená verze třídy AItemContainer v balíčku game77.ck1e_world	516
Výpis 34.4:	Změny ve třídě ScenarioManager pro zpracování dalších konstant	518
Výpis 34.5:	Upravené metody ve třídě World v balíčku game77.ck1e_world	520
Výpis 34.6:	Upravená definice metody inicitalize() ve třídě game77.ck1e_world.Action	521
Výpis 34.7:	Výsek ze zprávy ze závěrečného testu na hladině Level.WORLD	521
Výpis 35.1:	Definice seznamu ACTIONS potřebných akcí	524
Výpis 35.2:	Nová definice funkce executeStandardCommand(String) v modulu actions	525
Výpis 35.3:	Definice metody move(String[]) realizující přesun do sousedního prostoru	526
Výpis 35.4:	Výsek klíčové části zprávy o provedení testu	526
Výpis 35.5:	Definice metody take(String[]) realizující přesun h-objektu z prostoru do batohu	527
Výpis 35.6:	Definice metody put(String[]) realizující přesun h-objektu z prostoru do batohu	527
Výpis 35.7:	Definice metody help(String[]) realizující nápovědu	528
Výpis 35.8:	Závěr zprávy o testu na hladině Level.BASIC	528
Výpis 36.1:	Typy kroků, které je třeba zařadit do chybového scénáře	532
Výpis 36.2:	Začátek definice chybového scénáře	533
Výpis 36.3:	Závěr zprávy o testu na hladině TRIPLET	534

Výpis 36.4:	Upravená definice statické metody executeCommand(String) ve třídě Action	535
Výpis 36.5:	Upravená definice statické metody executeStandardCommand(String)	535
Výpis 36.6:	Upravená definice statické metody move(String[])	536
Výpis 36.7:	Upravená definice třídy Item	538
Výpis 36.8:	Upravená definice konstruktoru instancí třídy World	538
Výpis 36.9:	Upravené definice metod addItem(Item), removeItem(Item) a initialize() ve třídě Bag	540
Výpis 36.10:	Upravené definice metody take(String[]) ve třídě Action	540
Výpis 36.11:	Upravené definice metody put(String[]) ve třídě Action	541
Výpis 37.1:	Kroky scénáře HAPPY s nestandardními příkazy a definovanými konstantami	544
Výpis 37.2:	Finální definice seznamu ACTIONS všech instancí třídy Action	544
Výpis 37.3:	Výchozí podoba definic nestandardních akcí	545
Výpis 37.4:	Upravený startovní krok všech scénářů ve třídě SceneManager	548
Výpis 37.5:	Upravené nestandardní kroky ve scénáři HAPPY	549
Výpis 37.6:	Typy kroků se špatně zadanými příkazy, jež je třeba prověřit	549
Výpis 37.7:	Počátek scénáře MISTAKES_NS	551
Výpis 37.8:	Definice metod conditions() a tests() ve třídě Game	552
Výpis 37.9:	Definice atributů CONDITIONS a TESTS ve třídě Action	553
Výpis 37.10:	Upravená metoda initialize() ve třídě Action	553
Výpis 37.11:	Upravená definice metody wakeUp(String[]) ve třídě Action	554
Výpis 38.1:	Výchozí definice třídy Main v balíčku game77.ck1l_user_io	556
Výpis 38.2:	Definice metody multirun(IGame)	557
Výpis 38.3:	Definice funkce currentState(IGame)	558
Výpis 38.4:	Definice rozhraní IUI	560
Výpis 38.5:	Upravená definice metody main(String[]) ve třídě Main2	561
Výpis 38.6:	Upravené definice metod run(...) a multirun(...) ve třídě Main2	561
Výpis 38.7:	Definice třídy Console v modulu interfaces	562
Výpis 38.8:	Definice třídy PrimitiveGUI	565

Seznam S2

Seznam obrázků

Obrázek 1.1: Průběh popularity nejpopulárnějších IDE (zdroj: https://www.jrebel.com/blog/best-java-ide)	43
Obrázek 2.1: Konzolové okno otevřené po spuštění dávky <code>!_JShell.bat</code> na mém počítači	46
Obrázek 5.1: Vytvořený svět spolu s robotem.....	100
Obrázek 6.1: Stromová struktura balíčků	104
Obrázek 6.2: Svět Robota po provedení akcí z výpisu 6.5	113
Obrázek 6.3: Svět Robota po provedení akcí z výpisu 6.6	113
Obrázek 7.1: Výsledná podoba dvorku se čtyřmi roboty.....	133
Obrázek 8.1: Vývojový diagram cyklu s počáteční podmínkou (cyklu <code>while</code>).....	141
Obrázek 8.2: Syntaktický diagram cyklu s počáteční podmínkou (cyklu <code>while</code>)	142
Obrázek 8.3: Vývojový diagram cyklu s ukončovací podmínkou (cyklu <code>do...while</code>).....	143
Obrázek 8.4: Syntaktický diagram cyklu s ukončovací podmínkou (cyklu <code>do...while</code>)	144
Obrázek 8.5: Vývojový diagram cyklu s parametrem (cyklu <code>for</code>)	145
Obrázek 8.6: Syntaktický diagram cyklu s parametrem (cyklu <code>for</code>)	146
Obrázek 8.7: Vývojový diagram cyklu s podmínkou uprostřed	150
Obrázek 9.1: Vývojový diagram podmíněného příkazu.....	153
Obrázek 9.2: Vývojový diagram úplného podmíněného příkazu.....	154
Obrázek 9.3: Syntaktický diagram obecného podmíněného příkazu	154
Obrázek 9.4: Vývojový diagram složeného podmíněného příkazu	156
Obrázek 9.5: Bludiště s robotem	156
Obrázek 9.6: Vývojový diagram přepínače – příkazu <code>switch</code>	159
Obrázek 9.7: Syntaktický diagram přepínače – příkazu <code>switch</code>	159
Obrázek 10.1: Diagram tříd balíčku <code>shapes77.canvas_1</code>	176
Obrázek 10.2: Okno plátna se třemi vytvořenými světlý	177
Obrázek 13.1: Diagram tříd balíčku <code>shapes77.canvas_2</code>	215
Obrázek 14.1: Diagram tříd balíčku <code>shapes77.canvas_3</code>	223
Obrázek 14.2: Diagram tříd balíčku <code>shapes77.canvas_4</code>	227
Obrázek 14.3: Diagram tříd balíčku <code>shapes77.canvas_4</code> po odstranění šipek závislostí a přesunu ikon tříd a interfejsů do přehlednějšího uspořádání	228
Obrázek 15.1: Zmenšení počtu závislostí po aplikaci vzoru Prostředník	243
Obrázek 15.2: Podoba diagramu tříd po zavedení abstrakce, které se musejí všechny komunikující objekty přizpůsobit	245
Obrázek 15.3: Diagram tříd balíčku <code>shapes77.canvasmanager</code>	248
Obrázek 16.1: Diagramy tříd možných implementací vzoru Adaptér	256

Obrázek 17.1	285
Obrázek 18.1 Rodičovský podobjekt.....	298
Obrázek 18.2: Schematická demonstrace vnitřního uspořádání instance třídy Vnučka.....	304
Obrázek 18.3: Malý popis.....	305
Obrázek 19.1: Diagram tříd balíčku shapes77. geom.....	317
Obrázek 19.2: Principiální diagram uspořádání tříd podle návrhového vzoru Stav	318
Obrázek 19.3: Diagram tříd balíčku b77_java_nz2._19_abstract zobrazující třídu Robot4_4 a její pomocné třídy.....	326
Obrázek 21.1: Hierarchie dědění výjimek	354
Obrázek 22.1: Hlavní datové typy patřící do knihovny kolekcí.....	364
Obrázek 22.2: Syntaktický diagram „dvojtečkového“ cyklu for (cyklu „for each“).....	368
Obrázek 24.1: Rozdělení interních datových typů.....	392
Obrázek 25.1: Schéma postupu zpracování dat předávaných datovody	410
Obrázek 26.1: Exponenciální nárůst počtu tříd při přidávání funkcí.....	423
Obrázek 26.2: Třídy po aplikaci dekorátorů	424
Obrázek 29.1: Předběžný diagram tříd balíčku game77. api	467
Obrázek 30.1: Předběžný diagram tříd balíčku game77. api	480
Obrázek 38.1: Dialogové okno s výzvou k zadání údajů.....	566
Obrázek 38.2: Diagram tříd aktuálního stavu balíčku game77. ck1l_user_io	567

Seznam S3

Seznam tabulek

Tabulka 4.1: Přehled operátorů, jejich priorit a asociativity	79
Tabulka 4.2: Inkrementační a dekrementační operátory	87
Tabulka 20.1: Primitivní a obalové typy	335
Tabulka 26.1: Rodičovské třídy jednotlivých typů datových proudů	425
Tabulka 28.1: Porovnání metody shora dolů a zdola nahoru	453

Seznam S4

Seznam odboček – podšeděných bloků

Odbočka – podšeděný blok	29
Datová struktura strom	105
Historie robota Karla	111
Tisk na standardní výstup	126
Zásobník návratových adres – ZNA	128
Bílé znaky a uspořádání programu	173
Problémy s kódováním znaků	337
Prohlížení obsahu JAR-souborů	441
Co to je h-objekt	458
Návrhový vzor Fasáda	564



Nové vydání knihy o programování v jazyce Java si klade ambiciózní cíl: naučit každého, včetně začátečníků, skutečně programovat moderním, objektově orientovaným stylem, jímž se v dnešní době vyvíjí drtivá většina klíčových aplikací. Nespokojí se jen s tím, že naučí čtenáře napsat nějaký program. Mnohem důležitější je zvládnout správný návrh a pochopit základní principy, na nichž je objektově orientované programování postaveno. Nedržíte tedy v ruce jen příručku jazyka Java, ale učebnici objektově orientovaného programování v Javě. Je možné, že práci „pouhých“ kodérů v Javě bude časem vykonávat umělá inteligence, ale schopný architekt a návrhář aplikací je zatím stále nenahraditelný. A právě tím se můžete stát s pomocí této knihy.

Knihu lektorovali vedoucí různých programátorských týmů, aby posoudili, nakolik odráží potřeby a zvyky jejich každodenní praxe. Někteří z nich se rozhodli, že ji ve svých týmech budou používat jako povinnou literaturu pro nováčky. Lepší doporučení snad ani nemohla dostat.



Grada Publishing, a.s.,
U Průhonu 22, 170 00 Praha 7
tel.: +420 234 264 401
e-mail: obchod@grada.cz, www.grada.cz

CZ 549 Kč / SK 22,95 €

