

knihovna programátora

- Podrobný výklad vlastností jazyka od naprostých základů až po pokročilé, běžně neprobírané konstrukce
- Probírá i konstrukce, které budou zařazeny až do příštích verzí jazyka, a předvádí, jak tyto konstrukce vyzkoušet
- Vysvětluje nejenom jak probírané konstrukce používat, ale také proč jsou právě takové
- Využívá zabudované REPL prostředí pro demonstraci vykládaných konstrukcí bez zbytečného pomocného kódu
- Může sloužit současně jako učebnice i referenční příručka



RUDOLF PECINOVSKÝ

Java 14

Kompletní příručka jazyka

Ing. Rudolf Pecinovský, CSc. je absolventem *Fakulty Elektrotechnické ČVUT* z roku 1979. Titul CSc. získal v *Ústavu teorie informace a automatizace ČSAV* v roce 1983. Od počátku 80. let učí a publikuje, přičemž svůj výzkum soustředí především na oblast vstupních kurzů moderního programování pro naprosté začátečníky. V současné době učí na *Vysoké škole ekonomické v Praze*, na *Fakultě jaderné a fyzikálně inženýrské ČVUT* a na *Vysoké škole podnikání a práva*. Doposud mu vyšlo přes 60 knih, které byly přeloženy do pěti jazyků. Většina jeho knih je zaměřena na výuku moderního programování a na umění návrhu objektově orientované architektury.



knihovna programátora

RUDOLF PECINOVSKÝ

Java 14

Kompletní příručka jazyka

GRADA
Publishing

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**.

Rudolf Pecinovský

Java 14

Kompletní příručka jazyka

Vydala Grada Publishing, a.s.

U Průhonu 22, Praha 7

obchod@grada.cz, www.grada.cz

tel.: +420 234 264 401

jako svou 7620. publikaci

Odpovědný redaktor: Ivana Palasová

Návrh vnitřního layoutu Rudolf Pecinovský

Zlom Rudolf Pecinovský

Počet stran 578

První vydání, Praha 2020

Vytiskly Tiskárny Havlíčkův Brod, a.s.

© Grada Publishing, a.s., 2020

Cover Design © Grada Publishing, a. s., 2020

Cover Photo © Depositphotos

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

ISBN 978-80-271-1605-8 (ePub)

ISBN 978-80-271-1604-1 (pdf)

ISBN 978-80-271-1369-9 (print)

Všem, kteří se chtějí něco naučit

Stručný obsah

Stručný obsah	6
Podrobný obsah	8
Úvod	23
Část I Neobjektové konstrukce	31
1 Prostředí JShell	32
2 Základní datové typy a jejich literály	51
3 Proměnné	72
4 Základní operátory	85
5 Definice metod	104
6 Ostatní operátory	121
7 Pole	148
8 Rozhodování	164
9 Opakování části kódu	180
Část II Základní objektové konstrukce	201
10 Základy objektově orientovaného paradigmatu	202
11 Třídy a jejich členy	221
12 Vývojová prostředí a vytvoření aplikace	243
13 Balíčky a knihovny	262
14 Dokumentace API	283
15 Konstrukce <code>interface</code>	296
16 Podrobnosti o konstruktorech	314
17 Úvod do dědění implementace	331
18 Viditelnost členů tříd	348
19 Virtuální metody a jejich přebíjení	364
20 Abstraktní třídy	375

Část III Pokročilé objektové konstrukce	387
21 Výjimky a aserce	388
22 Generické datové typy a metody	416
23 Typové parametry a argumenty	436
24 Interní datové typy	457
25 Výčtové typy – třídy typu enum	472
26 Záznamy – třídy typu record.....	486
27 Lambda-výrazy	500
28 Anotace	510
29 Vlákna a paralelní procesy	523
30 Moduly.....	531
Část IV Přílohy	555
A Omezení JVM	556
B Tvorba jednoduchého GUI.....	557
Literatura.....	571
Rejstřík.....	573

Podrobný obsah

Stručný obsah	6
Podrobný obsah	8
Úvod	23
Komu je kniha určena	23
Koncepce výkladu	24
Rozdělení textu	24
Terminologie	25
Použité nástroje	25
Vývojová sada JDK	25
Vývojové prostředí JShell	25
Samostatné vývojové prostředí	26
Doprovodné programy	26
Předběžné definice nových konstrukcí	26
Předběžná funkce/konstrukce/vlastnost (Preview Feature)	27
Experimentální funkce/konstrukce/vlastnost (Experimental Feature)	27
Inkubační funkce/konstrukce/vlastnost (Incubating Feature, Incubator)	27
Syntaktické definice a diagramy	27
Použité typografické konvence	28
Odbočka – podšeděný blok	30
Zpětná vazba	30
Část I Neobjektové konstrukce	31
1 Prostředí JShell	32
1.1 Nejprve trochu terminologie	32
Objekt	32
Třída, datový typ	33
Proměnná	33
Atributy × metody	33
1.2 Charakteristika programu a prostředí JShell	33
1.3 Problémy s klávesnicí	33
1.4 Příprava programu JShell a první spuštění	34
Dávkové soubory pro Windows	34
Po spuštění	36
1.5 Úryvky (snippets)	37
Použití proměnných	38
Identifikace úryvků	38
Středník	38
Více objektů na řádku, zavlečené chyby	38
1.6 Příkazy (commands)	40
Vyloučení úryvku: /drop	40
Přehled aktivních úryvků: /list	40

Přehled všech úryvků: <code>/list -all</code>	41
Přehled objektů daného druhu.....	42
Uložení aktivních úryvků: <code>/save <file></code>	43
Uložení všech zadaných úryvků: <code>/save -all <file></code>	43
Uložení dosavadního průběhu seance: <code>/save -history <file></code>	43
Načtení skriptu: <code>/open <file></code>	43
Ukončení seance: <code>/exit</code>	43
Restart: <code>/reset</code>	44
Znovuzavedení: <code>/reload -restore</code>	44
Nastavení startovního skriptu: <code>/set -start <file></code>	44
Nápověda: <code>/?</code>	44
1.7 Základní syntaktická pravidla.....	45
Bílé znaky.....	45
Komentáře.....	45
1.8 Ovládání.....	47
Použití editoru.....	47
Nastavení vlastního editoru.....	49
1.9 Doprovodné programy.....	50
1.10 Shrnutí.....	50
2 Základní datové typy a jejich literály.....	51
2.1 Datové typy.....	51
Dělení datových typů.....	52
Primitivní datové typy.....	53
Objektové datové typy.....	54
Odkazy na objekty.....	54
2.2 Literály.....	55
Literály typu <code>boolean</code>	55
Literály typu <code>int</code>	55
Historická vsuvka – číselné soustavy.....	56
Názvy skupin bitů.....	57
Literály typu <code>long</code>	57
Literály typu <code>byte</code> a <code>short</code>	58
Literály typu <code>double</code>	58
Celé číslo s příponou.....	59
Obyčejné desetinné číslo.....	59
Číslo v exponentovém tvaru.....	59
Literály typu <code>float</code>	61
Literály typu <code>char</code>	61
Prázdný odkaz <code>null</code>	65
Literály typu <code>String</code>	65
Textové bloky.....	67
Literály typu <code>Class</code>	69
2.3 Ještě trocha terminologie.....	71
2.4 Nestandardní hodnoty reálných typů.....	71
2.5 Shrnutí.....	71
3 Proměnné.....	72
3.1 Pravidla pro tvorbu identifikátorů.....	72
Používání znaku <code>\$</code>	73
Konvence pro velikost písmen.....	73
3.2 Druhy typování.....	74
Statické × dynamické typování.....	74
Definice × odvození datového typu.....	75
Silné (přísné) × slabé typování.....	75
Shrnutí.....	76

3.3	Definice × deklarace.....	76
3.4	Deklarace proměnných.....	76
3.5	Středníky.....	79
3.6	Současná deklarace více proměnných.....	79
	Reakce prostředí <i>JShell</i>	79
3.7	Redeklarace proměnných v <i>JShell</i>	81
3.8	Deklarace s přiřazením počáteční hodnoty.....	82
	Pozor na velikost znaků.....	83
	Zpět k deklaraci s přiřazením počáteční hodnoty.....	83
3.9	Syntaktický diagram.....	83
3.10	Shrnutí.....	84
4	Základní operátory.....	85
4.0	Inicializace prostředí <i>JShell</i>	85
4.1	Nejprve trocha teorie.....	86
4.2	Operátor přiřazení =.....	87
	Přiřazení je výraz.....	87
4.3	Unární + a -.....	88
4.4	Aritmetické operátory + - * / %.....	88
	Operátor sčítání.....	88
	Sčítání stringů.....	88
	Operátor odčítání.....	89
	Operátor násobení.....	90
	Operátor dělení.....	90
	Operátor zbytku po dělení.....	90
4.5	Kulaté závorky ().....	91
	Alternativní řešení.....	92
4.6	Operátor přetypování (typ).....	92
	Implicitní přetypování.....	92
	Příklady implicitního přetypování.....	93
	Explicitní přetypování.....	94
	Priorita.....	95
	Kontrola.....	95
	Explicitní přetypování hodnot primitivních typů.....	96
	Příklady.....	96
	Přetypování instancí objektových datových typů.....	98
	Univerzální „přetypování“ na <i>String</i>	99
	Textový podpis.....	100
4.7	Specifika číselných typů.....	100
	Malé celočíselné typy.....	100
	Ztráta přesnosti.....	102
	Pořadí vyhodnocování.....	102
	První příklad.....	102
	Druhý příklad.....	103
4.8	Shrnutí.....	103
5	Definice metod.....	104
5.1	Historické ohlédnutí.....	104
5.2	Definice a volání metody.....	105
5.3	Volání metody.....	107
5.4	Metody s parametry.....	108
	Parametry versus argumenty.....	109
	Více parametrů.....	109
	Předávání hodnot parametrů.....	110
5.5	Metody vracějící hodnotu.....	111
5.6	Přetěžování metod.....	112

5.7 Lokální proměnné metod.....	113
Postup volání metody	114
Zásobník návratových adres – ZNA.....	115
Parametry × lokální proměnné	115
Životnost lokálních proměnných	115
5.8 Příklady	116
Jídlna.....	116
Návratová hodnota	116
Definice metod v editoru.....	117
5.9 Metody s proměnným počtem argumentů.....	118
5.10 Přehled definovaných metod	119
5.11 Syntaktický diagram	119
5.12 Shrnutí.....	120
6 Ostatní operátory	121
6.1 Inkrementační a dekrementační operátory ++ --	121
6.2 Porovnávací operátory < <= == != >= >	123
Testování shody desetinných čísel	124
Zvláštnosti porovnávání stringů	125
p12 == false.....	126
p13 == true.....	126
p23 == false.....	127
Porovnávání objektů reprezentujících hodnotu	127
6.3 Logické operátory ! & && 	127
6.4 Bitové operátory ~ & ^ << >> >>>	130
6.5 Složené přiřazovací operátory Op=.....	133
Příklady využití přetypování	133
6.6 Ternární operátor ?: – podmíněný výraz.....	134
Ještě jednou porovnávání reálných čísel	136
6.7 Přepínač – výraz switch	137
Pravidla.....	138
Příklad.....	138
6.8 Operátor instanceof	139
Shoda vzorů pro operátor instanceof.....	141
6.9 Zbylé operátory: new [] ()	142
Operátor new	142
Operátor []	144
Operátor . (tečka).....	144
Operátor volání metody ()	144
6.10 Priorita, asociativita a komutativita operátorů	145
Priorita	145
Asociativita.....	146
Komutativita	147
6.11 Shrnutí.....	147
7 Pole.....	148
7.0 Představení kapitoly	148
7.1 Strukturovaný datový typ – kontejner – pole	149
7.2 Deklarace a inicializace polí.....	150
Syntaxe zděděná od jazyků C/C++.....	151
7.3 Přiřazení hodnoty polí a přetypování polí.....	151
7.4 Počet prvků pole	153
7.5 Práce s prvky pole.....	154
7.6 Vícerozměrná pole – pole polí.....	155
Obdélníková pole.....	156

Zubatá pole.....	157
Inicializace dvourozměrného pole	159
Inicializace vícerozměrného pole.....	159
7.7 Proměnný počet argumentů metod.....	160
7.8 Arrays – knihovna metod pro práci s poli.....	160
7.9 Emulace předání argumentu odkazem.....	161
7.10 Pole a moderní programování.....	162
7.11 Shrnutí.....	163
8 Rozhodování.....	164
8.1 Jednoduchý podmíněný příkaz.....	164
8.2 Blok příkazů (složený příkaz)	166
Vnořování bloků.....	167
Proměnné lokální v bloku	167
8.3 Úplný podmíněný příkaz.....	170
8.4 Složený podmíněný příkaz.....	171
8.5 Přepínač – příkaz <code>switch</code>	174
Pravidla.....	174
Příklad.....	176
8.6 Shrnutí.....	179
9 Opakování části kódu	180
9.1 Obecný cyklus.....	180
9.2 Cyklus s ukončovací podmínkou – cyklus <code>do...while</code>	181
9.3 Cyklus s počáteční podmínkou – cyklus <code>while</code>	182
9.4 Cyklus s parametrem – cyklus <code>for</code>	184
Metody s proměnným počtem argumentů.....	186
9.5 „Dvojtečkový“ cyklus <code>for</code> (cyklus „ <code>for each</code> “).	188
9.6 Vnořování cyklů.....	189
9.7 Cyklus s prázdným tělem	191
9.8 Nekonečný cyklus.....	192
9.9 Cyklus s podmínkou uprostřed	193
9.10 Příkaz <code>break</code> s návěštím.....	194
9.11 Příkaz <code>continue</code>	197
9.12 Rekurze	198
Princip.....	198
Přímá a nepřímá rekurze.....	198
Přeplnění zásobníku návratových adres.....	199
9.13 Shrnutí.....	200

Část II Základní objektové konstrukce **201**

10 Základy objektově orientovaného paradigmatu	202
10.1 Předmluva	202
10.2 Trocha historie	203
10.3 Motivace OOP.....	203
10.4 Objekty.....	204
Členy objektů	204
10.5 Třídy a jejich instance.....	205
10.6 Třída jako objekt.....	206
10.7 Členy třídy a jejich instancí.....	207
Přežívající lokální proměnné	207
10.8 Zprávy.....	208
10.9 Metody.....	208

10.10	Entity	209
10.11	Polymorfismus, rozhraní, interfejs	209
	Rozhraní × implementace	210
	Atributy × vlastnosti	211
	Vlastnosti v knihovně/platformě/frameworku <i>JavaFX</i>	211
	Signatura × kontrakt	212
	Rozhraní × interface	212
	Interfejs a jeho instance	213
10.12	Objektové datové typy	213
10.13	Dědění	214
	Přirozené (nativní) dědění	215
	Dědění typu (rozhraní)	215
	Dědění implementace	216
	Problémy s děděním – substituční princip Liskové (LSP)	216
10.14	Vlastní instance třídy a mateřská třída objektu	217
10.15	Tři základní principy OOP	217
10.16	Jazyk UML	218
10.17	Správa paměti	219
10.18	Shrnutí	220
11	Třídy a jejich členy	221
11.1	Nejjednodušší definice třídy	221
11.2	Konstruktory	222
	Implicitní konstruktor	222
	Vlastní konstruktor a skrytý parametr this	222
	Proč se liší podpisy	223
	Definice tříd jako úryvky	224
11.3	Třída se všemi členy	224
	Statické (třídní) členy	225
	Instanční členy	226
	Konstrukce objektů	227
11.4	Kvalifikace posílaných zpráv	227
	Implicitní kvalifikace	228
11.5	Přetěžování konstruktorů	229
	Kvalifikace klíčovým slovem this	232
11.6	Modifikátory přístupu a skrývání implementace	233
	Veřejné a „neveřejné“ datové typy	234
11.7	Přístupové metody	234
11.8	Modifikátor final	235
	Konstantní atributy	236
	Konstanty vyhodnotitelné v době překladu	236
	Konstantní lokální proměnné	236
	Efektivní konstanty	237
	Zveřejňování konstantních atributů	237
	Modifikátor final v procesu dědění	237
	Neměnnost objektů	237
11.9	Primitivní a obalové datové typy – autoboxing	238
	Převody stringů na hodnoty primitivních typů	239
11.10	Důležité metody klíčových tříd	240
	Třída Object	240
	Object clone()	240
	Mělké a hluboké kopie objektů	240
	boolean equals(Object)	241
	Class<?> getClass()	241
	int hashCode()	241
	String toString()	241

void finalize()	241
Třída String	242
boolean equals(Object)	242
Třída Class	242
boolean equals(Object)	242
String getName()	242
String getSimpleName()	242
String toString()	242
11.11 Shrnutí	242
12 Vývojová prostředí a vytvoření aplikace	243
12.1 IDE	243
BlueJ a BlueJ++	244
Nejpoužívanější IDE	244
12.2 Instalace a spuštění NetBeans	245
12.3 Vytvoření spustitelného projektu v NetBeans	245
Vytvoření nového projektu	245
Vytvoření nové třídy	247
Definice hlavní metody	249
12.4 Překlad a sestavení projektu	250
Nastavování parametrů překladu	251
Nový překlad a sestavení	252
Sestavení	253
12.5 Spuštění aplikace	253
Spustitelnost JAR-souboru – hlavní třída aplikace	253
Nastavení parametrů virtuálního stroje	255
Soubor MANIFEST.MF	255
Spuštění aplikace z příkazového řádku	256
Syntaktický diagram spuštění aplikace	257
Java	257
ArgumentVM	257
Spouštěná třída	257
ArgumentProgramu	258
Příklady	258
12.6 Zobrazování varovných hlášení	259
Doporučení	259
Vypnutí konkrétního hlášení	259
Proč vypínat varování	260
12.7 Shrnutí	261
13 Balíčky a knihovny	262
13.1 Velké programy a jejich problémy	262
13.2 Balíčky	263
Umístění zdrojových souborů	264
Kořenový (implicitní, defaultní, nepojmenovaný) balíček	264
Podbalíčky	265
Konvence pro názvy balíčků	265
Balíčky doprovodných programů a knihoven	265
13.3 Balíčky a NetBeans	266
13.4 Rozšiřujeme strom balíčků	267
Názvy tříd	269
13.5 Explicitní ukončení aplikace	270
13.6 Příkaz import	271
Import zadaného datového typu	271
Import všech typů ze zadaného balíčku	272
Podpora zadávání příkazu import ve vývojových prostředích	272
Výjimečnost balíčku java.lang	273

13.7	Příkaz import static	273
13.8	Syntaktický diagram	274
13.9	Používání knihoven	274
13.10	Typy se stejným názvem v různých balíčcích	277
	Shrnutí	279
13.11	Použití knihovny v JShell	279
	Nastavení proměnné <code>classpath</code>	280
	Nastavení importů	280
	Násilné ukončení aplikace	281
13.12	Shrnutí	282
14	Dokumentace API	283
14.1	Dokumentační komentáře a API	283
14.2	Proč psát srozumitelné a komentované programy	284
	POBLIOCHA	285
	Jak dokumentační komentáře zobrazovat	286
14.3	Jak psát dokumentační komentáře	286
14.4	Pomocné značky pro tvorbu dokumentace	287
14.5	Dokumentace balíčku a modulu	288
14.6	Vytvoření a zobrazení dokumentace	290
14.7	Struktura dokumentace API	292
	Struktura dokumentace datového typu	292
	Rychlé vyhledání	293
14.8	Zpřehlednění programu	293
14.9	Zakomentování a odkomentování části programu	295
14.10	Shrnutí	295
15	Konstrukce interface	296
15.1	Definice typického interfejsu	296
	Deklarace abstraktních metod	297
	Příklad	297
15.2	Implementace interfejsu třídou	298
15.3	Interfejs se všemi přípustnými typy členů	300
	Motivace k rozšíření – implicitní metody	300
	Statické členy	302
	Instanční členy	302
15.4	Dědění interfejsů	303
15.5	Příklad	303
	Plynulé posuny	303
	Plynulé změny velikosti	304
	Sloučení knihoven	306
15.6	Výhody implicitních metod při návrhu architektury	306
15.7	Řešení kolizí	307
15.8	Specifikace zdroje použité metody	309
	Možné problémy	309
15.9	Speciální interfejsy	311
	Značkovací interfejsy	311
	<code>java.lang.Cloneable</code>	311
	<code>java.io.Serializable</code>	311
	Současné trendy a doporučení	311
	Funkční interfejsy	311
	Interfejs <code>Iterable</code>	312
15.10	Shrnutí	313

16	Podrobnosti o konstruktorech	314
16.1	Opakování: co víme o konstruktorech instancí	314
16.2	Zavádění třídy – <code>java.lang.ClassLoader</code>	315
16.3	Statický konstruktor – konstruktor třídy	316
	Konstruktor interfejsu	316
16.4	Instanční inicializační blok	317
16.5	Dvě části těla konstrukturu instancí	317
16.6	Příklad	318
	Konstruktor třídy	324
	3-9 : Úvodní statický inicializační blok	324
	25 : Předčasné použití atributu	324
	8 : Nekorektní použití metod	324
	42 : Předčasné použití konstanty	324
	62 : Nekorektní volání konstrukturu	325
	Inicializační část konstrukturu instancí	325
	12-15 : Úvodní instanční inicializační blok	325
	149 : Deklarace konstanty <code>loaded</code>	326
	153-157 : Inicializační výpočet	326
	165 : Použití <code>this</code> v inicializaci	326
	266-269 : Závěrečný inicializační blok	326
	Tělo konstrukturu instancí	326
	177-182 : Bezparametrický konstruktor	326
	190-196 : Jednparametrický konstruktor	327
	205-210 : Dvoupametrický konstruktor	327
	220-233 : Tříparametrický konstruktor	327
16.7	Experimenty	327
16.8	Doporučení	328
	Jediný statický inicializační blok	328
	Bez instančních inicializačních bloků	328
	Inicializovat všechny atributy jednotně	329
16.9	Skutečný název metody konstrukturu	329
16.10	Shrnutí	330
17	Úvod do dědění implementace	331
17.0	Představení kapitoly	331
17.1	Úvodní poznámky	332
17.2	Definice dceřiné třídy	332
17.3	Rodičovský podobjekt	334
	Dědění implementace od více rodičů	335
17.4	Konstruktor	336
	Konstrukce rodičovského podobjektu	336
17.5	Přetížené verze konstrukturu – použití <code>super</code> × <code>this</code>	338
17.6	Konstruktory rodiče a potomka	340
17.7	Demonstrace chování konstrukturu	340
	Definice třídy <code>Graddaughter</code>	340
	Provedení akce před příkazem <code>this()</code> nebo <code>super()</code>	341
	Definice metody <code>constructorReport(Object, Class)</code>	343
	Spuštění testu	344
	Zavedení třídy	344
	Tisk nehotových objektů	344
	Preference vlastních metod	345
	Dokončení testu	345
	Rodičovský podobjekt je abstrakce	345
17.8	Zákaz vytváření potomků třídy	347
17.9	Shrnutí	347

18 Viditelnost členů tříd	348
18.1 Úpravy použitého projektu.....	348
18.2 Trocha terminologie.....	349
Posílání zpráv a volání metod.....	349
Přetěžování×přebíjení×zakrývání×přepisování×předefinování metod	349
Přetěžování metod.....	349
Přebíjení metod.....	349
Zakrývání metod.....	350
Přepsání či předefinování metod.....	350
18.3 Chráněné členy – modifikátor přístupu protected	350
Shrnutí.....	353
18.4 Dědění metod	353
Zděděné, dále neupravované metody.....	353
Zděděné metody, pro něž potomek definuje „lepší“ implementaci.....	354
Kompatibilita signatur.....	354
18.5 Zakrývání metod předka (method hiding).....	355
18.6 Metody, které není možno v potomku zakrýt či přebít – modifikátor final	357
18.7 Zakrývání atributů předka	359
18.8 Metody nově definované v potomku	360
Proč je situace jednoduchá jen zdánlivě.....	361
Anotace @Override	361
Statically × dynamicky typované jazyky.....	362
18.9 Závěr	363
18.10 Shrnutí.....	363
19 Virtuální metody a jejich přebíjení	364
19.1 Princip.....	364
Časná a pozdní vazba.....	365
Virtuální metody.....	365
19.2 Které metody jsou v Javě virtuální	366
19.3 Chování virtuálních metod.....	366
19.4 Zdokonalení třídy Square	368
Přebíjení metody copy()	369
Problémy s nastavováním velikosti.....	369
První návrh definice metody setSize(int, int)	370
Test prvního návrhu.....	371
Oprava.....	372
19.5 Co se nám na dědění nelíbí	374
19.6 Shrnutí.....	374
20 Abstraktní třídy	375
20.1 Abstraktní třídy a jejich role v dědické hierarchii	375
Vytváříme hybrida.....	376
Abstraktní třída bez abstraktních metod.....	377
20.2 Konstruktor abstraktní třídy	377
20.3 Deklarace a implementace abstraktních metod.....	378
20.4 Účel abstraktních tříd	380
20.5 Proč společný rodič	380
20.6 Účel abstraktních metod	381
20.7 Návrhový vzor Šablonová metoda (Template method)	381
Princip.....	381
Implicitní metody interfejsů.....	382
Architektura balíčku eu.pedu.lib20s.geom	382
Metoda toString()	384
20.8 Shrnutí.....	386

Část III Pokročilé objektové konstrukce	387
21 Výjimky a aserce	388
21.0 Představení kapitoly	388
21.1 Co to jsou výjimky	389
21.2 Analýza chybové zprávy	389
Oznámení o chybě	389
Jak chyba vznikla – výpis zásobníku návratových adres	390
21.3 Nejdůležitější výjimky	392
21.4 Vyhození výjimky	393
Oddělené vytvoření výjimky	394
21.5 Výjimky a nedosažitelný kód	395
21.6 Co výjimky umí	395
21.7 Hierarchie dědění výjimek	396
21.8 Zachycení vyhozené výjimky	398
Chování metody <code>exceptionCatching(int)</code>	399
21.9 Syntaktický diagram bloku <code>try ... catch</code>	400
Několik současně odchyťovaných výjimek	400
Společná reakce na několik výjimek	401
Společný úklid – blok <code>finally</code>	402
Příklad	402
21.10 Definice vlastních výjimek	404
21.11 Kontrolované výjimky	405
21.12 Převedení kontrolované výjimky na nekontrolovanou	407
21.13 Informace o skutečném původci výjimky	408
21.14 Ověřování podmínek – příkaz <code>assert</code>	410
Design by Contract	411
21.15 Kdopak mne to volal	414
21.16 Shrnutí	415
22 Generické datové typy a metody	416
22.1 Motivace	416
22.2 Generické a parametrizované datové typy	419
22.3 Definice generických typů	422
22.4 Použití generických typů	423
22.5 Překlad generických datových typů a očišťování	425
22.6 Rizika nepoužití typových argumentů	425
22.7 Varování překladače a jejich potlačení	428
Varování o použití předběžných funkcí	429
Další varování	430
22.8 Generické metody	430
22.9 Shrnutí	435
23 Typové parametry a argumenty	436
23.1 Omezení typových argumentů	436
Typové argumenty s více předky	437
Vzájemné závislosti typových parametrů	437
23.2 Překlad a očišťování podrobněji	438
Doporučené pořadí omezujících interfejsů	438
Ztráta informace při běhu	440
Přemostovací metody	440
23.3 Zakázané operace	442
Za typové parametry nelze dosazovat primitivní typy	442
Typové parametry třídy není možno použít u statických členů	442
Nelze vytvořit instanci typového parametru	442

Reflexe	443
Nelze vytvořit pole instancí typového parametru ani parametrizovaného typu.....	444
Výjimky	445
23.4 Proměnný počet argumentů – @SafeVarargs	445
Omezení.....	446
Vytvoření pole hodnot.....	447
23.5 Nejednoznačnosti a kolize.....	447
Falešně přetížená metoda.....	447
Nová metoda koliduje se zděděnou.....	448
Kolize požadovaných interfejsů.....	449
Kolize implementovaných interfejsů.....	450
Potomci a předci generických typů – špatné pochopení dědičnosti.....	451
23.6 Žolíky.....	452
23.7 Příklad: datový typ <code>Interval<T extends Comparable<? super T></code>	453
23.8 Shrnutí.....	456
24 Interní datové typy	457
24.1 Motivace.....	457
Pomocný soukromý typ.....	458
Objekt znající útroby a implementující veřejné rozhraní.....	458
Sdružení souvisejících typů	458
24.2 Terminologie.....	459
24.3 Společné charakteristiky interních typů	460
24.4 Globální interní (členské) datové typy	461
24.5 Vnořené datové typy	461
24.6 Vnitřní třídy	462
Interní interfejsy a výčtové typy bez modifikátoru <code>static</code>	462
24.7 Příklad na vnořené a vnitřní třídy	463
Vnořená <code>Elements</code> × vnitřní <code>SAIterator</code>	464
Veřejná <code>Elements</code> × soukromá <code>SAIterator</code>	465
Definice třídy <code>SparseArray.Element</code>	465
Definice třídy <code>SparseArray.SAIterator</code>	466
Definice třídy <code>SparseArrayTest</code>	466
24.8 Lokální třídy.....	469
Pojmenované lokální třídy	470
Anonymní třídy	470
Použití anonymních tříd.....	471
24.9 Shrnutí.....	471
25 Výčtové typy – třídy typu <code>enum</code>	472
25.1 Nejjednodušší definice	472
Překladačem přidané atributy a metody	474
Atribut <code>\$VALUES</code>	474
<code>public static final NázevTypu[] values()</code>	474
<code>public static NázevTypu valueOf(String name)</code>	474
25.2 Třída <code>Enum</code>	474
Zděděné metody	475
<code>public final String name()</code>	475
<code>public final int ordinal()</code>	475
<code>public final int compareTo(E o)</code>	475
<code>public final Class<E> getDeclaringClass()</code>	476
<code>public static <T extends Enum<T>> T valueOf(Class<T> enumType, String name)</code>	476
Přebité verze metod zděděných od třídy <code>Object</code>	476
<code>protected final Object clone() throws CloneNotSupportedException</code>	476
<code>equals(Object)</code>	476

hashCode()	476
public String toString()	476
Serializace	476
25.3 Použití výčtových typů v programu	477
Přepínač	477
Přidaná anonymní třída	478
Cyklus	479
25.4 Složitější definice výčtových typů	479
25.5 Akční výčtové typy	483
Class-objekty instancí funkčních výčtových typů	484
25.6 Shrnutí	485
26 Záznamy – třídy typu record	486
26.1 Teoretická přehleda	486
Odkazové a hodnotové datové typy	487
Proměnné a neměnné hodnotové typy	487
Převrácení	488
Problémy a motivace	488
Koncepční vlastnosti	489
26.2 Základní syntaxe	489
26.3 Automaticky definované členy	490
26.4 Některé možnosti	490
26.5 Kanonický konstruktor	491
26.6 Možnosti a omezení	493
26.7 Komplexnější definice	494
Prověra konstruktorů	496
Definované metody	497
Řešení vzniklých problémů	499
26.8 Shrnutí	499
27 Lambda-výrazy	500
27.1 Motivace	500
27.2 Koncepce lambda-výrazů	501
27.3 Funkční interfejsy	501
27.4 Syntaxe lambda-výrazů	504
Jednoduchý příklad	505
Lambda-výrazy zastupující metody	505
Lambda-výraz zastupující konstruktor	506
27.5 Použití lokálních proměnných z okolního bloku	508
27.6 Shrnutí	509
28 Anotace	510
28.1 Co jsou anotace	510
28.2 Označování deklarací anotacemi	511
28.3 Kde všude můžeme anotace použít	513
Anotování balíčků	514
Anotování parametru this	515
28.4 Anotace ve standardní knihovně	515
Standardní anotace v balíčku java.lang	515
@Deprecated	515
@Override	516
@SuppressWarnings	516
@SafeVarargs	516
@FunctionalInterface	517
Standardní anotace v balíčku javax.annotation	517
Metaanotace	517

@Documented.....	517
@Inherited.....	517
@Repeatable.....	518
@Retention.....	518
@Target.....	518
28.5 Syntaxe definice anotací.....	519
Jednoduchá značkovácí anotace.....	521
28.6 Získávání informací o anotacích za běhu programu.....	522
28.7 Shrnutí.....	522
29 Vlákna a paralelní procesy.....	523
29.1 Paralelní provádění více činností.....	523
Kooperativní plánování.....	524
Preemptivní plánování.....	524
Použité plánování.....	524
29.2 Vlákna a jejich stavy.....	524
29.3 Sdílení zdrojů.....	526
29.4 Kritické sekce a monitory.....	527
29.5 Synchronizace.....	527
29.6 Uvolnění kritické sekce.....	528
29.7 Jemnější způsoby synchronizace.....	529
Modifikátor <i>volatile</i>	529
Atomické objekty.....	529
29.8 Novější metody práce s vlákny.....	529
29.9 Shrnutí.....	530
30 Moduly.....	531
30.1 Motivace.....	531
Problémy předchozích verzí <i>Javy</i>	532
Cíle projektu <i>Jigsaw</i>	533
Dosažené výhody.....	533
30.2 Srovnání instalace <i>Javy 8</i> a <i>Javy 14</i>	534
30.3 Modul × Baliček.....	535
30.4 Soubor <i>module-info.java</i>	536
Syntaktický diagram deklarace modulu.....	536
Název modulu.....	538
Direktiva <i>requires</i>	538
Direktiva <i>exports</i>	539
Direktiva <i>opens</i> a modifikátor <i>open</i>	539
Direktiva <i>uses</i>	539
Direktiva <i>provides</i>	539
30.5 Modulární JAR-soubor.....	540
30.6 Proměnná <i>modulepath</i>	540
30.7 Vytvoření modulární aplikace.....	540
Vytvoření projektu.....	541
Definice modulu.....	542
Přidání zdrojových souborů.....	543
Odstraňování chyb z nepokrytých závislostí.....	544
Přidání modulu <i>eu.pedu.lib20s.geom</i>	546
Přidání modulu <i>eu.pedu.lib20s.canvas</i>	547
Přidání modulu <i>eu.pedu.lib20s.canvasmanager</i>	548
Závěrečná podoba deklarací modulů.....	549
JAR-soubor s více moduly.....	549
30.8 Klasifikace modulů.....	549
Běžný modul (<i>normal module</i>).....	550
Otevřený modul (<i>open module</i>).....	550

Automatický modul (automatic module).....	550
Vlastnosti automatických modulů	552
Nepojmenované moduly (unnamed modules).....	552
30.9 Moduly a platforma <i>JShell</i>	552
30.10 Shrnutí.....	553

Část IV Přílohy 555

A Omezení JVM	556
B Tvorba jednoduchého GUI.....	557
B.1 Trocha historie – přehled rozšířených platform	557
Platforma AWT.....	557
Platforma Swing.....	558
Platforma SWT.....	558
Platforma JavaFX.....	558
B.2 Základní koncepce platform pro tvorbu GUI.....	559
GUI a využívání vláken.....	559
Modalita dialogových oken.....	560
B.3 Prostředky pro triviální okenní vstup a výstup.....	560
Třída <code>eu.pedu.lib20s.util.IO</code>	560
<code>void setDialogsPosition(int x, int y)</code>	561
<code>boolean confirm(Object question)</code>	561
<code>int choose(Object question, String... buttons)</code>	561
<code>double enter(Object prompt, double defaultDouble) int</code> <code>enter(Object prompt, int defaultInt) String enter(Object prompt,</code> <code>String defaultString)</code>	561
<code>String select(Object prompt, String... options)</code>	561
<code>void inform(Object text)</code>	561
Třída <code>javax.swing.JOptionPane</code>	561
<code>showMessageDialog</code>	561
<code>showConfirmDialog</code>	561
<code>showOptionDialog</code>	562
<code>showInputDialog</code>	562
B.4 Základy koncepce platform AWT a Swing.....	562
Komponenty a kontejnery.....	562
Správce rozvržení – <code>LayoutManager</code>	562
<code>BorderLayout</code>	563
<code>BoxLayout</code>	563
<code>FlowLayout</code>	563
Kontejnery – okna a panely.....	563
<code>JFrame</code>	563
<code>Box</code>	563
<code>JPanel</code>	563
Události a jejich posluchači.....	564
B.5 Příklad.....	564
Literatura.....	571
Rejstřík.....	573

Úvod

Tato kniha seznamuje se čtrnáctou verzí jazyka *Java*. Soustředí se především na výklad vlastností jazyka a minimalizuje výklad probírající obsah standardní knihovny – tomu se budou věnovat další příručky z této série.

Komu je kniha určena

Kniha je určena všem, kteří se chtějí naučit jazyk *Java*. Nevyžaduje od čtenáře žádné předchozí znalosti programování a snaží se, byť stručně, vysvětlit vše, co je potřeba.

Naprostým začátečníkům bych sice doporučil, aby začali s některou z mých příruček úvodu do programování, v nichž nevysvětluji, jak se program zapisuje, ale soustředím se především na to, jak se vymýšlí. V těchto příručkách spolu navrhujeme architekturu programu. Zakódování navrženého programu (tj. jeho zápis v programovacím jazyce) přenecháváme generátoru kódu, který je integrální součástí použitého vývojového prostředí. K výkladu zápisu programu v kódu, na nějž se soustředí tato učebnice, pak přejdeme až v okamžiku, kdy složitost navrhovaných programů překročí možnosti onoho generátoru. V té době ale již mají účastníci kurzu základy objektivě funkcionálního paradigmatu zažité a nedělají proto chyby, s nimiž zápasí studenti, kteří začali studiem syntaxe použitého programovacího jazyka.

Na druhou stranu nechci nikoho zrazovat, a proto jsem se v této knize pokusil důkladně vysvětlit i základní programové konstrukce a doporučené způsoby jejich využití, takže i naprostí začátečníci zde najdou všechny potřebné informace.

Jak už jsem řekl, kniha se soustředí na výklad jazyka. Tím, že výklad knihoven (např. práce s kolekcemi, datovody, soubory a datovými proudy, regulárními výrazy či vlákny) odložím do jiných příruček, ušetřím prostor, který mohu věnovat podrobnému výkladu těch vlastností jazyka, na které v jiných příručkách často nezbyvá místo. Jejich neznalost ale vede k hodinám marného pátrání po skutečném původci vzniklé chyby a následným experimentálním změnám programu s myšlenkou: „Co kdyby to náhodou vyšlo?“

Koncepce výkladu

Značná část učebnic začíná definicí programu typu *Hello world*, což je koncepce převzatá z historické učebnice jazyka C vydané v roce 1978. Základní nevýhodou této koncepce je, že na počátku používá několik programových konstrukcí, které čtenáři vysvětlí až někde v průběhu dalšího výkladu.

Tehdy to dost dobře jinak ani nešlo. Od té doby se objevila řada nástrojů, které umožňují koncipovat výklad tak, aby se v něm používaly pouze konstrukce, které již byly vysvětleny. O to se pokouším i v této knize a používám doposud nevysvětlené konstrukce opravdu výjimečně pouze v situacích, kdy to výrazně zpřehlední výklad. Aby se mi to podařilo, používám k výkladu program *JShell*, který je standardní součástí vývojářské sady.

Knihu jsem se navíc pokusil napsat tak, aby mohla sloužit stejně dobře jako učebnice jazyka i jako referenční příručka, ve které by bylo možno v případě potřeby najít klíčové informace rychleji a snáze než na internetu. Pasáže vysvětlující jednotlivé programové konstrukce jsem se proto snažil jasně oddělit od pasáží probírajících teorii, jak řešit tu kterou třídu problémů, a sloužících především začátečníkům, kteří ještě nemají s programováním žádné zkušenosti.

Rozdělení textu

Text je rozdělen do čtyř částí. První část začíná úvodem do prostředí *JShell*, které bude v celé první části využíváno ke spouštění demonstračních programů. Zbytek první části pak probírá algoritmické konstrukce a prostředky, které *Java* nabízí k jejich realizaci.

Druhá část se soustředí na základy objektově orientovaného programování. Začíná teoretickou kapitolou vysvětlující hlavní zásady objektově orientovaného paradigmatu. Po ní následují kapitoly postupně probírající základní objektové konstrukce a jejich účel.

Třetí část je věnována výkladu nadstavbových objektových konstrukcí včetně rozboru některých základních pravidel, jejichž nedostatečné pochopení dělá studentům často problémy. Závěr třetí části je věnován koncepci modularity. Modularita sice není součástí jazyka, ale platformy; nicméně její osvojení přináší výrazné zvýšení spolehlivosti vyvíjených programů a do jisté míry i efektivitu jejich vývoje i následného provozu.

Čtvrtá část obsahuje přílohy. První z nich vás seznámí s omezeními jazyka *Java*, druhá pak ukáže, jak se v *Javě* navrhuje GUI. Nejprve představí nástroje použitelné pro superjednoduché GUI, pak principiálně naznačí, jak se navrhuje komplexní GUI.

Terminologie



Při zavádění české terminologie se bohužel potýkáme často s tím, že v anglické literatuře zavádějí mnozí autoři terminologii vlastní. V oblasti, kterou se má zabývat tato publikace, je terminologický guláš obzvláště vydatný. Budu-li proto uvádět ve svém výkladu anglické ekvivalenty zaváděných termínů, budu používat terminologii zavedenou v oficiální referenční publikaci *Java® Language Specification – Java SE 14 Edition* ([5]), na níž se budu občas odvolávat zkratkou [JLS](#).

Použité nástroje

Pro úspěšné studium knihy budete potřebovat několik programů, které musíte nejprve stáhnout a instalovat.

Vývojová sada JDK

Především budete potřebovat mít instalovanou vývojovou sadu pro *Java 14* označovanou *JDK*, což je zkratka z anglického *Java Development Kit*. Stáhnete ji ze stránky <http://www.oracle.com/technetwork/java/javase/downloads/>. Jenom musíte před vlastním stažením nastavit přepínač u seznamu stáhnutelných souborů do polohy **Accept License Agreement**, protože jinak vás stránka daný program stáhnout nenechá.

Spolu s vývojovou sadou vřele doporučuji stáhnout i dokumentaci. Tu najdete na téže stránce, jenom musíte popojet trochu níže do sekce **Additional Resources**, ve které najdete podsekcí **Java SE 14 Documentation**.

Vývojové prostředí *JShell*

Pro většinu výkladu budu využívat prostředí *JShell*, které je součástí *JDK* a se kterým vás seznámí úvodní kapitola. (Podrobnější seznámení s tímto prostředím najdete v příručce [Java 9 – JShell](#).) Výhodou tohoto přístupu je, že součástí programů nemusí být nejrůznější vata, bez které byste standardní program v *Javě* nespustili. Navíc již nemusíte instalovat žádné další vývojové prostředí, takže nebudete muset zápasit s důsledky známé skutečnosti, že zvládnutí současných profesionálních vývojových prostředí je náročnější než zvládnutí základů jazyka.

V závěru knihy budu potřebovat vysvětlit některé konstrukce, které se v prostředí *JShell* dost dobře vysvětlit nedají. Až na to dojde, tak vás upozorním.

Samostatné vývojové prostředí

Od kapitoly [12 Vývojová prostředí a vytvoření aplikace](#) na straně [243](#) začnu vedle prostředí *JShell* používat i prostředí *NetBeans*. Budu v něm demonstrovat vlastnosti, které se projeví až u samostatně vyvíjených programů. Nelpím přitom na tom, abyste používali právě toto prostředí, ale doporučuji je těm, kteří ještě nemají s profesionálními vývojovými prostředími žádné zkušenosti.

Adresy, z nichž si můžete stáhnout některé z doporučovaných prostředí, najdete v pasáži [Nejpoužívanější IDE](#) na straně [244](#). Každopádně budete-li chtít experimentovat s vybraným profesionálním prostředím hned od počátku, nic vám v tom nebrání.

Doprovodné programy

Všechny doprovodné programy zmiňované a používané v textu najdete na stránce knihy na adrese http://knihy.pecinovsky.cz/62_j14lang. Zde jsou umístěny ZIP-soubory obsahující doprovodné programy nebo jejich odvozeniny.

Doprovodné programy jsou zpočátku skripty prostředí *JShell*, v druhé části to pak jsou projekty pro vývojové prostředí *NetBeans*, které lze snadno transformovat na projekty pro vaše oblíbené vývojové prostředí. Podrobnosti o struktuře doprovodných programů najdete stránce s programy.

Knihu jsem se snažil zalomit tak, aby všechny výpisy programů a komunikace s počítačem, které se vejdu na jednu stránku, byly vždy na jedné stránce a nelámaly se přes hranu stránky. Čtenářům papírové verze by se tak měly minimalizovat problémy při studiu doprovodných textů diskutujících obsah těchto výpisů.

Čtenáři elektronické verze jsou na tom s listováním hůře. Těm bych doporučil, aby si text knihy otevřeli dvakrát a umístili oba dokumenty vedle sebe. (Na většině současných počítačů s širokými monitory by to neměl být problém.) Pak lze mít v jednom dokumentu otevřenou stránku s výpisem a ve druhém dokumentu stránku s rozбором obsahu daného výpisu.

Druhou možností je, že si studovaný výpis vytisknete (v doprovodných programech najdete příslušné soubory i s čísly řádků výpisu v knize), budete se dívat do výpisu a na čteče číst doprovodný text.

Předběžné definice nových konstrukcí

Od verze 10 bylo zavedeno vydávání nových verzí dvakrát do roka: na jaře (typicky v březnu) vycházejí sudé verze, a na podzim (typicky v říjnu) verze liché. Protože dotažení návrhu některých konstrukcí trvá delší dobu, bylo v zájmu maximalizace zpětné vazby od uživatelů zavedeno, že vydané verze podporují i konstrukce, které jsou teprve ve vývoji, a nedoporučuje se je proto používat ve finálních verzích aplikací.

Ve standardní instalaci JDK se můžete setkat se třemi typy rozpracovaných konstrukcí, funkcionalit a aplikací:

Předběžná funkce/konstrukce/vlastnost (Preview Feature¹)

Jako předběžná je označena konstrukce jazyka nebo funkce VM, která je plně specifikována, plně implementována, ale nemusí být ve své definitivní verzi. Má iniciovat zpětnou vazbu vývojářů, kteří se ji pokusí použít ve svých aplikacích.

Při standardním nastavení tyto budoucí novinky překladač ani běhové prostředí nepodporují. Jejich podporu překladačem, resp. virtuálním strojem, resp. programem `javadoc`, nastavíte zadáním parametru

```
--enable-preview
```

příslušného programu (překladače, virtuálního stroje, programu `javadoc`).

Experimentální funkce/konstrukce/vlastnost (Experimental Feature)

Experimentální funkce představují rané verze funkcí, většinou na úrovni VM. Tyto funkce mohou být neúplné nebo dokonce nestabilní. Ve většině případů potřebují povolit pomocí vyhrazených příznaků.

Experimentální prvek je považován za zhruba z 25 % „hotový“, zatímco předběžný prvek by měl být „hotový“ alespoň z 95 %.

Inkubační funkce/konstrukce/vlastnost (Incubating Feature, Incubator²)

Inkubační funkce jsou experimentální API distribuované ve formě samostatných modulů se jmény s předponou `jdk.incubator`. Abyste s nimi mohli pracovat, musíte tyto moduly explicitně přidat.

Syntaktické definice a diagramy

V testu je uváděna řada nejrůznějších programových konstrukcí, jejichž zápis se řídí přesně danými syntaktickými pravidly. Většina učebnic možné způsoby zápisu zpravidla formálně nedefinuje, anebo používá syntaktické definice odvozené z Backusovy normální formy. Syntaktické definice jsou výhodné pro strojové zpracování, lidé se v nich často špatně orientují. Rozhodl jsem se proto zobrazovat syntaktická pravidla zápisu jednotlivých konstrukcí pomocí syntaktických diagramů.

Syntaktický diagram ukazuje, jak je možno zobrazovanou konstrukci zapsat. Pojede-li po čarách, tak jakýkoliv průjezd generuje syntakticky správnou konstrukci. Toho, kdo syntaktické diagramy ještě nezná a chtěl by rychle některý vidět, bych odkázal např. na diagram na obrázku [2.1](#) na straně [56](#).

¹ Podrobněji je koncepce předběžných vlastností popsána v JEP 12, které najdete na adrese <https://openjdk.java.net/jeps/12>.

² Podrobněji se o koncepci inkubačních modulů dozvíte v JEP 11, které najdete na adrese <https://openjdk.java.net/jeps/11>.

Použité typografické konvence

K tomu, abyste se v textu lépe vyznali a také abyste si vykládanou látku lépe zapamatovali, používám několik prostředků pro odlišení a zvýraznění textu.

- Termíny** První výskyt nějakého termínu a další texty, které chci zvýraznit, vysazuji **tučně**.
- Název* Názvy firem a jejich produktů vysazuji *kurzivou*. Kurzivou vysazuji také názvy kapitol, podkapitol a oddílů, na které se v textu odkazují.
- [Odkaz](#) Celá kniha je prošpikovaná křížovými odkazy na související pasáže. Není-li odkazovaný objekt (kapitola, obrázek, výpis programu, ...) na některé ze sousedních stránek, je pro čtenáře tištěné verze doplněn o číslo stránky, na níž se nachází. Čtenářům elektronické verze stačí, když na něj klepnou, a použitý prohlížeč by je měl na odkazovaný objekt ihned přenést.
- Citace** Texty, které si můžete přečíst na displeji, např. názvy polí v dialogových oknech či názvy příkazů v nabídkách, vysazuji **tučným bezpatkovým písmem**.
- Adresy* Názvy souborů a internetové adresy vysazuji *obyčejným bezpatkovým písmem*.
- Program** Identifikátory a další části programů zmíněné v běžném textu vysazuji **neproporcionálním písmem**, které je v elektronických verzích pro zvýraznění tmavě červené.
- metoda(?)* Při odkazech na metody budu v závorkách za názvem metody vždy uvádět seznam typů jejich parametrů – např. `equals(Object)`. Nebude-li v danou chvíli jasné, jaké má zmiňovaná metoda parametry, budu do závorek psát otazník – např. `metoda(?)`.
- zadání** Ve výpisech komunikace s prostředím *JShell* bude zadání uživatele vysazeno tučně (v elektronických verzích pro zvýraznění tmavě červeně), a odpovědi prostředí netučně (a černě).

Komentář Pokud se v programech objeví komentáře, budou podbarveny zeleně.

Kromě výše zmíněných částí textu, které považuji za důležité zvýraznit nebo alespoň odlišit od okolního textu, najdete v textu ještě řadu doplňujících poznámek a vysvětlivek. Všechny budou v jednotném rámečku, který bude označen ikonou charakterizující druh informace, kterou vám chce poznámka či vysvětlivka předat.



Symbol jing-jang bude uvozovat poznámky, s nimiž se setkáte na počátku každé kapitoly. Zde vám vždy prozradím, co se v dané kapitole naučíte.



Píšící ruka označuje obyčejnou poznámku, která pouze doplňuje informace z hlavního proudu výkladu o nějakou zajímavost.



Symbol znamení raka označuje upozornění, že následný výklad se týká funkcionality, která je prozatím definována jako předběžná (preliminary). Lze očekávat, že bude v některé z příštích verzí zprovozněna, ale výsledné vlastnosti se mohou od stávajících lišit.



Otevřená schránka s dopisy označuje informace o projektu, s nímž budeme v dalším textu pracovat, nebo v něm najdete vzorové řešení aplikující probranou látku. Příslušný projekt získáte pomocí generátoru projektů popsaného výše.



Obrázek knihy označuje poznámku týkající se používané terminologie. Tato poznámka většinou upozorňuje na další používané termíny označující stejnou skutečnost nebo na konvence, které se k probírané problematice vztahují. Seznam všech terminologických poznámek najdete v rejstříku pod heslem „terminologie“.



Ruka s hrozícím prstem upozorňuje na věci, které byste měli určitě vědět a na které byste si měli dát pozor, protože jejich zanedbání vás většinou dostane do problémů.



Usměváček vás bude upozorňovat na různé tipy, kterými můžete vylepšit svůj program nebo zefektivnit svoji práci.



Mračoun vás naopak bude upozorňovat na různá úskalí programovacího jazyka nebo programů, s nimiž budeme pracovat, a bude vám radit, jak se těmto nástrahám vyhnout či jak to zařídit, aby vám alespoň pokud možno nevadily.



Brýle označují tzv. „poznámky pro šfouraly“, ve kterých se vás snažím seznámit s některými zajímavými vlastnostmi probírané konstrukce nebo upozorňuji na některé souvislosti, avšak které nejsou k pochopení látky nezbytné.

Odbočka – podšeděný blok

Občas je potřeba vysvětlit něco, co nezapadá přímo do okolního textu. V takových případech používám podšeděný blok se silnou čarou po straně. Tento podšeděný blok je takovou drobnou odbočkou od ostatního výkladu. Nadpis podšeděného bloku pak najdete i v podrobném obsahu mezi nečíslovanými nadpisy.

Zpětná vazba

Přes veškeré úsilí, které jsme knize já i moji spolupracovníci věnovali, nemohu vyloučit, že v textu či doprovodných příkladech zůstaly skryté nějaké chyby. Předem se za ně omlouvám. Objevíte-li proto v knize nějakou chybu nebo budete-li mít návrh na nějaké její vylepšení, pošlete prosím e-mail s předmětem 62_Java14_DOTAZ na adresu rudolf@pecinovsky.cz. Na stránku knihy http://knihy.pecinovsky.cz/62_j14lang se pak pokusím co nejdříve zanást příslušná errata s opravou, kterou zapracujeme do případného dalšího vydání.

Tento mail pošlete i v případě, když vám bude někde připadat text nepřiliš srozumitelný nebo budete-li mít nějaký dotaz, ať už k vykládané látce či použitému vývojovému prostředí. Bude-li se dotaz týkat něčeho obecnějšího, zveřejním na stránce knihy odpověď i pro ostatní, které by mohl obdobný dotaz napadnout za pár dní, anebo jsou natolik ostýchaví, že si netroufnou sami se zeptat.

Dopředu se omlouvám, že vzhledem k velkému pracovnímu zatížení občas odpovídám na maily až se značným zpožděním.

Část I: Neobjektové konstrukce

Tato část vás nejprve seznámí s prostředím *JShell*, které budu ve svém příkladu využívat k tomu, abychom získali okamžitou odpověď na zadané programové obraty. Budeme tak moci od začátku zkoušet probírané konstrukce, aniž bychom museli používat něco, co jsme ještě neprobrali. Poté postupně probereme algoritmické konstrukce, které vyšší programovací jazyky nabízely ještě před příchodem objektového paradigmatu, kterému bude věnována druhá část.