

knihovna programátora

- Učebnice pro ty, kteří nechtějí zůstat obyčejnými kodéry, ale chtějí se stát špičkovými architekty
- Probírá novinky Javy 8, které ovlivňují návrh architektury programu
- Soustředí se na návrh programů a osvojení klíčových architektonických zásad
- Vysvětluje a procvičuje návrhové vzory, refaktoraci kódu, vývoj řízený testy a další oblasti, které běžné učebnice ignorují
- Vše průběžně procvičuje na příkladech řešených spolu se čtenářem
- Doporučená učebnice na řadě středních škol i univerzit



RUDOLF PECINOVSKÝ

Java 8

Úvod do objektové architektury
pro mírně pokročilé

Java 8

úvod do objektové architektury
pro mírně pokročilé

Rudolf Pecinovský
2014

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 8

Úvod do objektové architektury pro mírně pokročilé

TIRÁŽ TIŠTĚNÉ PUBLIKACE

Vydala Grada Publishing a.s.
U Průhonu 22, Praha 7
jako svoji 5670. publikaci

Odborní lektori:

doc. Ing. Pavel Herout, Ph.D.,
doc. MUDr. Jiří Kofránek, CSc.,
doc. Ing. Vojtěch Merunka, Ph.D.,
doc. Ing. Miroslav Virius, CSc.

Odpovědný redaktor: Martin Vondráček, Ladislava Soukupová

Návrh vnitřního layoutu: Rudolf Pecinovský

Zlom: Rudolf Pecinovský

Počet stran 656

První vydání, Praha 2014

Vytiskla tiskárna PROTISK, s. r. o.

V knize použité názvy mohou být ochrannými známkami nebo registrovanými ochrannými známkami příslušných vlastníků.

Copyright © Grada Publishing, a.s., 2014

Cover Photo © fotobanka Allphoto.cz

ISBN 978-80-247-4638-8

TIRÁŽ ELEKTRONICKÉ PUBLIKACE

ISBN 978-80-247-9480-8 (ve formátu PDF)

ISBN 978-80-247-9481-5 (ve formátu EPUB)

*Mé ženě Jarušce a dětem
Štěpánce, Pavlínce, Ivance a Michalovi*

Stručný obsah

Skrytí spoluautoři	22
Úvod	23

Část I: Vývojové prostředí 29

1. Co byste měli znát z prvního dílu	30
2. Vývojové prostředí <i>NetBeans</i>	46
3. Projekty v <i>NetBeans</i> – Library	76
4. Vytváříme nový projekt – AHA	99
5. Práce na připraveném projektu – Elevator	111
6. Spolupráce projektů – Vehicle	138
7. Testovací třída – VehicleTest , Robot	162
8. Ladění programů – Robot	190

Část II: Vylepšování architektury 201

9. Program ve výjimečné situaci	202
10. Návrhový vzor <i>Tovární metoda</i>	228
11. Návrhový vzor Stav – Robot4	243
12. Návrhový vzor Stavitel – RingBuilder	260
13. Návrhový vzor <i>Dekorátor</i> – SmoothVehicle	284
14. Implicitní implementace – RingVehicle , ControlledVehicle	300
15. Generické datové typy a metody	320
16. Pokročilejší práce s typovými parametry	342
17. Funkční interfejsy a lambda-výrazy	358
18. Rekurzivní volání	386
19. Interní datové typy	397
20. Kontejnery a datovody	424

Část III: Dědění implementace	455
<hr/>	
21. Podrobnosti o konstruktorech tříd a instancí	456
22. Úvod do dědění implementace: Mother – Daughter – Granddaughter	473
23. Zakrývání atributů a metod	498
24. Virtuální metody a jejich přebíjení	515
25. Pasti a propasti dědění implementace	532
26. Vytváříme rodičovskou třídu – ARobot1	555
Část IV: Další užitečné programové konstrukce	575
<hr/>	
27. Učíme program přemýšlet	576
28. Ještě jednu rundu, prosím	603
29. Další důležité datové struktury	619
30. O čem jsme ještě nehovořili	638
Rejstřík	642

Podrobný obsah

Skrutí spoluautoři	22
Úvod	23
Komu je kniha určena.....	23
Koncepte knihy	23
Co se naučíte, uspořádání knihy.....	24
Programovací jazyk	25
Potřebné vybavení.....	25
Doprovodné projekty.....	26
Doplňková literatura	26
Použité konvence	27
Místní nabídka	27
Formátování	27
Odbočka.....	28
Část I: Vývojové prostředí	29
1. Co byste měli znát z prvního dílu	30
1.1 Přehled látky prvního dílu.....	30
1.2 Definice × deklarace	31
1.3 Co je to objekt.....	31
1.4 Datový typ, třída, class-objekt.....	32
1.5 Zpráva × metoda, polymorfismus	33
1.6 Rozhraní × interfejs	33
1.7 Zapouzdření a skrývání implementace.....	34
1.8 Datové typy a jejich dědění	35
Vlastní instance třídy a mateřská třída objektu	36
LSP – Liskov Substitution Principle	36
Přetěžování × přebíjení × zakrývání metod	37
1.9 Odkazové a hodnotové datové typy	38
1.10 Návrhové vzory	38
1.11 Modul × komponenta × knihovna × framework	40
Modul	40
Komponenta	40
Knihovna.....	41
Framework.....	41
1.12 Změny šablon	42
1.13 Knihovna CanvasManager	44
1.14 Shrnutí – co jsme se naučili	44
2. Vývojové prostředí NetBeans	46
2.1 Instalace	47
Instalace pro Windows.....	48
2.2 První spuštění.....	50

2.3	Aplikační okno, panely a karty	51
	Změny rozměrů panelů	53
	Minimalizace a obnovení panelů a karet	53
	Další možnosti	53
2.4	Otevření existujícího projektu	53
2.5	Navigátor a jeho ikony	57
2.6	Úprava nastavení prostředí	57
2.7	General – obecná nastavení	58
2.8	Editor – nastavení editoru	59
	Karta General	59
	Braces Matching	59
	Camel Case Behavior	59
	Search	59
	Karta Folding	60
	Karta Formatting	60
	Karta Code Completion	61
	Language	61
	Karta Code Templates	62
	Karta Hints	63
	Karta Highlighting	63
	Karta Macros	63
	Karta OnSave	64
	Karta Spellchecker	64
2.9	Fonts & Colors – nastavení písma a barev	64
	Nastavení písma	64
	Vybarvení komentářů	65
2.10	Keymap – klávesové zkratky	67
2.11	Java – nastavení pro Javu	67
2.12	Team	67
	Karta Action Items	67
2.13	Appearance – nastavení vzhledu	67
	Karta Document Tabs	68
	Karta Windows	69
	Karta Look and Feel	69
2.14	Miscellaneous – zbylá nastavení	69
	Karta CSS Preprocessors	70
	Karta Diff	70
	Karta Files	70
	Karta Output	70
	Karta Terminal	71
2.15	Nastavení panelů nástrojů	71
2.16	Export a import nastavení	72
	Export	73
	Import	73
2.17	Shrnutí – co jsme se naučili	74
3.	Projekty v NetBeans – Library	76
3.1	Balíčky programů tohoto dílu	76
3.2	Balíčková struktura knihovny	77
3.3	Složky se zdrojovými soubory	77
3.4	Balíčky na kartě projektů	80
3.5	Práce s balíčky	81
	Vytvoření nového balíčku	81
	Přesun tříd mezi balíčky	82
	Importy z vlastního balíčku	85

	Přejmenování balíčku	85
3.6	Překlad a sestavení projektu	87
3.7	Programátorská dokumentace (API)	87
	Dokumentace při psaní kódu.....	87
	Možnosti okna dokumentace.....	88
	Samostatná karta dokumentace.....	89
	Vytvoření dokumentace projektu	89
3.8	Karta souborů	90
3.9	Vlastnosti projektu	91
	Stránka Sources	91
	Stránka Libraries	92
	Stránka Compiling	92
	Stránka Documenting	92
	Stránka Run	93
3.10	Přejmenování projektu.....	93
3.11	Definice projektu jako knihovny	94
3.12	Shrnutí – co jsme se naučili	96
4.	Vytváříme nový projekt – AHA	99
4.1	Vytvoření nového projektu	99
4.2	Spuštění aplikace	103
4.3	Vytvoření kopie třídy	103
4.4	Nápověda při psaní kódu.....	104
4.5	Zadání spouštěcí třídy projektu	105
4.6	Spouštěcí konfigurace	106
4.7	Vytvoření a spuštění aplikace	107
4.8	Paralelní spuštění více aplikací.....	108
4.9	Shrnutí – co jsme se naučili	109
5.	Práce na připraveném projektu – Elevator	111
5.1	Poloprázdná třída a metoda	111
5.2	Zadání	112
5.3	Analýza problému.....	113
	Sjednocení různých řešení	113
	Implementované interfejsy	114
	Okolí	114
	Konstruktory	115
	Dva přístupy k řešení problému	116
	Potřebné metody.....	117
5.4	Interfejs IElevator	118
5.5	Vzorový projekt.....	120
5.6	Testovací třída	121
	Přizpůsobující se společná testovací třída	121
	Inicializace a finalizace bloku testů v dané třídě	122
	Třídy jako objekty – class-objekt třídy.....	123
	Zafixování testované třídy.....	123
	Vynechání konkrétního testu.....	124
	Spuštění a vyhodnocení testů	125
5.7	Definice vlastní třídy	126
	Atributy	126
	Konstruktory a metody interfejsů IPaintable a IMovable	127
	Interfejs a data.....	128
	Postup při návrhu metod deklarovaných v interfejsu IElevator	129
	Metoda goTo(int) – předehra.....	129
	Metody floor2y(int) a y2floor(int)	129

	Metoda <code>goTo(int)</code> – realizace	130
	Metoda <code>comeTo(IMovable)</code>	130
	Metoda <code>enter(IMovable)</code>	130
	Metody <code>exitLeft()</code> a <code>exitRight()</code>	131
	Test převozu pasažéra	131
	Metody <code>transportRight(IMovable, int)</code> a <code>transportLeft(IMovable, int)</code>	132
5.8	Porovnání řešení	132
5.9	Práce s více soubory	135
5.10	Shrnutí – co jsme se naučili	136
6.	Spolupráce projektů – <code>Vehicle</code>	138
6.1	Zadání	138
6.2	Vytvoření nové třídy	139
	Zakomentování a odkomentování části kódu	141
6.3	Dokumentace balíčku	141
6.4	Použití frameworku či knihovny	143
	Třída <code>IO</code> jako aplikace návrhového vzoru <i>Fasáda</i>	144
	Zkopírování zdrojových souborů	144
	Podporované způsoby připojení potřebného projektu	144
	Připojení celého potřebného projektu	145
	Přidání JAR-souboru mezi knihovny	146
	Přidání propojení na knihovnu	148
6.5	Oprava špatného nastavení	149
	Nevytvořený JAR-soubor	150
	Přestěhování souborů na jiné místo disku	150
6.6	Poloautomatická implementace interfejsu	150
6.7	Konstruktory	151
	Poloautomatická generace konstruktoru	151
	Poloautomatické doplnění komentářových značek	152
	Doplnění těl konstruktorů	153
	Dokončení těla konstruktoru	153
	Výraz → lokální proměnná	155
	Lokální proměnná → atribut	156
6.8	Rychlý test	157
6.9	Historie změn	158
	Podrobnosti o barvách	159
6.10	Shrnutí – co jsme se naučili	161
7.	Testovací třída – <code>VehicleTest</code>, <code>Robot</code>	162
7.1	Vytvoření	162
7.2	Obsah testovací třídy	165
	Inicializace a finalizace	165
	Těla poloprázdných metod	165
7.3	Šablona testovací třídy	166
	Parametry anotace <code>@Test</code>	167
7.4	Využití služeb třídy <code>IndentingReporter</code>	167
	Popis některých metod	168
	Použití odsazení	173
	Získání názvu spouštěného testu	173
7.5	Spuštění testů	175
	Spuštění konkrétního testu	175
	Spuštění všech testů dané třídy	175
	Spuštění všech testů projektu	176
7.6	Definice inicializačních a finalizačních metod	176
	Získání správce plátna	177
	Lokální proměnná → statická konstanta	177

7.7	Nechtěné automatické doplnění identifikátoru.....	180
7.8	Vytvoření požadovaných testů.....	181
	Test funkce přípravku.....	181
	Test implementovaných metod.....	181
7.9	Definice přístupových metod testované třídy.....	183
7.10	Lokalizace souboru v projektu.....	184
7.11	Přejmenování třídy spolu s testem.....	184
7.12	Přesun do nového balíčku.....	185
7.13	Vyhledávání a nahrazování textu.....	186
7.14	Shrnutí – co jsme se naučili.....	188
8.	Ladění programů – Robot	190
8.1	Metody ladění.....	190
	Kontrolní tisky.....	191
	Používání ladícího programu.....	191
8.2	Nastavení zarážky v řádku kódu.....	191
8.3	Možnosti krokování.....	192
8.4	Zobrazování dat.....	193
8.5	Zásobník volání.....	195
8.6	Zarážka na entitě.....	195
	Trvalost zářezek.....	197
8.7	Záložky (bookmark).....	198
8.8	Úkoly.....	198
8.9	Shrnutí – co jsme se naučili.....	198

Část II: Vylepšování architektury 201

9.	Program ve výjimečné situaci.....	202
9.1	Co to jsou výjimky.....	202
9.2	Nejdůležitější výjimky.....	203
9.3	Vyhození výjimky.....	204
9.4	Výjimky a nedosažitelný kód.....	206
9.5	Co výjimky umí.....	206
	getMessage().....	206
	toString().....	207
	printStackTrace().....	207
	printStackTrace(PrintStream).....	207
9.6	Hierarchie dědění výjimek.....	208
9.7	Zachycení vyhozené výjimky.....	209
	Analýza rekurzivní metody.....	211
9.8	Několik současně odchyťovaných výjimek.....	212
	Společná reakce na několik výjimek.....	213
9.9	Společný úklid – blok finally	214
9.10	Testování správného vyhození výjimky.....	216
	Tělo metody testující správné vyhození výjimky.....	216
	Specifikace očekávané výjimky v anotaci.....	217
9.11	Definice vlastních výjimek.....	218
9.12	Kontrolované výjimky.....	218
9.13	Převedení kontrolované výjimky na nekontrolovanou.....	220
9.14	Informace o skutečném původci výjimky.....	221
9.15	Ověřování podmínek – příkaz assert	222
	Design by Contract.....	223
9.16	Shrnutí – co jsme se naučili.....	225

10. Návrhový vzor <i>Tovární metoda</i>	228
10.1 Motivace	228
10.2 Jak na to	230
10.3 Použití v projektu s výtahy	231
10.4 Programování proti rozhraní	234
10.5 Použití tovární třídy v projektu s vozidly	235
Definice interfejsu IVehicle	235
Testovací třída VehicleTest	237
10.6 Možnost výběru testované třídy	238
Přepínání mezi pevně zadanou a volitelnou tovární třídou	239
10.7 Možnost využití konstruktoru třídy	240
10.8 Shrnutí – co jsme se naučili	241
11. Návrhový vzor <i>Stav – Robot4</i>	243
11.1 Řešený problém	243
11.2 Vozidla na šachovnici	244
11.3 Společné rozhraní otočných vozidel IVehicle	244
11.4 Různé chování v závislosti na směru	245
11.5 Jednostavové třídy	245
11.6 Čtyřstavová třída	246
11.7 Stavové rozhraní	246
11.8 Definice jednostavových tříd	247
11.9 Definice vícestavové třídy	251
11.10 Testovací třída	255
11.11 Zásady použití vzoru <i>Stav</i>	257
11.12 Shrnutí – co jsme se naučili	258
12. Návrhový vzor <i>Stavitel – RingBuilder</i>	260
12.1 Řešený problém	260
12.2 Dvě skupiny požadavků na segment	261
12.3 Definice segmentů	263
Nastavení barvy	264
Konstruktory	265
Test správného vytvoření segmentů	266
Přidání následníka	267
Potřebné atributy	267
Zbylé metody	268
12.4 Zdánlivý problém s viditelností segmentů	268
12.5 Definice dopravního okruhu	269
Správa vytvořeného okruhu	270
Zobrazení okruhu	270
Přizpůsobení se změně kroku plátna	270
Oznámení startovního segmentu	270
Konstrukce okruhu	270
12.6 Návrhový vzor <i>Stavitel</i>	271
12.7 Definice stavitele – RingBuilder	272
Atributy	272
Konstruktor	273
Start stavby okruhu	273
Zřetězení volání metod	274
Pokračování ve stavbě okruhu	274
Ukončení stavby okruhu	274
Test stavby okruhů	275
12.8 Ověřování podmínek	276

12.9	Test vyhazování výjimky.....	276
12.10	Dokončení definice okruhu	278
	Nastavení políčkové pozice	278
	Prozrazení políčkového rozměru	278
	Přizpůsobení se změně kroku plátna.....	279
12.11	Extrakce části kódu do samostatné metody.....	279
12.12	Test vybudovaného okruhu.....	281
12.13	Továrna na okruhy	282
12.14	Shrnutí – co jsme se naučili	283
13.	Návrhový vzor <i>Dekorátor</i> – <i>SmoothVehicle</i>	284
13.1	Modifikace chování skupiny objektů	284
13.2	Plynule posuvná vozidla.....	285
13.3	Definice dekorující třídy.....	285
	Delegát a konstruktory	290
	Implementace metod pro porovnání objektů	290
	Implementace zbylých metod	291
	Ještě trochu kosmetiky	291
13.4	Definice těla metody <i>goForward()</i>	292
13.5	Doplnění metody delegující zodpovědnost na atribut	293
13.6	Přidání vlastnosti.....	294
13.7	Dokončení úprav	296
13.8	Test	296
13.9	Princip vzoru <i>Dekorátor</i>	297
13.10	Shrnutí – co jsme se naučili	298
14.	Implicitní implementace – <i>RingVehicle</i>, <i>ControlledVehicle</i>	300
14.1	Dekorátor přidávající další funkčnost	300
14.2	Třída <i>MultiMover</i> a interfejs <i>IMultiMovable</i>	301
14.3	Definice třídy <i>RingVehicle</i>	301
14.4	Implicitní definice metod interfejsu	302
14.5	Statické metody definované v interfejsu	305
14.6	Šablona interfejsů	306
14.7	Čím se liší interfejs od třídy.....	306
14.8	Výhody implicitní implementace	307
14.9	Úprava interfejsu <i>IVehicle</i>	307
14.10	Doplnění konstruktorů továrních objektů	308
14.11	Rozšíření interfejsu <i>IVehicleFactory</i>	309
	Test.....	311
14.12	Pokračování definice přesunu	312
14.13	Vypuštění vozidla na okruh	312
14.14	Test	313
14.15	Vozidlo ovládané z klávesnice	314
14.16	Návrhový vzor <i>Adaptér</i> (<i>Adapter</i>)	315
14.17	Návrh třídy <i>ControlledVehicle</i>	315
14.18	Přebití implicitních definic.....	316
14.19	Testování.....	316
	Mechanismus reakce na klávesnici	317
14.20	Shrnutí – co jsme se naučili	318
15.	Generické datové typy a metody	320
15.1	Motivace	320
15.2	Generické a parametrizované datové typy.....	324

15.3	Definice generických typů	326
15.4	Použití generických typů	328
15.5	Rizika nepoužití typových parametrů	330
15.6	Varování překladače a jejich potlačení.....	333
	Proč vypínat varování	334
15.7	Překlad generických datových typů a očišťování.....	335
15.8	Omezení typových atributů na instanční členy	336
15.9	Generické metody	336
15.10	Shrnutí – co jsme se naučili	340
16.	Pokročilejší práce s typovými parametry	342
16.1	Omezení typových parametrů	342
16.2	Typové parametry s více předky.....	343
16.3	Potomci a předci generických typů	344
16.4	Žolíky	344
16.5	Příklad: datový typ <code>Interval<T extends Comparable<? super T>></code>	346
16.6	Ternární operátor <code>?:</code> – podmíněný výraz	351
16.7	Definice parametrizovaného datového typu	352
	Grupy	353
	Deklarace <code>IGroup<B, G extends IGroup<B, G>></code>	354
	Definice třídy <code>DirectionGroup</code>	354
	Na co potřebujeme interfejs <code>IGroup</code>	356
16.8	Shrnutí – co jsme se naučili	356
17.	Funkční interfejsy a lambda-výrazy	358
17.1	Motivace	358
17.2	Funkční interfejs (functional interface)	359
17.3	Lambda-výrazy	362
17.4	Použití lambda výrazů v programu	363
17.5	Předčasné zhasínání	365
	Metoda <code>stopBlinking()</code>	365
	Modifikátor <code>volatile</code> a synchronizace vláken	366
	Test ukončení neexistujícího blikání – <code>testWrongStopBlinking()</code>	367
	Reakce na ukončení blikání.....	367
	Test správné reakce na předčasné spuštění.....	368
	Test korektního ukončení blikání – <code>testStoppedMovingAndBlinking()</code>	369
17.6	Alternativní definice funkčních objektů	370
17.7	Světlo umožňující ovlivnit tvar žárovky	372
	Získání žárovky	373
	Požadavky na typ žárovky	373
	Uložení žárovky	374
	Uložení továrního objektu.....	375
	Upravená definice třídy <code>Light</code>	375
	Testy	375
17.8	Generická verze třídy – třída <code>LightG <B extends IChangeable & IColorable></code>	379
	Důsledky definice třídy <code>LightG</code> jako generické	380
17.9	Sjednocení definic otoček robota	381
17.10	Shrnutí – co jsme se naučili	383
18.	Rekurzivní volání	386
18.1	Princip	386
18.2	Přímá a nepřímá rekurze.....	387
18.3	Přeplnění zásobníku návratových adres.....	388
18.4	Pojezdy tam zpět – metoda <code>zigZag</code>	388

1. Úkol	389
2. Otočka	389
3. Délka pojezdu	389
4. Cílová pozice	390
5. Předání metody multipřesouvači	390
Odbočka: rekurze versus zpětné volání	392
Test správného naprogramování přesunu	392
18.5 Objížďení čtverce	394
18.6 Shrnutí – co jsme se naučili	395
19. Interní datové typy	397
19.1 Motivace	397
19.2 Terminologie	398
19.3 Společné charakteristiky interních typů	399
19.4 Použití	400
Pomocný soukromý typ	401
Objekt znající útroby a implementující veřejné rozhraní	401
Sdružení souvisejících typů	402
19.5 Globální interní (členské) datové typy	402
19.6 Vnořené datové typy	403
19.7 Pomocná vnořená přepravka	403
Řešený problém	403
První nástřel: poloveřejná přepravka	404
Test	406
Co je na předchozím řešení nešikovné	406
19.8 Vnořená tovární třída	409
Výhody a nevýhody jednotlivých možností	409
19.9 Vnitřní třídy	411
Blikající světlo s vnitřní třídou	412
Hraniční obdélník objektu na plátně	415
19.10 Lokální třídy	419
Pojmenované lokální třídy	420
Anonymní třídy	420
Blikající světlo s anonymní třídou	420
Použití anonymních tříd	422
19.11 Shrnutí – co jsme se naučili	422
20. Kontejnery a datovody	424
20.1 Kontejnery	424
Zvláštnosti programových kontejnerů	425
Přepravy	425
Pole (array)	425
Kolekce (collection)	427
Mapy, slovníky (map, dictionary)	427
20.2 Motivace pro zavedení datovodů	428
Deklarativní a imperativní styl programování	429
20.3 Datovody (streams)	430
Druhy operací	431
Práce datovodu	432
20.4 Vytváření datovodů z kolekcí a polí	432
20.5 Použití datovodu – blikající světla	433
Třída <code>StreamTest</code>	434
Pomocná metoda <code>streamBlink(Stream<Light>,String)</code>	435
20.6 Porovnání sériového a paralelního datovodu	437
20.7 Použití metody <code>forEach(Runnable)</code>	437
20.8 Použití filtrů	438

20.9	Řazení objektů v datovodu	439
20.10	Složitější příklad	440
0.	Zadání	440
1.	Rozbor	441
2.	Test – metoda <code>testMovementsStepObj()</code>	441
3.	Vytvoření a zpracování proudu kroků – metoda <code>movementsStepObj(String, Collection<? extends IChangeable>)</code>	443
4.	Přesun objektů v daném kroku – metoda <code>moveInStepAllObjects(String, Collection<? extends IChangeable>)</code>	444
	Definice metod „plynulé“ verze	445
20.11	Konverze prvků v datovodu	446
	Metoda <code>createAndDrive(IVehicleFactory, String, Position...)</code>	446
	Pomocná metoda <code>goInDirections(String)</code>	448
	Test	448
20.12	Vytvoření vlastního datovodu	449
20.13	Shrnutí – co jsme se naučili	450

Část III: Dědění implementace 455

21.	Podrobnosti o konstruktorech tříd a instancí	456
21.1	Opakování: co víme o konstruktorech instancí	456
21.2	Zavádění třídy – <code>java.lang.ClassLoader</code>	457
21.3	Statický konstruktor – konstruktor třídy	458
21.4	Instanční inicializační blok	458
21.5	Dvojitost těla konstrukturu instancí	459
21.6	Příklad	459
21.7	Statický konstruktor, konstruktor třídy	465
	Důležitá pravidla	465
	8 – 14: Úvodní statický inicializační blok	465
	29: Předčasné použití atributu	465
	13: Nekorektní použití metod	466
	46: Předčasné použití konstanty	466
	66: Nekorektní volání konstrukturu	466
	Doporučení: jediný statický inicializační blok	467
21.8	Konstruktor instancí	467
	„Roztroušená“ část	467
	17 – 20: Úvodní instanční inicializační blok	467
	146: Deklarace konstanty <code>loaded</code>	468
	150 – 154: Inicializační výpočet	468
	160: Použití <code>this</code>	468
	250 – 253: Závěrečný blok	468
	Tělo osloveného konstrukturu	468
	170 – 175: Bezparametrický konstruktor	469
	182 – 188: Jednparametrický konstruktor	469
	196 – 201: Dvoupametrický konstruktor	469
	209 – 222: Tříparametrický konstruktor	469
21.9	Experimenty	470
21.10	Shrnutí – co jsme se naučili	470
22.	Úvod do dědění implementace: <code>Mother – Daughter – Granddaughter</code>	473
22.1	Úvodní poznámky	474
22.2	Definice dceřiné třídy	474
22.3	Rodičovský podobjekt	476
22.4	Konstruktor	477

	Konstrukce rodičovského podobjektu	477
22.5	Přetížené verze konstruktorů – použití <code>super</code> × <code>this</code>	478
	Test.....	481
22.6	Konstruktory rodiče a potomka	481
22.7	Emulace dědění dekorátorem	482
	Přípony názvů typů v přípravku	484
22.8	Demonstrace chování konstruktorů	484
	Konstrukce podpisu	487
	Zpráva o zavedení třídy	488
	Demonstrace	489
	Rodičovský podobjekt je abstrakce.....	490
22.9	Vytváření instancí tříd využívajících dekorátor	491
22.10	Chráněné členy – modifikátor přístupu <code>protected</code>	493
22.11	Zákaz vytváření potomků třídy	495
22.12	Shrnutí – co jsme se naučili	496
23.	Zakrývání atributů a metod	498
23.1	Posílání zpráv a volání metod	498
23.2	Dědění metod.....	499
	Zděděné, dále neupravované metody.....	499
	Zděděné metody, pro něž potomek definuje „lepší“ implementaci.....	500
	Kompatibilita signatur	500
23.3	Zakrývání metod předka (method hiding)	501
23.4	Metody, které není možno v potomku zakrýt či přebít – modifikátor <code>final</code>	504
23.5	Třídy, které nemohou mít potomky.....	506
23.6	Zakrývání atributů předka.....	506
	Emulace zakrývání v D-třídách.....	508
23.7	Metody nově definované v potomku	508
	Statically × dynamicky typované jazyky	509
	Proč je situace jednoduchá jen zdánlivě.....	510
23.8	Zakrývání interních datových typů	510
23.9	Závěr	513
23.10	Shrnutí – co jsme se naučili	514
24.	Virtuální metody a jejich přebíjení.....	515
24.1	Virtuální metody a jejich přebíjení	515
	Časná a pozdní vazba.....	516
	Virtuální metody.....	516
24.2	Které metody jsou v Javě virtuální	517
24.3	Chování virtuálních metod	517
24.4	Emulace virtuálních metod v dekorátoru	521
24.5	Zdokonalení třídy <code>Square</code>	521
	Přebíjení metody <code>copy()</code>	522
	Problémy s nastavováním velikosti	522
	První návrh definice metody <code>setSize(int, int)</code>	523
	Test prvního návrhu	524
	Oprava.....	526
24.6	Co se nám na dědění nelíbí	527
24.7	Návrhový vzor <i>Šablonová metoda (Template method)</i>	528
	Princip.....	528
	Implicitní metody interfejsů	528
	Metoda <code>toString()</code>	529
24.8	Shrnutí – co jsme se naučili	530
25.	Pasti a propasti dědění implementace.....	532

25.1	Třída <code>XRectangle</code>	532
	Testovací třída	533
	Podklady pro vlastní řešení.....	534
	Definice konstruktorů	534
	Definice tovární třídy	536
	Metoda <code>paint(Painter)</code>	536
	Změny pozice a velikosti	537
	Upravená podoba definice třídy	538
25.2	Co je na uvedeném řešení nevhodné	538
25.3	Řešení definicí atributu	541
25.4	Řešení sloučením dědění a dekorátoru.....	541
	Typové parametry.....	542
	Předci.....	546
	Statické členy.....	546
	Instanční členy.....	546
25.5	Samostatná úloha: Terč.....	547
25.6	Virtuální metody v konstruktoru	547
	Definice třídy <code>Aureole</code>	547
	Test objektů se svatozáří.....	548
	Řešení 1: Změna řešení	552
	Řešení 2: Devirtualizace metody	552
	Řešení 3: Využití rodičovské verze metody.....	553
	Řešení 4: Definice ekvivalentní soukromé metody.....	553
25.7	Shrnutí – co jsme se naučili	553
26.	Vytváříme rodičovskou třídu – <code>ARobot1</code>	555
26.1	Abstraktní metody a třídy	555
	Abstraktní a konkrétní metody.....	555
	Interfejsy	556
	Třídy.....	556
	Abstraktní a konkrétní třídy.....	556
	Účel abstraktních tříd.....	557
	Účel abstraktních metod.....	557
26.2	Proč společný rodič	558
26.3	Návrhový vzor <i>Štáv</i> s rodičovskou třídou.....	558
	Vytvoření prázdného společného rodiče	559
	Příprava potomků	559
	Členy třídy	560
	Konstantní atributy instancí	560
	Konstruktory	560
	Metody instancí	562
	Ověření regresním testem	563
26.4	Rodičovská třída segmentů okruhu	564
	Specifikace předků	564
	<code>IRingSegment</code>	564
	<code>IChangeable</code>	564
	<code>copy()</code>	565
	<code>IRingBuildSegment</code>	565
	Společný abstraktní rodič.....	566
	Definice potomků	567
26.5	Návrhový vzor <i>Adaptér</i> podruhé.....	570
26.6	Společný rodič dekorátorů	571
26.7	Použití ve třídě <code>ControlledVehicle</code>	572
26.8	Společný rodič výtahů	572
26.9	Shrnutí – co jsme se naučili	573

Část IV: Další užitečné programové konstrukce	575
27. Učíme program přemýšlet	576
27.1 Jednoduchý podmíněný příkaz	577
27.2 Předčasné ukončení metody	578
27.3 Kdy assert a kdy if	578
27.4 Blok příkazů (složený příkaz)	579
Formátování bloků příkazů	579
Blok je chápán jako jeden příkaz	580
Další vlastnosti bloku příkazů	581
Vnořování bloků příkazů	581
27.5 Metoda equals(Object)	583
Kontrakt metody	583
Definice metody	583
27.6 Metoda hashCode()	585
27.7 Neměnnost objektů	587
27.8 Zanořování podmíněných příkazů	588
Architektura	589
27.9 Výběr ze dvou možností	591
27.10 Kaskáda možností	592
Tovární metoda jednosměrného vozidla	594
27.11 Přepínač – příkaz switch	595
Slučování návěstí	597
27.12 Přepínač nad výčtovým typem	598
Kvalifikace v návěstích	598
27.13 Přepínač nad řetězci	599
27.14 Shrnutí – co jsme se naučili	600
28. Ještě jednu rundu, prosím	603
28.1 Měření času v Javě	603
28.2 Cykly	604
28.3 Jak máme rychlý počítač – cyklus s koncovou podmínkou	604
28.4 Jeden test nestačí – cyklus s počáteční podmínkou	606
28.5 Cyklus s parametrem	607
28.6 Nekonečný cyklus	609
28.7 Vnořování cyklů	610
28.8 Cyklus s podmínkou uprostřed	611
28.9 Příkaz break s návěstím	612
28.10 Cyklus s prázdným tělem	614
28.11 Dvojtečkový cyklus for	615
28.12 Shrnutí – co jsme se naučili	617
29. Další důležité datové struktury	619
29.1 Pracujeme s náhodou	619
29.2 Kontejnery	621
Statické a dynamické kontejnery	621
Kolekce (Collection)	623
Množina (Set)	623
Seznam (List)	624
Fronta (Queue)	624
Oboustranná fronta (Deque)	625
Zásobník (Stack)	625
Strom (Tree)	625
Graf (Graph)	626

Mapa (Map), Slovník (Dictionary)	626
29.3 Standardní knihovna kolekcí Javy	627
29.4 Převod datovodu na kolekci či pole	628
29.5 Návrhový vzor lterátor (lterator).....	628
Princip.....	629
Interfejsy <code>java.util.Iterator<E></code> a <code>java.lang.Iterable<E></code>	629
Použití iterátorů v Javě	630
Odebírání objektů během cyklu	631
Zobecnění možnosti cyklu <code>for(:)</code>	633
29.6 Shrnutí – co jsme se naučili	635
30. O čem jsme ještě nehovořili	638
30.1 Užitečné třídy ze standardní knihovny	638
30.2 Výčtové datové typy	639
30.3 Datový typ <code>Optional<T></code>	639
30.4 Regulární výrazy (regular expressions)	640
30.5 Seznam doporučené a nedoporučené literatury	641
30.6 Slovo na závěr	641
Rejstřík.....	642

Skrytí spoluautoři

Při tvorbě takto rozsáhlé knihy se autor nemůže spolehnout pouze sám na sebe. Je známou věcí, že když po sobě autor čte svůj rukopis, čte velmi často to, co chtěl napsat, a ne to, co doopravdy napsal. S kontrolou jazykových chyb mu může pomoci redaktor a jazykový korektor. S kontrolou odborných zaškokbrtnutí mu však musí pomoci někdo jiný.

S kontrolou odborné správnosti musí pomoci někdo, kdo je odborně na výši a odhalí, že se autor vyjadřuje nepřesně nebo dokonce chybně. S kontrolou srozumitelnosti výkladu musí zase pomoci někdo, kdo se chce látku naučit a je ochoten text podrobně pročíst a upozorňovat na místa, kde výkladu zcela nerozumí nebo se mu zdá, že by bylo vhodné ještě něco doplnit. Úplně dokonalé pak je, když váš text čte zkušený učitel, který umí odhalit nejenom oba výše popsané druhy nepřesností či dokonce chyb, ale také naznačit možná potenciální nedorozumění a dezinterpretace vzniklá z podobnosti vykládané látky s jinými oblastmi, o nichž se výklad nezmiňuje.

Měl jsem to štěstí, že se mi podařilo získat spolupracovníky (a tím i kritiky) ze všech tří zmiňovaných skupin. Všem bych jim chtěl tímto poděkovat. Řekl bych, že díky jejich připomínkám a doporučením je výsledný text kvalitnější.

Všechny tyto dobrovolné spolupracovníky vnímám jako spoluautory. Proto mi dovolu, abych je tu alespoň abecedně vyjmenoval – byli to: Milan Augustin, Tibor Bako, Jakub Hadam, Michael Charvát, Jiří Kofránek, Jiří Kubala, Petr Kuchař, David Král, Filip Malý, Nikolas Patrik, Jarmila Pavlíčková, Luboš Pavlíček, Josef Svoboda, Petr Vidrman, Martin Vondráček a Martin Žamberský.

Úvod

Otevíráte knížku, která vás chce naučit programovat moderním, objektově orientovaným stylem. Stylem, jímž se v dnešní době vyvíjí drtivá většina klíčových aplikací, ale k jehož výuce ještě řada škol nedospěla. Po nastudování této knížky budou proto mnozí z vás vědět o moderním programování víc než leckterý z vašich učitelů.

Komu je kniha určena

Kniha je určena programátorům, kteří potřebují získat hlubší vhled do problematiky objektově orientovaného programování a návrhu architektury objektově orientovaných programů. K jejímu studiu přitom nepotřebují žádné velké předběžné zkušenosti. Stačí znalosti na úrovni mých začátečnických učebnic.

Kniha je reakcí na nepřetržité nářky vedoucích programátorských týmů, kteří posílají nastoupivší absolventy do mých přeškolovacích kurzů. Stěžují si, že školy opouštějí možná skvělí kodéři, ale neschopní architekti, že tito absolventi znají několik programovacích jazyků a řadu užitečných frameworků, ale mají problém s návrhem kvalitní architektury zadávaného programu.

Většina škol, učebnic a výukových kurzů se soustředí především na výklad syntaxe probíraného programovacího jazyka a probrání hlavních knihoven používané platformy. Jejich autoři se soustřeďují na detaily v bláhové naději, že jejich studenti a čtenáři pak v průběhu následující praxe pochopí vyšší principy. Když se ve výukovém programu objeví kurzy návrhových vzorů, končí často jako defilé technik, jak to či ono šikovně zakódovat.

Koncepce knihy

Ve svých knihách se pokouším tento stereotyp změnit. Soustředím se především na předvádění toho, jak program navrhnout a probrat věci, které považuji za důležité a přitom je v jiných učebnicích většinou nenajdete. Přitom se pokouším vše demonstrovat na příkladech. Na rozdíl od některých učebnic se však nechci omezovat pouze na AHA-příklady, jejichž jediným cílem je, aby po jejich prostudování student prohlásil „Aha! Takto to funguje!“, i když se jim v některých chvílích nevyhnu. Snažím se ale co největší část látky demonstrovat na složitějších příkladech, které však nejsou zahlceny spoustou šumu, za který považuji kód, který se

musí naprogramovat, i když s vykládanou látkou přímo nesouvisí. Dávám přitom přednost příkladům používajícím grafiku, protože jsou pro daný účel většinou nejnázornější.

Kniha je koncipována jako druhý díl knihy *Java 7 – Učebnice objektové architektury pro začátečníky*¹. Předpokládá proto, že znáte látku zhruba na úrovni prvního dílu. Protože ale očekávám, že knihu bude číst i řada čtenářů, kteří získali své první programátorské zkušenosti z jiných zdrojů, tak pro jistotu na počátku nejdůležitější věci velice stručně zopakují.

Co se naučíte, uspořádání knihy

Snažil jsem se v knize soustředit na oblasti, které se do jiných knih nevešly, a přitom je jejich osvojení velmi důležité, protože jejich neznalost je v lepším případě příčinou těžkopádných řešení a v horším případě příčinou špatně odhalitelných chyb.

V první části nejprve bleskově zopakují základní látku prvního dílu a pak vás seznámím s jedním ze tří nejpoužívanějších profesionálních vývojových prostředí: s prostředím *NetBeans*. V minulém dílu jsme pracovali s vývojovým prostředím *BlueJ*. To je sice asi nejlepší prostředí pro získání základních představ a návyků objektově orientovaného programování, ale pro profesionální práci se příliš nehodí. Protože je ale zvládnutí vývojového prostředí zhruba stejně náročné jako zvládnutí programovacího jazyka, je mu věnován celý zbytek první části. Naučíte se je konfigurovat, vytvářet v něm projekty a využívat jeho výhodných vlastností pro zefektivnění své práce.

V druhé části si postupně ukážeme různé programové konstrukce a postupy, které můžete využít při návrhu architektury projektu. Postupně se seznámíte s dalšími návrhovými vzory, osvojíte si definici interfejsů obsahujících definice metod včetně jejich implementace, dozvíte se o možnosti parametrizace datových typů a jejich vzájemném vnořování, naučíte se používat lambda-výrazy, které umožňují pracovat s částmi kódu jako s proměnnými, a získáte první zkušenosti s prací s kontejnery a datovody.

Ve třetí části se ponoříme do výkladu dědění implementace. Začneme podrobným výkladem zavádění objektů a tříd a budeme pokračovat základními vlastnostmi dědění implementace. Dozvíte se, jak vytvářet třídy, jejichž předky jsou jiné třídy, a hlavně na co si dát při konstrukci takovýchto tříd pozor. Pokusím

¹ Kniha je současně slibovaným druhým dílem starší učebnice *OOP – Naučte se myslet a programovat objektově*. Ta sice vyšla v jiném nakladatelství, ale protože dané nakladatelství nedokázalo před několikaletí upomínky dostát svým závazkům, tak jsem z tvorby druhého dílu vycouval.

se vám demonstrovat některé vlastnosti dědění na třídách, které dědění emulují využitím návrhového vzoru *Dekorátor* a předvedu vám, jak definovat společného rodiče skupině tříd, které mají některé společné vlastnosti.

Ve čtvrté, závěrečné části, vám představím algoritmické konstrukce, jejichž používání je sice postupně nahrazováno jinými technikami (ty probírají předchozí části učebnice), ale prozatím se bez nich stále ještě neobejdeme. Navíc se ve starších programech s některými novějšími konstrukcemi ani nesetkáte.

Programovací jazyk

Přestože má kniha ve svém názvu uveden jazyk Java, tak musím dopředu oznámit, že se nejedná o učebnici jazyka Java. Těch je k dispozici více než dost. Jedná se o učebnici objektově orientovaného programování a jazyk Java je zde používán jako jazyk, v němž jsou zapsány programy demonstrující probíranou látku. Tento jazyk jsem zvolil z několika důvodů:

- ☞ Je to stále s odstupem nejpoužívanější objektově orientovaný jazyk. Programátoři v Javě jsou stále nejžádanější (a mají i jedny z nejvyšších platů).
- ☞ Je to jazyk poskytující všechny klíčové konstrukce používané v moderním programování.
- ☞ Vytvořené programy nejsou omezeny na jediný operační systém, ale můžete je přenášet mezi různými operačními systémy.
- ☞ A vlastnost neocenitelná pro studium a začátečnické experimenty: nástroje pro vývoj všech druhů aplikací od čipových karet až po rozsáhlé aplikace běžící na několika počítačích můžete sehnat zdarma.

Potřebné vybavení

Pro úspěšné studium této knihy budete potřebovat:

- ☞ chuť naučit se objektově programovat a výdrž v situacích, kdy se vám nebude zcela dařit,
- ☞ rozumně výkonný počítač,
- ☞ základní vývojovou sadu Javy (JDK) ve verzi 8 a vyšší, kterou si můžete stáhnout se stránek <http://www.oracle.com/technetwork/java/javase/downloads>.
- ☞ vývojové prostředí *NetBeans* ve verzi 8 a vyšší, které si můžete stáhnout ze stránek <https://netbeans.org>.
- ☞ doprovodné příklady, které si můžete stáhnout ze stránky knihy http://knihy.pecinovsky.cz/uoa2_j8.

Doprovodné projekty

Na stránce knihy na adrese http://knihy.pecinovsky.cz/uoa2_j8 najdete také soubor s generátorem projektů použitých v knize. Téměř každá kapitola má svůj doprovodný projekt. Některé kapitoly jich mají dokonce několik. Možná bude někomu připadat zbytečné vytvářet takové množství projektů, když jsou jednotlivé sekce umísťovány do samostatných balíčků a řada projektů ve svém obsahu vychází z projektů předchozích, k nimž buď něco přidá a/nebo něco jiného drobně upraví. Důvodem tohoto uspořádání je snaha vyjít začátečníkům vstříc a umožnit jim v každé kapitole začít znovu s funkčním projektem nezávisle na tom, do jakého stavu přivedli projekt z kapitoly přechozí.

První díl byl určen pro naprosté začátečníky, kterým jsem se snažil vyjít vstříc, a protože vím, jaký problém s angličtinou má značná část studentů, používal jsem pro větší názornost české identifikátory. Pořízením této učebnice jste se přihlásili mezi pokročilejší programátory, a ti se bez znalosti angličtiny neobejdou. V této branži platí: *programátor musí umět anglicky, anebo musí změnit zaměstnání*. V tomto dílu jsou proto už všechny identifikátory anglicky. České zůstávají pouze dokumentační komentáře.

Doplňková literatura

Knihy je tlustá, ale ani tak se do ní nevešlo vše, co by bylo potřeba. Navíc se mi nechtělo podrobně rozebírat témata, která už jsem vysvětlil v publikaci, kterou si můžete zdarma stáhnout. Hovořím konkrétně o knize *Java 5.0 – Novinky jazyka a upgrade aplikací*, kterou si můžete zdarma stáhnout na mých webových stránkách na adrese <http://knihy.pecinovsky.cz/java5novinky>.

Kromě toho se na svých stránkách <http://vyuka.pecinovsky.cz> chystám postupně zveřejňovat materiály, které budu připravovat pro studenty a případné polotovary budoucích publikací. Když si uvědomím, že při psaní knih vychází honorář asi tak na 7 korun za hodinu práce, tak se domnívám, že nic neztratím, když budu budoucí texty poskytovat zdarma a ponechám na rozhodnutí čtenářů, zda si jich cení tak, že jsou za ně ochotni něco zaplatit.