

knihovna programátora

- Učebnice pro ty, kteří nechtějí zůstat obyčejnými kodéry, ale chtějí se stát špičkovými architekty
- Postupuje podle metodiky Architecture First
- 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 7

učebnice objektové architektury
pro začátečníky



O autorovi

Rudolf Pecinovský patří ke špičkovým odborníkům na výuku programování. Publikoval již 43 učebnic, které byly přeloženy do pěti jazyků, a nepřeberné množství článků a příspěvků na odborných konferencích. Je autorem metodiky výuky programování *Karel*, navazující metodiky *Baltík* a moderní metodiky výuky objektově orientovaného programování známé pod anglickým názvem *Architecture First*. Učí programování na VŠE a FIT ČVUT. Současně pracuje jako Senior EDU Expert ve firmě ICZ a.s., kde má na starosti doškolování profesionálních programátorů a organizaci kurzů, které si objednávají jiné firmy.



O knize

Tato kniha je třetím vydáním populární učebnice programování, která je na našem trhu zcela ojedinělá. Na rozdíl od ostatních učebnic, které se soustředí na výuku syntaxe jazyka a práce s knihovnami, se tato kniha soustředí především na výklad architektonických konstrukcí. Neučí čtenáře kódovat, ale snaží se jej naučit, jak programy navrhovat. Učí jej, jak má při programování myslet. Reaguje tak na známou skutečnost, že kodérů je hodně, ale dobrých softwarových architektů je proklatě málo (proto také mají několikanásobně vyšší platy).

Kniha je sice primárně určena začátečníkům, ale ohlasy na předchozí vydání ukázaly, že v ní najdou poučení i zkušení programátoři. Většina učebnic a kurzů programování totiž vyvolává falešnou představu, že objektově programovat znamená používat třídy a dědění. Tato kniha je první, která ukazuje, že objektově orientované programování přináší především jiný způsob myšlení. Jak výstižně napsal jeden čtenář: „*Myslel jsem si, že nejsem žádné programátorské ucho. Když jsem ale přečetl vaši učebnici, otevřel jsem oči a hubu. Konečně jsem pochopil věci, které mi ostatní učebnice nedokázaly vysvětlit.*“

Kniha vznikla na základě dlouholetých autorových zkušeností se školením profesionálních programátorů, výukou programování na univerzitě i vedením žákovských programátorských kroužků. Autor v ní uvádí čtenáře krok za krokem do tajů objektově orientovaného programování a ukazuje mu, jak možnosti této moderní technologie co nejlépe využít a kde si dát naopak pozor na její úskalí.

Výklad je postaven na příkladech, které autor spolu s čtenářem postupně řeší a přitom čtenáře učí nejenom základním programátorským návykům a dovednostem, ale předvede mu i nejrůznější užitečné triky, z nichž mnohé nikde jinde vysvětlené nenajdete. Současně upozorňuje na nejčastější začátečnické chyby, které před svými čtenáři ostatní učebnice většinou tají. Navíc probírá i řadu témat (např. návrhové vzory), která jsou většinou probírána až v pokročilých, nebo dokonce nadstavbových kurzech, přestože patří do základní výbavy objektového programátora.

knihovna programátora

RUDOLF PECINOVSKÝ

Java 7

**učebnice objektové architektury
pro začátečníky**

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.

Java 7

učebnice objektové architektury pro začátečníky

Rudolf Pecinovský

Vydala Grada Publishing, a.s. U Průhonu 22, Praha 7
jako svou 4971. publikaci

Odborní lektoři:

doc. ing. Alena Buchalcevóva, Ph.D., doc. ing. Pavel Herout, Ph.D.,
doc. MUDr. Jiří Kofránek, CSc., doc. ing. Vojtěch Merunka,
Ph.D., prof. RNDr. PhDr. Antonín Slabý, CSc., doc. ing. Miroslav Virius, CSc.

Odpovědný redaktor: Martin Vondráček

Návrh vnitřního layoutu: Rudolf Pecinovský

Zlom: Rudolf Pecinovský

Počet stran 496

Vydání 1., 2012

Vytiskla Tiskárna PROTISK, s.r.o., České Budějovice

© Grada Publishing, a.s., 2012

Cover Photo © allphoto.cz

ISBN 978-80-247-3665-5 (tištěná verze)

ISBN 978-80-247-8325-3 (elektronická verze ve formátu PDF)

ISBN 978-80-247-8326-0 (elektronická verze ve formátu EPUB)

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

Stručný obsah

Skrytí spoluautoři	17
Předmluva k prvnímu vydání	18
Úvod	19
Část 1: Interaktivní režim	31
1. Seznamujeme se s nástroji	32
2. Objekty a třídy	54
3. Testovací třída	79
4. Práce s daty	90
5. Výlet do nitra objektů	111
6. Programátorská dokumentace	122
7. Rozhraní × interface	128
8. Pokročilá práce s rozhraním	150
9. Dědění tříd	176
Část 2: Základy práce v textovém režimu	199
10. Vytváříme vlastní třídu	200
11. Přidáváme parametry	221
12. Přidáváme atributy a metody	232
13. Pokročilejší práce s daty	259
14. Komentáře a dokumentace	284
15. Operace a operátory	307
16. Definice testovací třídy	334
17. Ladění programů	351
18. Implementace rozhraní	364
19. Samostatná aplikace – UFO	387
Část 3: Základní programovací techniky	407
20. Refaktorace	408
21. Hodnotové a odkazové objektové typy	431
22. Složitější rozšíření funkčnosti	452
23. Budete si to přát zabalit?	465
Rejstřík	490

Podrobný obsah

Skrytí spoluautoři.....	17
Předmluva k prvnímu vydání	18
Úvod	19
Co je nového ve 3. vydání	19
Komu je kniha určena.....	19
Co se naučíte.....	20
Styl výuky	22
Programovací jazyk	23
Uspořádání	24
Jazyk programů.....	25
Potřebné vybavení	26
Sada JDK (Java Development Kit).....	26
Vývojové prostředí	27
Proč právě <i>BlueJ</i>	27
Doprovodné programy	28
Doprovodné animace.....	28
Použité konvence.....	28
Odbočka	30
Vaše poznámky a připomínky.....	30
Část 1: Interaktivní režim	31
1. Seznamujeme se s nástroji.....	32
1.1 Trocha historie	32
První počítače.....	32
Co je to program	34
Program musí být především spolehlivý	34
1.2 Objektově orientované programování – OOP	35
Vývoj metodik programování.....	35
1.3 Překladače, interprety, platformy	37
Operační systém a platforma.....	37
Programovací jazyky	38
1.4 Java a její zvláštnosti.....	40
Klíčové vlastnosti Javy.....	40
Objektově orientovaná	40
Jednoduchá	41
Multiplatformní.....	41
Java je jazyk i platforma.....	41
Vývojářská sada	42
1.5 Vývojové prostředí <i>BlueJ</i>	42
1.6 Projekty a <i>BlueJ</i>	43
Windows a substituované disky.....	44
Vyhledání a otevření projektu	46

1.7	Diagram tříd	46
	Manipulace s třídami v diagramu	48
1.8	Shrnutí – co jsme se naučili	52
2.	Objekty a třídy	54
2.1	Nejprve trocha teorie	54
	Principy OOP	54
	Objekty	55
	Třídy a jejich instance	55
	Třída jako objekt	56
	Zprávy	57
	Metody	58
2.2	Výchozí projekt	59
	Stereotypy a nestandardní druhy tříd	60
2.3	Třídy a jejich instance	61
	Vytváříme instanci	61
	Pravidla pro tvorbu identifikátorů v jazyku Java	63
	Vytváříme instanci – pokračování	64
	Proměnné a zásobník odkazů	66
	Posíláme instanci zprávu	67
	Vytváříme další instance	68
	Rušení instancí a správa paměti	69
2.4	Restartování virtuálního stroje	70
2.5	Instance versus odkaz	70
2.6	Úvod do návrhových vzorů	73
	Knihovní třída (Utility class)	74
	Statická tovární metoda (Static factory method)	75
	Jedináček (Singleton)	75
	Výčtový typ (Enumerated type)	76
2.7	Shrnutí – co jsme se naučili	76
3.	Testovací třída	79
3.1	Možnost uložení provedených akcí	79
3.2	Vytvoření testovací třídy	80
3.3	Struktura testovací třídy	80
	Testovací přípravek	81
	Vlastní testy	81
3.4	Definujeme testovací přípravek	81
3.5	Definujeme testovací metody	84
	Další testy	86
	Spuštění všech testů	86
3.6	Shrnutí – co jsme se naučili	88
4.	Práce s daty	90
4.1	Zprávy žádající o hodnotu	90
	Datové typy	91
	Primitivní datové typy	92
	Objektové datové typy	93
	Přístupové metody	94
	Vracení hodnot primitivních typů	95
	Vracení hodnot objektových typů	96
4.2	Parametry metod	99
	Vývolání konstruktoru s parametry	100
	Zadávání hodnot typu String	102
	Modifikace testovacího přípravku	103
	Funkce testů s novým přípravkem	104
	Parametry objektových typů	105

	Nastavování hodnot vlastností	105
	Zadávání hodnot objektových typů	105
	Získání doposud nepoužité barvy	105
	Test demonstrující použití objektových parametrů	106
4.3	Metody třídy – statické metody	108
	Smazání plátna	109
4.4	Shrnutí – co jsme se naučili	110
5.	Výlet do nitra objektů	111
5.1	Atributy (datové členy)	111
	Atributy instancí	112
	Atributy třídy – statické atributy	114
	Instance třídy jako její atributy	116
	Přímé zadávání hodnot parametrů objektových typů	116
5.2	Zkrácený zápis zadávaných zpráv	118
5.3	Návrhový vzor <i>Převrátka</i>	118
5.4	Shrnutí – co jsme se naučili	121
6.	Programátorská dokumentace	122
6.1	Dokumentace aktuální třídy	123
6.2	Dokumentace celého projektu	124
6.3	Dokumentace standardní knihovny	125
6.4	Shrnutí – co jsme se naučili	126
7.	Rozhraní × interface	128
7.1	Teoretický úvod	128
	Motivace	129
	Deklarace × definice	130
	Rozhraní × implementace	130
	Atributy × vlastnosti	131
	Signatura × kontrakt	131
	Rozhraní × interface	132
	Interfejs a jeho instance	133
7.2	Použití v programu	133
	Otevíráme nový projekt	134
7.3	Implementace rozhraní tříd	136
	Implementace rozhraní v diagramu tříd	137
	Zrušení implementace	138
	Důsledky implementace rozhraní	138
7.4	Návrhový vzor Služebník	138
7.5	Nový projekt	139
7.6	Přidání mnohotvaru	141
	Import třídy s dosažitelným zdrojovým kódem	141
	Představení třídy Mnohotvar	141
	Název mnohotvaru	142
	Mnohotvar se skládá z kopií	142
	Metody s proměnným počtem parametrů	143
	Přidání testovací třídy dané třídy	144
	Testovací přípravek mnohotvaru	144
	Plynulé přesuny mnohotvaru	147
7.7	Shrnutí – co jsme se naučili	147
8.	Pokročilá práce s rozhraním	150
8.1	Nevýhody aktuálního řešení a možnosti jejich odstranění	150
8.2	Implementace více rozhraní	151
8.3	Kompresor a jím využívaná rozhraní	153
	Rafinovanější změny velikosti tvarů	153

8.4	Návrhový vzor <i>Prázdný objekt (Null Object)</i>	154
8.5	Dědění rozhraní	155
	Trocha teorie o dědění	155
	Aplikace dědění rozhraní na náš projekt	156
	Přidání značkovacího rozhraní <i>IKreslený</i>	157
8.6	Návrhový vzor <i>Prototyp (Prototype)</i>	159
	Demonstrační test	160
	Proč?	161
	Závěr	162
8.7	Test demonstrující nepříjemné chování grafických objektů	162
8.8	Nová koncepce projektu	163
	Návrhový vzor <i>Prostředník (Mediator)</i>	163
	Inverze závislosti	165
	Návrhový vzor <i>Pozorovatel (Observer)</i> , hollywoodsky princip	166
8.9	Nový projekt	167
	Převod testů do nového projektu	168
	Nový přípravek pro třídu <i>MnohotvarTest</i>	169
	Nový přípravek pro třídu <i>Testy</i>	170
	Nový přípravek pro třídu <i>ITvarTest</i>	171
8.10	Ještě jednou k dědění rozhraní	172
8.11	Shrnutí – co jsme se naučili	173
9.	Dědění tříd	176
9.1	Tři druhy dědění	176
	Přirozené (nativní) dědění	177
	Dědění typu	177
	Dědění implementace	178
9.2	Základy dědění tříd	178
	Princip dědění	179
	Univerzální (pra)rodič <i>Object</i>	180
	Instance třídy <i>Object</i> jako parametr či návratová hodnota	181
9.3	Pokusy s děděním	181
	Překrývání metod	183
9.4	Jediný implementační předek	185
9.5	Abstraktní třídy a jejich role v dědicke hierarchii	185
	Experimenty s abstraktní třídou	187
	Účel abstraktních tříd	188
9.6	Návrhový vzor <i>Šablonová metoda</i>	189
9.7	Zavedení abstraktních tříd do projektu	190
9.8	Implementace	195
9.9	Shrnutí – co jsme se naučili	195

Část 2: Základy práce v textovém režimu 199

10.	Vytváříme vlastní třídu	200
10.1	První vlastní třída	200
10.2	Zdrojový kód třídy	201
	Prázdná třída	201
	Bílé znaky a uspořádání programu	203
10.3	Soubory projektu	203
10.4	Odstranění třídy	206
10.5	Implicitní konstruktor	207
10.6	Přejmenování třídy	212
10.7	Ladění	213

	Syntaktické chyby	214
	Běhové chyby	215
	Logické (sémantické) chyby	218
10.8	Shrnutí – co jsme se naučili	218
11.	Přidáváme parametry	221
11.1	Konstruktor s parametry	221
11.2	Použití skrytého parametru <code>this</code>	223
11.3	Přetěžování	227
11.4	Testování	228
	TDD – vývoj řízený testy	228
	Testovací třída	229
	Testovací přípravek	229
11.5	Shrnutí – co jsme se v kapitole naučili	230
12.	Přidáváme atributy a metody	232
12.1	Deklarace atributů	232
	Modifikátory přístupu	234
	Vylepšujeme <code>Strom</code>	234
	Možné důsledky zveřejnění atributů	235
	Modifikátory konstantnosti	236
12.2	Definujeme vlastní metodu	237
	Test vytvořených metod	239
	Reakce na chybu v testu	241
	Nejprve testy, pak program?	242
	Někdy jsou věci složitější	245
	Použití metod vracejících hodnotu	246
12.3	Definice metod vracejících hodnotu	248
	Parametry a návratové hodnoty objektových typů	248
12.4	Přístupové metody	249
	Atributy a vlastnosti našich stromů	250
12.5	Kvalifikace a klíčové slovo <code>this</code>	251
	Příklad	252
	Kvalifikace atributů	254
	Příklad: <code>Světlo</code>	254
12.6	Shrnutí – co jsme se naučili	256
13.	Pokročilejší práce s daty	259
13.1	Atributy a metody třídy (statické atributy a metody)	259
	Atributy třídy	260
	Metody třídy	260
	Úkoly	262
13.2	Čtení chybových hlášení	263
13.3	Lokální proměnné	266
13.4	Konstanty a literály	269
	Konstanty objektových typů	271
13.5	Správná podoba literálů	272
	<code>boolean</code>	272
	<code>int</code>	272
	<code>long</code>	273
	<code>short, byte</code>	273
	<code>double</code>	274
	<code>float</code>	275
	<code>char</code>	275
	<code>String</code>	276
	<code>null</code>	277
13.6	Překrývání metod	277

	Opakování.....	277
	Anotace <code>@Override</code>	278
13.7	Metoda <code>toString()</code> – podpis objektu	279
	Sčítání řetězců.....	280
	Jak definovat metodu <code>toString()</code>	280
13.8	Shrnutí – co jsme se v kapitole naučili.....	281
14.	Komentáře a dokumentace	284
14.1	Zapouzdření a skrývání implementace	284
	Rozhraní \times implementace	285
	Signatura \times kontrakt	286
14.2	Komentáře a dokumentace	287
	Proč psát srozumitelné programy.....	287
	Druhy komentářů	289
	Dokumentační komentáře.....	289
14.3	Zakomentování a odkomentování částí programu.....	290
14.4	Pomocné značky pro tvorbu dokumentace	290
14.5	Okomentování třídy <code>Stream</code>	292
14.6	<i>BlueJ</i> a programátorská dokumentace	300
14.7	Uspořádání jednotlivých prvků v těle třídy	301
14.8	Prázdná standardní třída	303
14.9	Shrnutí – co jsme se naučili	304
15.	Operace a operátory	307
15.1	Jednoduché okenní vstupy a výstupy.....	307
	Textové řetězce	308
	Rozdíl mezi prázdným řetězcem a null.....	309
	Čísla	310
15.2	Podrobnosti o operátorech	312
	Binární operátory + - * / %	313
	Sčítání, odčítání, násobení	313
	Slučování řetězců +	313
	Dělení /	314
	Zbytek po dělení (dělení modulo) %	315
	Unární operátory + -	315
	Kulaté závorky ().....	316
	Přiřazovací operátor =	316
	Složené přiřazovací operátory +=, -=, *=, /=, %=	317
	Operátor přetypování (typ)	318
	Explicitní a implicitní přetypování.....	320
	Univerzální přetypování na <code>String</code>	320
15.3	Primitivní a obalové datové typy	321
15.4	Počítáme instance	321
15.5	Inkrementační a dekrementační operátory	324
	Způsoby předávání hodnot.....	327
	Jiný způsob inicializace rodného čísla.....	328
15.6	Standardní výstup	329
	Standardní chybový výstup.....	331
15.7	Shrnutí – co jsme se naučili	331
16.	Definice testovací třídy	334
16.1	Opakování	334
	Knihovna <code>JUnit</code>	335
16.2	Útroby prázdné testovací třídy	336
16.3	Přípravek	338
	Ruční úprava přípravku.....	339

	Interaktivní doplnění přípravku	340
16.4	Automaticky generované testy	341
16.5	Vlastní testy	342
16.6	Úklid	343
16.7	Metody assertEquals a assertTrue	344
16.8	Pomocné metody z rodiny assertEquals	345
16.9	Vylepšení třídy Testy2	348
16.10	Vzájemné volání testovacích metod	348
16.11	Shrnutí – co jsme se naučili	350
17.	Ladění programů	351
17.1	Krokování programu	352
17.2	Okno debuggeru	356
	Vlákna	356
	Pořadí volání – zásobník návratových adres	357
	Atributy třídy	358
	Atributy instancí	358
	Lokální proměnné	358
17.3	Krokování konstruktoru	359
17.4	Atributy a proměnné objektových typů	359
17.5	Už nezastavuj – ruším zarážky	361
17.6	Předčasný konec programu	361
17.7	Pozastavení běžícího programu	361
17.8	Shrnutí – co jsme se naučili	362
18.	Implementace rozhraní	364
18.1	Syntaxe interfejsu	364
	Zakomentovaná anotace @Override	366
	Signatura × kontrakt	367
18.2	Implementace rozhraní ve zdrojovém kódu	367
18.3	Přizpůsobení tříd novému projektu	369
	Překlad třídy Svět1o	370
	Překlad pro zjištění chyby	370
	Přidání implementované metody	371
	Překlad třídy Svět1oTest a spuštění testů	372
	Definice přípravku	372
	Dokončení definice metody nakresli(Kreslítko)	373
	Překlad třídy Strom	374
	Metoda nakresli(Kreslítko)	374
	Metoda alej()	375
	Atribut pro SprávcePlátna	375
	Vyhledávání a nahrazování textů v souborech	376
	Úpravy třídy StromTest a spuštění testů	377
	Testovací přípravek	377
	Metoda testAlej()	378
	Metoda testPosuny()	378
	Metoda testSmažZobraz()	379
	Metoda testZarámuj()	379
	Metoda testZarámujStatic()	381
	Závěrečné úpravy	381
	Úpravy posunových metod	381
	Efektivita vykreslování	382
	Zefektivnění přesunu	383
	Vnořený blok	383
	Další úpravy	384
18.4	Shrnutí – co jsme se naučili	384

19.	Samostatná aplikace – UFO.....	387
19.1	Poloprázdná třída a zástupné metody	387
19.2	Závěrečný příklad – UFO	388
	Předběžné poznámky	389
	Stručný přehled.....	389
	Třída <i>Dispečer</i>	391
	Jednodušší varianta.....	392
	Varianta ovládaná z klávesnice	392
	Třída <i>UFO_Moje</i>	393
	Atributy.....	393
	Konstruktor.....	394
	Metoda <i>getKrokTahu()</i>	394
	Metoda <i>setRychlost(int,int)</i>	394
	Metody <i>getX(), getY(), getXRychlost(), getYRychlost(), getXTah(),</i> <i>getYTah()</i>	394
	Metoda <i>zobraz()</i>	394
	Metoda <i>popojed(int)</i>	395
	Metody <i>vpravo(), vlevo(), vzhůru(), dolů(), vypniMotory()</i>	396
	Metoda <i>toString()</i>	396
	Třída <i>UFO_Demo</i>	396
	Třída <i>UFOTest</i>	396
19.3	<i>BlueJ</i> a editace větších souborů.....	397
	Podbarvování bloků a formátování textu	397
	Grafický posuvník.....	398
	Nápověda při zadávání volané metody.....	400
19.4	Vytvoření samostatné aplikace	400
	Třída spouštějící aplikaci	400
	Prohlížení obsahu JAR-souborů.....	401
	Vytvoření souboru JAR s aplikací.....	402
	Stěhování projektu mezi platformami.....	404
	Problémy s kódováním znaků	405
19.5	Shrnutí – co jsme se naučili	406
Část 3: Základní programovací techniky		407
20.	Refaktorace	408
20.1	Jedináček (Singleton)	408
20.2	Ukázkový příklad	409
20.3	Třídy <i>ČernáDíraTest</i> a <i>TŘÍDA</i>	411
20.4	Třída <i>ČernáDíra</i> – výchozí verze	411
20.5	Pachy v kódu	415
20.6	Refaktorování.....	416
20.7	Refaktorace třídy <i>ČernáDíra</i>	417
	1. krok: Převod pomocných proměnných na atributy	418
	2. krok: Definice obálky pro zbylé pomocné proměnné	420
	Předání parametru hodnotou a odkazem	421
	3. krok: Úprava metody <i>spolkní(Elipsa)</i> s využitím obálky	422
	4. krok: Vyjmutí kódu do samostatných metod.....	424
	5. krok: Další úprava definovaných metod.....	424
	Použití přesouvače a kompresoru	426
	Odstranění obálky.....	426
	Shrnutí	427
20.8	Shrnutí – co jsme se naučili	429
21.	Hodnotové a odkazové objektové typy.....	431

21.1	Převraky	431
21.2	Implementace několika rozhraní	433
21.3	Implementace rozhraní IPosuvný třídou Strom	433
	Test správnosti řešení	434
21.4	Hodnotové a odkazové objektové typy	436
	Odkazové datové typy	436
	Hodnotové typy	437
	Program demonstrující rozdíl	437
21.5	Operátory vracející logickou hodnotu	439
	Operátor rovnosti ==	439
	Operátor nerovnosti !=	440
	Operátory porovnání < <= >= >	440
	Operátor negace !	441
	Operátor logické konjunkce &&	441
	Operátor logické disjunkce 	441
	Operátor instanceof	441
21.6	Metoda equals(Object)	442
21.7	Metoda equals(Object) pro třídu Pozice	442
21.8	Proměnné a neměnné hodnotové typy	445
21.9	Projekt Zlomky	446
	Spolupráce instancí různých tříd	447
	Třídy ZlomekTest a TŘÍDA	448
	Knihovni třída Funkce	448
	Splnění požadavků na funkcionalitu	448
	Typy parametrů a návratových hodnot dceřiných metod	450
21.10	Shrnutí – co jsme se naučili	450
22.	Složitější rozšíření funkčnosti	452
22.1	Implementace rozhraní INafukovací	452
	1. krok: Vytvoření testu	453
	2. krok: Doplnění zástupných verzí přidávaných metod	453
	3. krok: Definice těla metody getRozměr()	455
	4. krok: Definice těla metody setRozměr(Rozměr)	455
	5. krok: Definice nových atributů	456
	6. krok: Kopírování těla konstrukturu do těla metody	457
	7. krok: Dočasné „odkonstantnění“ některých atributů	457
	8. krok: Definice potřebných lokálních proměnných	457
	9. krok: Odstranění tvorby nových instancí koruny a kmene	458
	10. krok: Jediné, nepřerušitelné překreslení	458
	11. krok: Vracení koruny a kmene mezi konstanty	459
	12. krok: Vyvolání metody setRozměr(int, int) v konstrukturu	460
	13. krok: Odstranění zdvojeného kódu z konstrukturu	461
	14. krok: Přidání kvalifikace atributů do příkazů k jejich nastavení	461
22.2	Implementace rozhraní ITvar	462
	15. krok: Implementace rozhraní ITvar a její test	462
	16. krok: Implementace rozhraní ITvar	462
	17. krok: Test správnosti implementace	463
22.3	Shrnutí – co jsme se naučili	464
23.	Budete si to přát zabalit?	465
23.1	Velké programy a jejich problémy	466
23.2	Balíčky	466
	Podbalíčky	468
	Názvy balíčků	468
	Uspořádání podbalíčků s programy k minulému vydání knihy	468
	Názvy tříd	470

23.3	Balíčky a <i>BlueJ</i>	470
	Příprava stromu balíčků pro <i>BlueJ</i> ve správci souborů	471
	Příprava stromu balíčků v <i>BlueJ</i>	471
	Vytvoření struktury balíčků pro tuto kapitolu.....	471
	Putování stromem balíčků	472
	Odstraňování balíčků	473
	Zavírání a otevírání projektů	474
23.4	Naplňujeme balíčky	475
	Automatické vložení příkazu <code>package</code>	476
23.5	Složitější uspořádání balíčků	476
23.6	Balíčky a příkaz <code>import</code>	477
	Balíček <code>cz.pecinovsky.česky.mojj_7.správce</code>	477
	Balíček <code>cz.pecinovsky.česky.mojj_7.příklady.zlomky</code>	478
	Zprovoznění balíčku <code>cz.pecinovsky.česky.mojj_7.správceplátna</code>	478
	Import celého balíčku	480
	Import a podbalíčky	481
	Balíček <code>java.lang</code>	481
	Změna balíčku	481
	Otevření projektu se stromem balíčků	482
23.7	Přístupová práva v rámci balíčku	483
23.8	Neveřejné třídy	484
23.9	Degenerovanost kořenového balíčku	485
23.10	Tvorba vlastních aplikací	486
23.11	Statický import	486
23.12	Shrnutí – co jsme se naučili	487
	Rejstřík	490

Skrytí spoluautoři

Knihu bych nemohl dokončit v předepsaném čase, kdyby mi s ní nepomohli moji studenti, kteří se podíleli především na přípravě doprovodných programů a dalšího podpůrného programového vybavení a přispěli i řadou připomínek k obsahu a stylu výkladu. Dovolte mi proto abecedně uvést alespoň ty nejzasloužilejší.

Daniel Bartoň se podílel na odladění některých knihovních programů, převodu knihoven do angličtiny pro druhý díl učebnice a úpravách speciální třídy umožňující definici jediné testovací třídy pro celou skupinu tříd se společným rodičem.

Sergej Bobuskyy měl hlavní podíl na vývoji pluginu BJ2NB, který doplňuje možnosti prostředí *NetBeans*, v němž budeme pracovat v druhém dílu, o schopnost průběžného zobrazování diagramu tříd a jeho provázání se zdrojovým kódem, na které si zvyknete u prostředí *BlueJ*, jež se používá v dílu prvním.

Martin Fiala vyvinul pro plugin BJ2NB editor kopenogramů.

David Král pomáhal s přípravou podkladů pro generátor projektů a s úpravami tohoto generátoru v rámci jeho průběžného zdokonalování.

Filip Malý doplnil a upravil sadu maker, která automatizují některé činnosti spojené s přípravou rukopisu a jeho následného převodu do finální podoby. Podílel se i na vývoji knihovny pro automatizované testování vyvinutých programů.

Vladimír Oraný v rámci své diplomové práce vyvinul (a před vydáním knihy na poslední chvíli ještě upravil) speciální knihovnu umožňující výrazně rozšířit možnosti deklarace požadavků, kterým musejí vyhovovat testované programy. Setkáte se s ní v druhém dílu.

Jarmila Pavlíčková a **Luboš Pavlíček** nejsou mí studenti, ale kolegové na VŠE. Mnohé z formulací použitých ve výkladu se vytříbily na základě našich četných (a mnohdy i vášnivých) debat o výuce programování. Jarmile bych chtěl navíc poděkovat za to, že mne osvobozuje od řady administrativních úkonů, které jsou pro mne téměř nevládnutelnou překážkou.

Martin Vondráček podrobně pročetl celý rukopis a přispěl řadou poznámek k zvýšení jeho čitelnosti, srozumitelnosti a odborné přesnosti.

Na závěr pak musím vyjádřit svůj velký dík firmě ICZ a veškerému osazenstvu oddělení Realizace. Bez jejich podpory by kniha nevznikla.

Předmluva k prvnímu vydání

Rudu Pecinovského jsem poprvé potkal v době, kdy jsme oba studovali na Jaderné fakultě ČVUT v Praze. Doopravdy jsme se ale poznali až mnohem později, když jsme na počátku devadesátých let spolupracovali na překladu manuálů k jistému dodnes populárnímu programovému prostředí. Brzy jsme zjistili, že máme jeden společný zájem – učit lidi, jak kvalitně psát programy.

V současné době dominuje při tvorbě aplikací objektově orientované programování. Moderní vývojové nástroje, které jsou na trhu k dispozici, jeho znalost předpokládají, aplikační knihovny z něj vycházejí, softwarové firmy ho vyžadují, nově vznikající programovací jazyky jsou čistě objektové. A když už jsme u těch jazyků: Java je dnes asi nejpoužívanější jazyk pro vývoj nových aplikací a zcela určitě to je jazyk, který se nejdynamičtěji rozvíjí. Přesto téměř všechny učebnice Javy, které na trhu najdete, začínají procedurálním programováním a k objektově orientovanému programování se dostanou až ke konci. Objekty pak často vypadají jako nepříliš pohodlná nadstavba nad procedurálním programováním.

Řekl jsem, že tak vypadají *téměř* všechny knihy. Kniha Rudy Pecinovského je totiž velmi příjemnou výjimkou. Je to učebnice, která objekty opravdu začíná a prvních několik kapitol se ani ničím jiným nezabývá. Teprve poté, co zvládnete základní pojmy a dovednosti objektově orientovaného programování, se začne zabývat konstrukcemi, jako je cyklus nebo podmínka. Tento postup, který si autor vyzkoušel na začínajících programátorech v programátorských kroužcích a který používá při výuce profesionálů, vás naučí od počátku myslet objektově. Ukazuje objekty jako něco opravdu přirozeného, jako něco, co výrazně usnadňuje přemýšlení o řešené úloze.

Při čtení Rudovy knihy jsem občas litoval, že už umím programovat, a tak jen doufám, že slibované další díly budou stejně dobré.

M. Virius

Úvod

Otevíráte třetí vydání knížky, která vás chce naučit programovat moderním, objektivě 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.

Co je nového ve 3. vydání

Oproti předchozímu vydání je kniha od základů přepracovaná. Kniha je nyní rozdělena do dvou dílů. První díl se soustředí především na výklad základních architektonických principů, které by si měl čtenář osvojit předtím, než se pustí do kódování složitějších projektů. Jeho cílem je, aby čtenáři přešly tyto principy do krve dřív, než se začne soustředit na kód.

Druhý díl pak získané návyky prohloubí a seznámí čtenáře s řadou dalších programátorských technik. Vysvětlí hlubší souvislosti, na něž v běžných učebnicích již nezbyvá místo, a doplní čtenářovy znalosti syntaxe jazyka. Současně učí práci s profesionálním vývojovým nástrojem a seznámí čtenáře s jeho základními vlastnostmi.

Druhý díl je možno použít jako samostatnou učebnici pro ty, kteří již zvládli základní architektonické principy návrhu objektových aplikací z jiných učebnic.

Komu je kniha určena

Tato knížka je určena těm, kteří to se svojí touhou naučit se moderně programovat myslí vážně a chtějí se naučit programovat dobře. Zaměřuje se na ty, kteří nechtějí zůstat obyčejnými kodéry, jejichž práci postupně přebírají nejrůznější nástroje, ale chtějí se propracovat mezi špičkové architekty, kteří umějí navrhnout optimální řešení splňující požadavky zákazníka (a jsou podle toho také ohodnoceni).

Knížka je sice primárně určena těm, kteří ještě nikdy neprogramovali, ale ukázalo se, že se z ní dozvedí hodně nového i poměrně zkušení programátoři. První vydání jsem psal na základě zkušeností z mnoha kurzů. Při vedení těchto kurzů jsem si uvědomil, že to, co děti (a částečně i dospělí, kteří s programováním teprve začínají) pochopí poměrně snadno, zvládají programátoři s předchozími zkušenostmi z neobjektového programování obtížně. Spoustu úsilí totiž musí věnovat

tomu, aby se nejprve odnaučili mnohé z toho, co si před tím pracně osvojili. Teprve pak začnou pomalu vstřebávat jiný způsob programátorského myšlení, jež objektově orientované programování vyžaduje¹.

Od čtenářů předchozích vydání jsem dostal řadu mailů, v nichž mi psali, že jsou sice zkušenými programátory, ale teprve zde pochopili některé věci, které jim ostatní učebnice a kurzy nedokázaly vysvětlit. Uvědomil jsem si přitom, že knihu mohou s výhodou použít i ti, kteří již nějakou dobu programují a potřebují se přeškolit z klasického, strukturovaného programování na programování objektově orientované (80 % účastníků mých kurzů).

Tady bych ale chtěl případně zkušenější čtenáře upozornit na jedno úskalí: dospělý člověk již, na rozdíl od dětí, nedokáže přijmout nové informace, aniž by je okamžitě nespojoval se svými dosavadními znalostmi a zkušenostmi. Dozví-li se proto informaci, která neodpovídá zcela jeho dosavadním zkušenostem, tak ji podvědomě „ohne“, aby se s nimi dostala do souladu. To je nejčastějším zdrojem problémů zkušenějších programátorů při přechodu na nové paradigma, protože to, co si zapamatují, bývá v důsledku výše zmíněného „ohnutí“ něco trochu jiného, než co se jim snažil přednášející sdělit.

Co se naučíte

Musím vás upozornit na to, že kniha, kterou držíte v ruce, se od běžných učebnic poněkud liší. Ostatní učebnice jsou totiž většinou především učebnicemi nějakého programovacího jazyka. Jejich autoři se proto ve svém výkladu soustředí hlavně na výklad vlastností popisovaného jazyka a jeho knihoven. Bohužel se v nich ale nedozvíte skoro nic o tom, jak při návrhu programů přemýšlet, aby vás nezaskočily náhlé změny zadání, kterými je současné programování pověstné. Takovéto učebnice proto nevychovávaly návrháře, kteří by uměli program navrhnout, ale pouze kodéry, kteří umějí zanalyzované zadání zakódovat.

Vypadá to, jako když autoři předpokládají, že se při čtení jejich knihy naučíte programovat nějak sami od sebe obdobně, jako se to museli naučit oni. Zkušenosti s programátory, kteří navštěvují moje kurzy ve firmě či na univerzitě, však ukazují, že tohoto výsledku bývá dosaženo jen zřídka. Většina účastníků mých kurzů zná poměrně dobře konstrukce nějakého objektově orientovaného programovacího jazyka. Bohužel, skoro nikdo z nich v něm neumí objektově programovat², neumí přepnout na objektový způsob uvažování.

¹ Statistiky uvádějí, že typická doba přechodu na objektové paradigma je 12 až 18 měsíců, přičemž čím je programátor zkušenější, tím déle mu přerod trvá.

² Výzkum z přelomu století ukázal, že pouze 10 % programů psaných v objektově orientovaných jazycích je navrženo opravdu objektově. (Goddard, D. 1994. Is it really object oriented?