

Pridružite se diskusiji na p2p.wrox.com



PREVOD
DRUGOG IZDANJA



Java[®] 8

PROGRAMIRANJE



Java[®] 8

PROGRAMIRANJE

PREVOD DRUGOG IZDANJA

Yakov Fain

 kompjuter
biblioteka



WILEY

Wiley Publishing, Inc.

Izdavač:



Obalskih radnika 15, Beograd

Tel: 011/2520272

e-mail: kombib@gmail.com

internet: www.kombib.rs

Urednik: Mihailo J. Šolajić

Za izdavača, direktor:

Mihailo J. Šolajić

Autor: Yakov Fain

Prevod: Goran Janačković

Lektura: Miloš Jevtović

Slog : Zvonko Aleksić

Znak Kompjuter biblioteke:

Miloš Milosavljević

Štampa: „Pekograf“, Zemun

Tiraž: 500

Godina izdanja: 2015.

Broj knjige: 480

Izdanje: Prvo

ISBN: 978-86-7310-503-1

Java® Programming Second Edition

by Yakov Fain

ISBN: 978-1-118-95145-3

Copyright © 2015. Yakov Fain. All rights reserved.

Published by Wiley Publishing, Inc.

All right reserved. No part of this book may be reproduced or transmitted in any form or by means, electronic or mechanical, including photocopying, recording or by any information storage retrieval system, without permission from the Publisher.

Autorizovani prevod sa engleskog jezika edicije u izdanju „Wiley Publishing, Inc.“, Copyright © 2015.

Sva prava zadržana. Nije dozvoljeno da nijedan deo ove knjige bude reprodukovana ili snimljen na bilo koji način ili bilo kojim sredstvom, elektronskim ili mehaničkim, uključujući fotokopiranje, snimanje ili drugi sistem presnimavanja informacija, bez dozvole izdavača.

Zaštitni znaci

Kompjuter Biblioteka i „Wiley Publishing, Inc.“ su pokušali da u ovoj knjizi razgraniče sve zaštitne oznake od opisnih termina, prateći stil isticanja oznaka velikim slovima.

Autor i izdavač su učinili velike napore u pripremi ove knjige, čiji je sadržaj zasnovan na poslednjem (dostupnom) izdanju softvera. Delovi rukopisa su možda zasnovani na predizdanju softvera dobijenog od strane proizvođača. Autor i izdavač ne daju nikakve garancije u pogledu kompletnosti ili tačnosti navoda iz ove knjige, niti prihvataju ikakvu odgovornost za performanse ili gubitke, odnosno oštećenja nastala kao direktna ili indirektna posledica korišćenja informacija iz ove knjige.

CIP - Каталогизација у публикацији
Народна библиотека Србије, Београд

004.438JAVA
004.42:004.738.5

ФАИН, Јаков

Java 8 programiranje / Yakov Fain ; prevod drugog izdanja [Goran Janačković]. - 1. izd. - Beograd : Kompjuter biblioteka, 2015 (Zemun: Pekograf). - XXV, 583 str.: ilustr.; 24 cm. - (Kompjuter biblioteka; br. knj. 480)

Prevod dela: Java Programming. - Tiraž 500. - Registar.

ISBN 978-86-7310-503-1

a) Програмски језик „Java“
b) Интернет - Програмирање
COBISS.SR-ID 219150348

Kratak sadržaj

LEKCIJA 1	
Uvod u programski jezik Java	1
LEKCIJA 2	
Eclipse IDE okruženje	9
LEKCIJA 3	
Objektno-orijentisano programiranje pomoću Java jezika	21
LEKCIJA 4	
Metodi klase i konstruktori	37
LEKCIJA 5	
Java sintaksa: bitovi i delovi	47
LEKCIJA 6	
Paketi, interfejsi i enkapsulacija	61
LEKCIJA 7	
Programiranje korišćenjem apstraktnih klasa i interfejsa	73
LEKCIJA 8	
Osnove grafičkog interfejsa pomoću Swing biblioteke komponenata	83
LEKCIJA 9	
Rukovanje događajima u Swing grafičkom interfejsu	99
LEKCIJA 10	
Rukovanje greškama	115

LEKCIJA 11**Uvod u kolekcije** 127**LEKCIJA 12****Uvod u generičke tipove** 141**LEKCIJA 13****Lambda izrazi i funkcionalno programiranje** 151**LEKCIJA 14****Ulazno-izlazni tokovi** 171**LEKCIJA 15****Java serijalizacija** 185**LEKCIJA 16****Osnove mrežnog programiranja** 195**LEKCIJA 17****Konkurentnost i višenitno izvršavanje** 209**LEKCIJA 18****Osnove JavaFX grafičkog korisničkog interfejsa** 233**LEKCIJA 19****Razvoj JavaFX kalkulatora
i lks-oks igre** 251**LEKCIJA 20****Stream API interfejs** 281**LEKCIJA 21****Korišćenje relacionih baza podataka pomoću JDBC drajvera** 297**LEKCIJA 22****Prikazivanje tabličnih podataka u grafičkom korisničkom interfejsu** 311**LEKCIJA 23****Anotacije i refleksija** 321**LEKCIJA 24****Poziv udaljenih procedura** 335

LEKCIJA 25	
Java EE 7 platforma	345
LEKCIJA 26	
Programiranje korišćenjem servleta	355
LEKCIJA 27	
JavaServer Pages tehnologija	379
LEKCIJA 28	
Razvoj WebSocket web aplikacija	395
LEKCIJA 29	
Osnove JNDI interfejsa	413
LEKCIJA 30	
Osnove JMS i MOM	423
LEKCIJA 31	
Uvod u Enterprise JavaBeans	445
LEKCIJA 32	
Pregled Java Persistence API interfejsa	463
LEKCIJA 33	
Korišćenje RESTful web servisa	481
LEKCIJA 34	
Java Logging API interfejs	499
LEKCIJA 35	
Osnove testiranja programskih jedinica pomoću JUnit radnog okvira	511
LEKCIJA 36	
Automatizovanje razvoja pomoću Gradle alata	527
LEKCIJA 37	
Java intervjui za posao	551
INDEKS	565

Sadržaj

LEKCIJA 1

Uvod u programski jezik Java

Zašto učiti programski jezik Java?	1
Postavljanje ciljeva	2
„Životni ciklus“ Java programa	3
JDK i JRE	3
Preuzimanje i instaliranje Java SE alata	3
Instaliranje JDK 8 u MAC operativnim sistemima	3
Instaliranje JDK 8 u Windows operativnim sistemima	4
Vaš prvi Java program: Hello World	5
Prevođenje i izvršavanje Hello World programa	7
Probajte sami	7
Zahtevi	8
Korak po korak	8

LEKCIJA 2

Eclipse IDE okruženje

Uvod u Eclipse integrisano okruženje	9
Preuzimanje i instaliranje Eclipse okruženja	10
Kreiranje Hello projekta u Eclipse okruženju	11
Kreiranje HelloWorld klase u Eclipse okruženju	14
Java paketi	15
Kompletiranje generisanja koda	16
Dodatni materijali	18
Probajte sami	18
Zahtevi	19
Korak po korak	19

LEKCIJA 3

Objektno-orientisano programiranje pomoću Java jezika	21
Klase i objekti	21
Promenljive i tipovi podataka	23
Deklarisanje promenljivih	23
Final promenljive	23
Osnovni tipovi podataka	24
Domet promenljivih	25
Klase omotači i automatsko konvertovanje u objekte i osnovne tipove	26
Komentari u programu	26
Vaš prvi koristan program	27
Deklarisanje Tax klase	27
Dodavanje metoda u Tax klasu	28
Deklarisanje druge klase TestTax	29
Uslovna naredba if	30
Naredba switch	31
„Nasleđivanje“	32
Predefinisanje metoda	33
Dodatni materijali	33
Probajte sami	33
Zahtevi	33
Savet	34
Korak po korak	34

LEKCIJA 4

Metodi klase i konstruktori	37
Argumenti metoda	37
Preopterećenje operatora	38
Konstruktori	39
Ključna reč super	40
Ključna reč this	40
Prosleđivanje argumenata po vrednosti ili po referenci	42
Dometi promenljive	43
Ključna reč static	44
Probajte sami	45
Zahtevi	45
Korak po korak	45

LEKCIJA 5

Java sintaksa: bitovi i delovi	47
Polja	47
Više o stringovima	49
Operator ==	50
Petlje	50
Debugovanje Java programa	54

Dodatne informacije o if i switch naredbama	57
Načini korišćenja if naredbi	57
Argumenti u komandnoj liniji	58
Probajte sami	59
Zahtevi lekcije	59
Korak po korak	60
LEKCIJA 6	
Paketi, interfejsi i enkapsulacija	61
Java paketi	61
Enkapsulacija	62
Nivoi pristupa	63
Ključna reč final	63
Promenljiva final	64
Final metodi	64
Final klase	64
Interfejsi	65
Marker interfejsi	66
Podrazumevani metodi u interfejsima	67
Statički metodi u interfejsima	68
Kastovanje	68
Probajte sami	70
Zahtevi	70
Korak po korak	70
LEKCIJA 7	
Programiranje korišćenjem apstraktnih klasa i interfejsa	73
Apstraktne klase	73
Zadatak	73
Rešenje korišćenjem apstraktne klase	74
Polimorfizam	76
Kreiranje polimorfičnog interfejsa	77
Interfejsi i apstraktne klase	78
Probajte sami	79
Zahtevi lekcije	79
Korak po korak	79
Deo 1	79
Deo 2	81
LEKCIJA 8	
Osnove grafičkog interfejsa pomoću Swing biblioteke komponenata	83
Swing osnove	83
Menadžeri rasporeda elemenata	86
Jednostavni kalkulator pomoću FlowLayout menadžera	86
Kratak opis menadžera za definisanje rasporeda elemenata	87

FlowLayout	88
GridLayout	88
BorderLayout	90
Kombinovanje menadžera raporeda	90
BoxLayout	93
GridBagLayout	94
CardLayout	95
Kontejneri sa Absolute Layout rasporedom	96
Dodatni Swing elementi	96
Alati za kreiranje Swing interfejsa	97
Probajte sami	97
Zahtevi	97
Korak po korak	97

LEKCIJA 9

Rukovanje događajima u Swing grafičkom interfejsu	99
Osnove identifikovanja događaja	99
Učenje kalkulatora da računa	100
Registrowanje komponentata pomoću ActionListener interfejsa	101
Pronalaženje uzroka pojave događaja	102
How to Pass Data Between Objects	104
Pažnja, loša praksa!	104
Bolje rešenje korišćenjem javnog API interfejsa	105
DESIGN PATTERN MODEL-VIEW-CONTROLLER	107
Dodatni Swing „oslušivači“ pojave događaja	107
Način korišćenja adaptera	109
Unutrašnje klase	110
Anonimne unutrašnje klase	111
Probajte sami	112
Zahtevi	112
Korak po korak	112

LEKCIJA 10

Rukovanje greškama	115
Praćenje stanja steka	115
Java izuzeci	116
Hijerarhija izuzetaka	117
Try/catch blokovi	118
Korišćenje throws klauzule	119
Korišćenje finally klauzule	120
Try-With-Resources sintaksa	121
Ključna reč throw	122
Keirajte sopstvene izuzetke	123
Probajte sami	125
Zahtevi	125
Korak po korak	125

LEKCIJA 11

Uvod u kolekcije	127
Ponovno razmatranje polja	128
Interfejsi kolekcija iz java.util paketa	128
Dinamička polja pomoću ArrayList klase	129
Klase Hashtable i HashMap	132
Svojstva klasa	133
Klase Enumeration i Iterator	135
Klasa LinkedList	135
Klasa BitSet	137
Izbor prave kolekcije	138
Probajte sami	139
Zahtevi	139
Korak po korak	139

LEKCIJA 12

Uvod u generičke tipove	141
Generički tipovi i klase	141
Deklarisanje generičkih tipova	144
Džoker znaci	144
Kreiranje sopstvenih parametrizovanih klasa	146
Parametri povezanog tipa	147
Generički metodi	149
Probajte sami	150
Zahtevi	150
Korak po korak	150

LEKCIJA 13

Lambda izrazi i funkcionalno programiranje	151
Imperativni ili funkcionalni stil	152
Šta je lambda izraz?	153
Funkcionalni interfejsi	154
Reference metoda	155
Metodi i funkcije	157
Prosleđivanje funkcija u metode	158
Pristupanje kolekcijama pomoću foreach() metoda	160
Lambda izrazi u odnosu na „nasleđivanje“ i polimorfizam	162
Eliminisanje „nasleđivanja“	165
Function i BiFunction interfejsi	167
Probajte sami	169
Zahtevi	169
Korak po korak	170

LEKCIJA 14

Ulazno-izlazni tokovi	171
Tokovi bajtova	172
Baferovani tokovi	173
Tokovi karaktera	174
Povezivanje GUI i I/O tokova	175
Tokovi podataka	178
Pomoćne klase za rad sa datotekama	179
File klasa	179
NIO.2: Korišćenje Files, Path i Paths klasa i interfejsa	180
Šta predstavlja NIO?	182
Probajte sami	183
Zahtevi	184
Korak po korak	184

LEKCIJA 15

Java serijalizacija	185
Klasa ObjectOutputStream	187
Klasa ObjectInputStream	188
Interfejs Externalizable	189
Verzije klasa	191
Serijalizacija u polje bajtova	192
Probajte sami	193
Zahtevi	193
Korak po korak	194

LEKCIJA 16

Osnove mrežnog programiranja	195
Očitavanje podataka sa Interneta	196
Povezivanje pomoću HTTP proksi servera	198
Preuzimanje datoteka sa Interneta	199
Specificiranje parametara iz komandne linije za FileDownload program	200
Program za berzanske kvote	200
Programiranje soketa	203
Zbog čega se koriste soketi?	204
Server berzanskih podataka koji koristi sokete	204
Probajte sami	207
Zahtevi	207
Preporuke	207
Korak po korak	207

LEKCIJA 17

Konkurentnost i višenitno izvršavanje	209
Klasa Thread	210
Interfejs Runnable	211

Eliminisanje nasleđivanja	213
Zaustavljanje izvršavanja niti	213
Uklanjanje niti	215
Prioriteti niti	217
Sinhronizacija niti i uslovi konkurentnog pristupanja	217
Stanja niti	219
Wait i notify metodi	219
Pristup okruženju na osnovu konteksta u Java jeziku	221
Povezivanje niti	222
Korisni sadržaji java.util.concurrent paketa	224
Klasa ReentrantLock i ključna reč synchronized	224
Radni okvir Executor	225
Kratak prikaz konkurentnih kolekcija	228
Redovi	228
Kolekcije	229
SwingWorker nit	229
Probajte sami	232
Zahtevi	232
Korak po korak	232

LEKCIJA 18

Osnove JavaFX grafičkog korisničkog interfejsa **233**

Osnove JavaFX aplikacije	233
Korišćenje E(fx)clipse dodatka	234
Rasporedi elemenata	236
Jednostavna aplikacija sa HBox rasporedom elemenata	237
Primer aplikacije u kojoj se koristi GridPane raspored elemenata	239
Definisanje prikaza pomoću CSS koda	240
Rukovanje događajima	244
Svojstva i povezivanje	246
Probajte sami	250
Zahtevi	250
Korak po korak	250

LEKCIJA 19

Razvoj JavaFX kalkulatora i Iks-oks igre **251**

Dizajniranje kalkulatora pomoću Scene Builder alata	251
Dizajniranje Calcultor GUI interfejsa pomoću Scene Builder alata	254
Rukovanje događajima u Controller klasi	260
Prepoznavanje izvora događaja	261
Prosleđivanje podataka iz View u	
Controller deo aplikacije i nazad	263
Programiranje Iks-oks igre	265
Strategija igre	265
Dizajniranje Iks-oks GUI interfejsa pomoću FXML i CSS koda	266
Implementiranje strategije igre u Iks-oks kontroleru	273
Iks-oks Play meni	277

Iks-oks: šta probati sledeće?	277
JavaFX na Webu i mobilnim uređajima	278
Probajte sami	278
Zahtevi	278
Korak po korak	279

LEKCIJA 20

Stream API interfejs	281
Osnove tokova	281
Intermedijalne i terminalne operacije	282
Lenjo izvršavanje	282
Paralelna i sekvencijalna obrada	285
Sortiranje kolekcija i tokova	285
Sortiranje Java kolekcija	286
Korišćenje Comparable interfejsa	286
Korišćenje Comparator interfejsa	287
Sortiranje tokova	289
Ostali izvori tokova	290
Kreiranje tokova konačne veličine	290
Kreiranje tokova beskonačne veličine	291
Geenrisanje toka podataka	291
Korišćenje Stream API interfejsa i I/O tokova	292
Operacije prekidanja	293
Probajte sami	294
Zahtevi	294
Korak po korak	294

LEKCIJA 21

Korišćenje relacionih baza podataka pomoću JDBC drajvera	297
Tipovi JDBC drajvera	298
Instaliranje Derby DB sistema i kreiranje baze podataka	298
Jednostavni JDBC program	300
Obrada rezultujućih skupova podataka	302
Klasa PreparedStatement	304
Klasa CallableStatement	304
Klasa ResultSetMetaData	305
Skrolabilni rezultujući skupovi	
i klasa RowSet	307
Transakciona ažuriranja	308
Grupe klasa i DataSource interfejs	308
Probajte sami	309
Zahtevi	309
Preporuka	309
Korak po korak	309

LEKCIJA 22**Prikazivanje tabličnih podataka u grafičkom korisničkom interfejsu 311**

JTable komponenta i MVC paradigma	311
Model	312
Obavezni metodi modela tabela	313
Opcioni metodi modela tabela	316
Metodi za prikazivanje sadržaja ćelija	318
Rezime	320
Probajte sami	320
Zahtevi	320
Korak po korak	320

LEKCIJA 23**Anotacije i refleksija 321**

Javadoc anotacije	321
Osnove Java anotacija	322
@Override anotacija	323
@SuppressWarning anotacija	323
@Deprecated anotacija	324
@Inherited anotacija	324
@FunctionalInterface anotacija	324
@Documented anotacija	325
Samostalno definisane anotacije	325
Refleksija	328
Obrada anotacija u toku izvršavanja	330
Rezime	332
Probajte sami	332
Zahtevi	332
Korak po korak	333

LEKCIJA 24**Poziv udaljenih procedura 335**

Razvoj aplikacija pomoću RMI	336
Definisanje interfejsa za udaljeni pristup	336
Implementiranje interfejsa za udaljeni pristup	337
Registrowanje udaljenih objekata	338
Pisanje RMI klijenata	339
Bezbednosna razmatranja	340
Pronalaženje udaljenih objekata	341
Probajte sami	342
Zahtevi	342
Saveti	342
Korak po korak	342

LEKCIJA 25

Java EE 7 platforma	345
Opšti prikaz platforme	345
JCP, JSR i ostale skraćenice	346
Slojevi Java EE aplikacija	346
Kontejneri i aplikacioni serveri	348
Profili i uklanjanje nepotrebnih elemenata	350
Zbog čega treba koristiti Java EE?	350
Probajte sami	352
Zahtevi	352
Korak po korak	352

LEKCIJA 26

Programiranje korišćenjem servleta	355
Opšti prikaz	355
Tanak klijent	357
Kako pisati servlet?	357
Kako postaviti servlet?	358
Konfigurisanje GlassFish servera u Eclipse IDE okruženju	359
Kako se kreira servlet u Eclipse okruženju?	362
Isporučivanje web aplikacije u obliku WER datoteke	366
Tok podataka između pregledača i servleta	366
HTTP Get i Post zahtevi	367
Praćenje sesija	368
„Kolačići“	368
Modifikovanje URL adrese	369
HttpSession na strani servera	370
Filteri	373
Asinhroni servleti	375
Probajte sami	376
Zahtevi	377
Korak po korak	377

LEKCIJA 27

JavaServer Pages tehnologija	379
Java kod ugrađen u HTML	380
Implicitni JSP objekti	383
Pregled JSP tagova	383
Direktive	384
Deklaracije	384
Izrazi	384
Skriptleti	385
Komentari	385
Standardne akcije	385
Stranice o greškama	386
Java bean klase	387
Korišćenje JavaBeans klasa u JSP kodu	388

Koliko dugo egzistira bean objekat?	388
Učitavanje JSP koda iz servleta	389
Biblioteke tagova	390
JSTL specifikacija	392
Probajte sami	393
Zahtevi	393
Korak po korak	393

LEKCIJA 28

Razvoj WebSocket web aplikacija **395**

Nedostaci HTTP protokola	396
HTTP saveti za slanje podataka sa servera	396
Klijent-server komunikacija zasnovana na WebSocket protokolu	397
Web pregledač kao WebSocket klijent	397
Komunikacija sa serverom pomoću WebSocket protokola	399
Hello WebSocket primer	400
Nadgledanje WebSocket mrežnog saobraćaja	402
Slanje poruka	403
Prijem poruka korišćenjem @OnMessage anotacije	404
Enkoderi i dekoderi	405
Slanje poruke svim klijentima	409
Probajte sami	412
Zahtevi	412
Korak po korak	412

LEKCIJA 29

Osnove JNDI interfejsa **413**

Servisi naziva i upravljanja direktorijumima	413
Korišćenje InitialContext klase	414
Dobijanje reference InitialContext objekta	414
Ubacivanje JNDI resursa	415
Administriranje JNDI objekata	
na GlassFish serveru	416
Izvori podataka i JNDI interfejs	417
Lightweight Directory Access Protocol (LDAP)	419
Probajte sami	421
Zahtevi	421
Korak po korak	421

LEKCIJA 30

Osnove JMS i MOM **423**

Koncepti i terminologija koja se odnosi na razmenu poruka	423
Dva režima isporučivanja poruka	425
Osnove OpenMQ MOM provajdera	426
Pregled JMS API interfejsa	429
Tipovi poruka	429
Direktno slanje poruke do MOM provajdera	430

Direktan prijem poruka od MOM provajdera	431
Objavljivanje poruke	433
Prijavljivanje za prijem poruka o određenoj temi	434
Potvrde o prijemu poruka i podrška za transakcije	435
Selektori poruka	436
Slanje poruka iz Java EE kontejnera	437
Administriranje JMS objekata u GlassFish serveru	438
Probajte sami	441
Zahtevi	442
Saveti	442
Korak po korak	442

LEKCIJA 31

Uvod u Enterprise JavaBeans

445

Kome su neophodni EJB kontejneri?	445
EJB tipovi	446
Bean objekti za sesije bez stanja	447
Bean klasa	447
Klijentski pogled	447
Lokalne bean klase bez implementacije interfejsa	448
Lokalne Bean klase	450
Udaljene bean klase	452
Asinhroni metodi i konkurentnost	453
Bean klase za sesije sa stanjima	454
Singleton bean klase	455
Isporučivanje EJB aplikacija	456
Bean klase upravljane porukama	458
EJB objekti i transakcije	459
Vremenski servis	460
Rezime	461
Probajte sami	461
Zahtevi	461
Preporuka	461
Korak po korak	461

LEKCIJA 32

Pregled Java Persistence API interfejsa

463

Opšti prikaz	463
Mapiranje objekata u tabele baze podataka	464
Postavljanje upita o entitetima	466
JPQL jezik	466
Criteria API interfejs	467
Menadžer entiteta	468
Validacija bean objekata	471
Probajte sami	473
Zahtevi	473
Korak po korak	473

LEKCIJA 33

Korišćenje RESTful web servisa	481
SOAP web servisi	481
RESTful web servisi	482
Upotreba podataka u JSON formatu	483
Čitanje JSON podataka pomoću Streaming API interfejsa	484
Pisanje JSON koda pomoću Streaming API interfejsa	485
Pisanje JSON koda pomoću Object Model API interfejsa	486
RESTful server akcija	487
Kreiranje aplikacije	487
Kreiranje Java Bean klase Stock	488
Kreiranje StockService krajnje tačke	489
Kreiranje RESTful klijenata	493
Konteksti i ubacivanje zavisnosti	493
Probajte sami	495
Zahtevi	495
Preporuke	495
Korak po korak	496

LEKCIJA 34

Java Logging API interfejs	499
Java Logging API interfejs	500
Hello World aplikacija kreirana pomoću Java Logging API interfejsa	500
Hello World aplikacija sa globalnim loggerom	501
Hello World aplikacija sa loggerom na nivou klase	501
Korišćenje rukovalaca i podešavanje nivoa izveštavanja	502
Datoteka logging.properties	505
Generisanje logova pomoću FileHandler klase	506
Formatiranje i filtriranje	506
Formatiranje	506
Filtriranje	507
Radni okviri za logove	508
Probajte sami	509
Zahtevi	509
Korak po korak	510

LEKCIJA 35

Osnove testiranja programskih jedinica pomoću JUnit radnog okvira	511
Osnove korišćenja JUnit radnog okvira	512
Instaliranje JUnit okruženja	513
Promena podrazumevane strukture direktorijuma u Eclipse okruženju	513
Vaš prvi JUnit test primer	514
JUnit anotacije	517
Primena anotacija pri testiranju Tax klase	518
Test okruženje	521

Klase za izvršavanje JUnit testova	523
Probajte sami	524
Zahtevi	524
Korak po korak	524

LEKCIJA 36

Automatizovanje razvoja pomoću Gradle alata **527**

Hello World aplikacija u Ant alatu	528
Hello Word u Maven alatu	529
Gradle osnove	532
Hello World u Gradle alatu	533
Kreiranje Hello Word projekta	533
Izvršavanje Hello World aplikacije	535
Promena Gradle konvencija	536
Upravljanje zavisnostima pomoću Gradle alata	538
Repozitorijumi	540
Zavisnosti i konfiguracije	541
Uključivanje zavisnosti u JAR datoteku	543
Kreiranje WAR datoteke	546
Korišćenje Gradle alata u Eclipse IDE okruženju	547
Gradle Eclipse dodaci	547
Eclipse IDE okruženje i Gradle	548
Probajte sami	549
Zahtevi	550
Korak po korak	550

LEKCIJA 37

Java intervjui za posao **551**

Dolazak do intervjua	551
Uspešno obavljanje intervjua	552
Razmatranje ponude	553
Intervjuisanje programera poslovnih aplikacija	554
Da li treba polagati test za sertifikate?	555
Tehnička pitanja i odgovori	555
Epilog	563

INDEKS **565**

Uvod

Hvala vam što ste odlučili da učite Java jezik čitajući drugo izdanje ove knjige! Ova knjiga možda izgledao obimna, ali nije ukoliko se razmotre teme koje su u njoj obrađene, a i sadrži dobro režirane i korisne video materijale.

Volim ovu ediciju *dvadesetčetvoročasovne obuke* kompanije „Wiley Publishing“. Naziv ne podrazumeva da možete da naučite za 24 časa ono što je opisano u knjizi. U stvari, smatra se da imate obuku koju možete da pratite 24 časa dnevno. Svaka knjiga iz ove edicije, koju prati skup video materijala, sadrži minimum teorije da biste započeli učenje nečeg što je novo za vas.

Ova knjiga ima prateći materijal od šest sati video materijala posvećenog Java programiranju, u kome su demonstrirani savremeni koncepti, tehnike i tehnologije na način koji pojednostavljuje učenje i promovise bolje razumevanje procesa razvoja.

Programeri se često kategorizuju u početnike i standardne i napredne programere. Ukoliko ovladate svim materijalima prikazanim u ovoj knjizi, može smatrati da imate tehničke veštine standardnog Java programera. Često sam vodio tehničke intervjuje za kompaniju za koju radim i uvek sam bio veoma zadovoljan kada kandidat za poziciju srednjeg nivoa demonstrira razumevanje svih tema opisanih u ovoj knjizi.

KOME JE NAMENJENA OVA KNJIGA?

Ova knjiga je namenjena svima koji žele da nauče kako se programira korišćenjem Java jezika. Nije neophodno prethodno programersko iskustvo.

- Ovaj vodič mogu koristiti Java programeri da bi pronašli jednostavne funkcionalne primere u kojima se primenjuju neki elementi jezika.
- Iskusni Java programeri mogu koristiti ovu knjigu kao podsetnik, dok se pripremaju za intervju za tehnički posao.
- Ovaj vodič mogu koristiti univerzitetski studenti koji su zainteresovani da nešto nauče od osobe koja se praktično bavi razvojem poslovnog softvera već 25 godina i živi od toga.
- Univerzitetski profesori treba da budu zahvalni zbog toga što se svako poglavlje završava odeljkom „Probajte sami“ – pripremljenim zadatkom za svaku lekciju. Rešenja ovih zadataka su takođe obezbeđena.

Ova knjiga je vodič, ali ne u akademskom smislu. Napisao ju je praktičar i namenjena je praktičarima.

ŠTA SADRŽI OVA KNJIGA?

Da bi se neko zvao Java programer, treba da zna ne samo osnovnu sintaksu ovog programskog jezika, već i skup tehnologija koje se izvršavaju na strani servera, poznate pod nazivom Java EE (Enterprise Edition). Ova knjiga sadrži opis i jednog i drugog. U vreme dok ovo pišemo najnovija osnovna verzija je Java 8, a najnovije izdanje Java EE je 7. Te verzije su opisane u ovoj knjizi.

Java je jezik opšte namene – možete da programirate aplikacije koje se izvršavaju nezavisno na računaru korisnika, kao i aplikacije koje se povezuju sa udaljenim serverima. Možete da programirate aplikacije koje se izvršavaju isključivo na serveru. Java jezik možete da koristite za pisanje aplikacija za mobilne uređaje i programiranje igara. Živimo u Internet of Things (IoT) eri, a Java može da se ugradi u senzore u automobilima ili u kućne aparate.

U najvećem delu ove knjige opisane su Java programska sintaksa i tehnike koje se mogu primenjivati na računarima korisnika i serverima. Devet poglavlja je posvećeno Java EE tehnologijama koje se koriste za Java programe koji se izvršavaju na serverima. Završno poglavlje posvećeno je opisivanju procesa spremanja za intervju za Java tehničke poslove svih onih koji su zainteresovani da se prijave za posao Java programera.

KAKO JE STRUKTURIRANA OVA KNJIGA?

Ova knjiga je vodič. U svakom poglavlju opisano je kako se primenjuju određeni elementi i tehnike Java jezika ili su prikazane uvodne napomene o Java EE tehnologijama koje se izvršavaju na strani servera. Odeljci „Probajte sami“ služe za doradu primera opisanih u poglavljima. U prpratnim video materijalima obično je ilustrovano kako se izvršavaju određeni zadaci koji su opisani u odeljcima „Probajte sami“.

Možete da odlučite da pročitate poglavlje, a zatim isprobate primere i uradite zadatke, ili da pročitate poglavlje, pregledate video materijal, a zatim probate da samostalno uradite zadatak.

Lekcije su kratke i uređene. Cilj je brzo opisivanje materijala, tako da možete da ga praktično upotrebite što pre. Određeni čitaoci mogu da pomisle da su neophodna dodatna objašnjenja za određene teme; podstičemo vas da samostalno nastavite istraživanje. Postoji velika količina mrežno dostupnih materijala o svim Java temama, ali će vam ono što je prikazano u ovoj knjizi definitivno pomoći da razumete na šta treba da obratite posebnu pažnju i šta da tražite dalje.

ŠTA JE NEOPHODNO DA BISTE KORISTILI OVU KNJIGU?

Da biste izvršavali primere i rešavali zadatke opisane u ovoj knjizi, neophodno je da nabavite bilo koji softver – besplatno raspoloživ softver je opisan u knjizi. Kako se instaliraju Java Development Kit i Eclipse Integrated Development Environment (IDE) opisano je u prva dva poglavlja, a to je sve što vam je neophodno da počnete rad. U Poglavlju 21 preuzetećete Derby DB, besplatan sistem za upravljanje bazama podataka. U Poglavlju 25 ćete instalirati Java Application Server GlassFish - on se koristi za opisivanje Java EE tehnologija koje se izvršavaju na strani servera, a opisane su u poglavljima 25 do 33. Konačno, u Poglavlju 36 instaliraćete Gradle,

savremeni alat za automatizovani razvoj, koji koriste profesionalni Java programeri. Kada je neophodno da se instalira određeni softver, date su detaljne instrukcije i/ili video uputstva.

Sa stanovišta hardvera, možete da koristite ili PC sa Windows operativnim sistemom ili Apple računare sa Mac OS X operativnim sistemom. Ljubitelji Linuxa takođe mogu da izvršavaju sve primere iz ove knjige. Neophodno je da imate bar 2 GB RAM memorije na svom računaru da biste startovali sve primere koda iz ove knjige. Dodatna memorija može da doprinese da vaš Java kompajler i Eclipse IDE funkcionišu mnogo brže.

KAKO ČITATI OVU KNJIGU?

Ova knjiga je vodič, a u svakoj lekciji se pretpostavlja da su vam već poznati materijali koji su opisani u prethodnim lekcijama. Ukoliko ste Java početnik, preporučujemo vam da čitate ovu knjigu sekvencijalno. Obično je prikazano malo teorije, koju prati kod koji možete ili čitati ili praktično isprobati.

Svaka lekcija, osim poslednje, ima prateći video materijal koji omogućava da vidite kako da rešite zadatak opisan u odeljku „Probajte sami“, da isprobate kod primera ili da jednostavno instalirate i konfigurirate neki softver. Idealno, treba da pokušate da samostalno uradite sve zadatke iz odeljaka „Probajte sami“ i pregledate video materijale samo kada ne možete da uradite ili ne razumete instrukcije. Međutim, ukoliko više volite da sledite instruktora, prvo pogledajte video, a zatim pokušajte da samostalno ponovite određene instrukcije. Ako to da rezultate, onda je u redu.

Java je višepplatformski jezik, a programi pisani za Microsoft Windows treba da rade na isti način na, recimo, Mac OS X ili Linux računarima. Ja koristim Mac računar, ali postoji i poseban softver koji mi omogućava da startujem Microsoft Windows. U ovoj knjizi koristi se softver sa otvorenim kodom, Eclipse integrisano razvojno okruženje koje postoji na svim najvećim platformama i izgleda gotovo identično na svakoj od njih. Zbog toga, nezavisno od toga koji operativni sistem vam je omiljen, moći ćete da isprobate sve primere koda prikazanog u ovoj knjizi.

KONVENCIJE

Da bismo vam pomogli da lakše pratite tekst i programe, u knjizi se poštuju odgovarajuće konvencije.

NAPOMENA Napomene, saveti, preporuke, trikovi i komentari na trenutnu diskusiju označeni su na poseban način i prikazani iskošenim slovima.

SAVET Reference, kao što je ova, ukazuju na URL adresu na kojoj se nalazi video materijal sa instrukcijama, koji se upotpunjuje odgovarajuća lekcija.

Sledeći stilovi primenjuju se u tekstu:

- *Naglašavaju* se novi pojmovi i značajne reči kada se prvi put pominju.
- Nazivi datoteka, URL adrese i kod u tekstu označavaju se na sledeći način: `persistence.properties`.
- Kod se prikazuje na sledeći način:

Koristi se monofont tip bez naglašavanja u većini primera koda.

IZVORNI KOD

Kada budete pregledali primere u knjizi, možete da se opredelite da li ćete da ukucavate ceo kod samostalno ili da koristite datoteke sa izvornim kodom za ovu knjigu. Sav izvorni kod prikazan u ovoj knjizi je raspoloživ za preuzimanje na stranici knjige na adresi www.wrox.com.

Kada pristupite strani na ovoj adresi da biste preuzeli izvorni kod iz knjige, jednostavno kliknite Download Code link na stranici sa detaljima o knjizi.

Nakon što preuzmete kod, samo otpakujte datoteku, koristeći svoj omiljeni arhiver. Alternativno, možete da pristupite glavnoj stranici za preuzimanje Wrox koda, koja se nalazi na adresi www.wrox.com/dynamic/books/download.aspx, a na kojoj je prikazan kod raspoloživ za ovu i sve ostale „Wrox“ knjige.

GREŠKE

„Wiley Publishing“ i „Wrox“ su uložili veliki napor da uklone sve greške iz teksta i koda. Međutim, niko nije savršen, pa se potkradaju greške. Ukoliko pronađete grešku u nekoj od naših knjiga, kao što su pravopisna greška ili neispravan deo koda, bićemo vam veoma zahvalni da nas obavestite o tome. Slanjem informacija o greškama možete da poštedite druge čitaoce frustracija, a u isto vreme ćete nama da pomognete da pružimo još kvalitetnije informacije.

Da biste pronašli stranicu sa pronađenim greškama u ovoj knjizi, pristupite www.wrox.com prezentaciji, a zatim locirajte knjigu, koristeći Search polje za pretraživanje ili jednu od listi sa naslovima knjiga. Nakon što pristupite stranici sa detaljima o knjizi, kliknite Book Errata link. Na toj stranici možete da pregledate sve greške koje su identifikovane u ovoj knjizi i koje su postavili „Wrox“ urednici. Celokupna lista sa linkovima na stranice sa greškama za sve knjige raspoloživa je i na adresi www.wrox.com/misc-pages/booklist.shtml.

Ukoliko ne identifikujete „vašu“ pronađenu grešku na Book Errata stranici, pristupite stranici www.wrox.com/contact/techsupport.shtml, a zatim ispunite formular i poljašite informacije koju grešku ste pronašli. Proverićemo podatke i, ukoliko su tačni, postaviti poruku na stranici sa greškama u ovoj knjizi, kao i rešenje problema u narednom izdanju ove knjige.

P2P.WROX.COM

Da biste diskutovali sa autorom knjige i drugim čitaocima, pridružite se P2P forumima na adresi p2p.wrox.com. Forumi su web sistemi koji omogućavaju da postavljate poruke koje se odnose na „Wrox“ knjige i povezane tehnologije i da interagujete sa drugim čitaocima i korisnicima tih teh-

nologija. Forumi pružaju mogućnost prijavljivanja za slanje novih informacija o interesantnim temama elektronskom poštom, odmah nakon što se pojave novi postovi na forumima. Na ovim forumima prisutni su „Wrox“ autori, urednici i drugi industrijski eksperti, kao i iskusni čitaoci.

Na adresi p2p.wrox.com možete pronaći veliki broj različitih foruma koji će vam pomoći ne samo u toku čitanja ove knjige, već i u toku razvoja vaših aplikacija. Da biste se pridružili forumima, neophodno je da uradite sledeće:

1. Pristupite p2p.wrox.com stranici i kliknite Register link.
2. Upoznajte pravila korišćenja i kliknite Agree.
3. Unesite neophodne informacije za pristupanje i opcione informacije koje želite da obezbedite, a zatim kliknite Submit.
4. Elektronskom poštom ćete primiti poruku sa informacijama kako da verifikujete svoj nalog i završite proces pristupanja.

SAVET Poruke na forumima možete da čitate i bez pristupanja P2P zajednici, ali, da biste postavljali poruke, morate da se registrujete.

Nakon registrovanja, možete da postavljate poruke i odgovarate na poruke koje su postavljali drugi korisnici. Možete da čitate poruke na Webu u bilo kom trenutku. Ukoliko želite da primate nove poruke iz nekog konkretnog foruma, tako što će vam biti prosledene elektronskom poštom, kliknite Subscribe to this Forum ikonu, koja se nalazi pored naziva foruma u listi foruma.

Više informacija o tome kako se koristi Wrox P2P možete pronaći u P2P uputstvu, koje sadrži odgovore na često postavljana pitanja (FAQ) kako funkcioniše softver foruma i brojna druga pitanja, specifična za P2P i „Wrox“ knjige. Da biste pristupili FAQ listi često postavljanih pitanja, kliknite FAW link na bilo kojoj P2P stranici.

3

Objektno-orijentisano programiranje pomoću Java jezika

Počevši od ovog poglavlja, pristupate učenju različitih elemenata Java jezika, uz kratak opis koji omogućava da započnete programiranje na najbrži mogući način. Međutim, svakako vas podstičemo da pogledate detaljniju Java SE dokumentaciju, koja je mrežno dostupna na adresi <http://docs.oracle.com/javase/8/>.

KLASE I OBJEKTI

Java je objektno-orijentisani jezik, što znači da ima konstrukcije za predstavljanje objekata iz realnog sveta. Svaki Java program sadrži bar jednu klasu, koja omogućava izvršavanje nečega i predstavljanje određenog tipa objekta. Na primer, najjednostavnija klasa `HelloWorld` omogućava prikazivanje pozdrava.

Klase u Java jeziku mogu da sadrže metode i polja (poznate i pod nazivom atributi). Metodi predstavljaju akcije ili funkcije koje klasa može da izvršava. Do Java 8 verzije, svaka funkcija je morala da bude prikazana kao metod određene klase. Lambda izrazi (videti odeljak „Rad sa tokovima“) omogućavaju veću slobodu prilikom korišćenja funkcija, ali je u ovom odeljku fokus na Java osnovama – klasama, metodima i poljima.

Kreirajmo i razmotrimo klasu `Car`. Ona će sadržati *metode*, pomoću kojih se opisuje ono što u određenom tipu vozila može da se uradi: startovanje i gašenje motora, ubrzavanje, kočenje, zaključavanje vrata i slično.

Ova klasa će imati i određena *polja*: boja kola, broj vrata, nalepnica sa cenom i drugo.

Listing 3-1 Klasa Car

```
class Car{  
    String color;
```

```
int numberOfDoors;

void startEngine() {
    // Određeni kod unosi se ovde
}

void stopEngine() {
    int tempCounter=0;
    // Određeni kod unosi se ovde
}
}
```

U nekim primerima koda videćete komentare oblika „Određeni kod unosi se ovde“. To je urađeno zato da vam neki dodatni kod ne odvraća pažnju od onoga što je relevantno za određenu temu. U ovom trenutku ne morate da brinete o algoritmu za startovanje motora. Upoznaćete strukturu Java klase.

Car klasa definiše zajednička svojstva različitih automobila: svi automobili imaju atribute, kao što su boja ili broj vrata, a svi omogućavaju izvršavanje određenih akcija. Možete da budete specifičniji, pa da kreirate drugu Java klasu pod nazivom `JamesBondCar`. To je i dalje automobil, ali sa određenim atributima koji su specifični za model kreiran za Džeјmsa Bonda (videti listing 3-2). Možete da kažete da je klasa `JamesBondCar` *potklasa* Car klase, odnosno, koristeći Java sintaksu, `JamesBondCar` klasa „nasleduje“ (eng. `extends`) Car klasu.

Listing 3-2 Klasa `JamesBondCar`

```
class JamesBondCar extends Car{
    int currentSubmergeDepth;
    boolean isGunOnBoard=true;
    final String MANUFACTURER;

    void submerge() {
        currentSubmergeDepth = 50;
        // Određeni kod unosi se ovde
    }

    void surface() {
        // Određeni kod unosi se ovde
    }
}
```

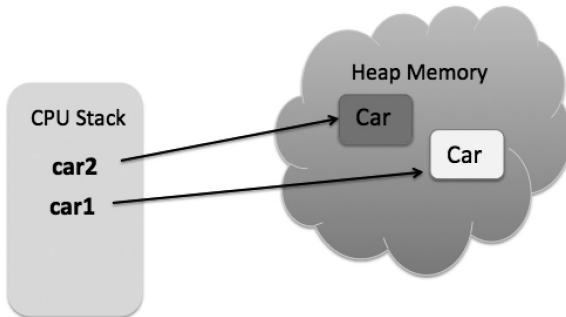
Kao što možete da naslutite na osnovu naziva metoda, automobil Džeјmsa Bonda ne samo da se vozi po zemlji, već i ispod vode, a zatim ponovo izranja na površinu. Međutim, čak i nakon definisnja svih atributa i metoda, `JamesBondCar` klasu ne možete da odmah koristite. Java klasa je nalik nekoj vrsti šablona u građevinarstvu ili inženjerstvu; dok ne kreirate realne objekte na osnovu šablona, ne možete ih koristiti.

Kreiranje objekata, poznatih kao instance, na osnovu klase ekvivalentno je kreiranju realnih automobila na osnovu šablona. Kreiranje instance klase predstavlja kreiranje objekta u memoriji računara na osnovu definicije klase.

Da biste instancirali klasu („da biste izveli automobil na put“), deklarirate promenljivu određenog tipa klase, a zatim koristite `new` operator za svaku novu instancu klase:

```
JamesBondCar car1 = new JamesBondCar();
JamesBondCar car2 = new JamesBondCar();
```

Sada promenljive `car1` i `car2` mogu da se koriste za označavanje prve i druge instance `JamesBondCar` klase, respektivno. Da budemo precizniji: deklarisanje promenljivih koje ukazuju na instance neophodno je ukoliko planirate da koristite ove instance u programu. Promenljive `car1` i `car2` postaju vaše pristupne tačke za odgovarajuće instance klase `Car`, kao što je prikazano na slici 3-1.



Slika 3-1 Instanciranje dva objekta `Car` klase

Naredbom `new JamesBondCar()` kreira se nova instanca ove klase u *hip* memoriji. U realnom svetu možete da kreirate veliki broj automobila na osnovu iste specifikacije. Iako svi oni predstavljaju jednu istu klasu, mogu da se razlikuju u određenim atributima – neki od njih su crveni, a drugi žuti, neki imaju dvoja vrata, a drugi četvora i tako dalje.

PROMENLJIVE I TIPOVI PODATAKA

Određene vrednosti koje predstavljaju objekat mogu se menjati u toku izvršavanja programa (promenljive), a neke druge ostaju iste (konstante). U ovom odeljku je detaljnije opisano korišćenje oba tipa.

Deklarisanje promenljivih

Java je statički tipizirani jezik: *promenljiva* u programu mora da bude deklarirana (da joj se dodeli naziv i tip podataka), pa se tek onda mogu dodeljivati vrednosti u vreme deklarisanja, ili kasnije, u nekom od metoda klase. Na primer, promenljiva `isGunOnBoard` je inicijalizovana u toku deklarisanja u listingu 3-2, a `currentSubmergeDepth` promenljiva je dobila vrednost u `submerge()` metodi.

Klasa `Car` iz listinga 33-1 definiše `color` promenljivu tipa `String`, koja se koristi za prikazivanje tekstualnih vrednosti; na primer, „crvena“, „plava“ i slično.

Final promenljive

Da bi bile čuvane vrednosti koje se nikada ne menjaju, neophodno je deklarirati `final` promenljivu (ili konstantu); samo dodajte ključnu reč `final` u deklaracionu liniju, kao što je prikazano u listingu 3-2.

```
final String MANUFACTURER = „J.B. Limited“;
```

Java programeri nazive `final` promenljivih pišu velikim slovima. Ukoliko se pitate kako se Java programeri dogovaraju o konvencijama za davanje naziva, proverite određene vodiče koji opisuju standarde kodiranja. Na primer, Google objavljuje standarde kodiranja na stranici <https://code.google.com/p/googlestyleguide/> za različite jezike.

Vrednost konstante može da se dodeli samo jednom, a pošto kreirate instancu specifičnog automobila, njegov proizvođač je poznat i ne može se menjati tokom „životnog veka“ ovog objekta. Deklarirate promenljivu `final` i odmah je inicijalizujete, kao što je prethodno prikazano.

Osnovni tipovi podataka

Prilikom deklarisanja klase kreirate novi tip podataka i možete da deklarirate promenljive ovog tipa, kao što ste imali priliku da vidite u primeru za `Car` klasu. Međutim, to nisu jednostavni tipovi podataka, jer uključuju polja i metode koji opisuju objekat ovog tipa. Sa druge strane, Java sadrži unapred definisane tipove podataka za skladištenje jednostavnih vrednosti, kao što su ceo broj ili karakter.

Postoji osam *osnovnih* tipova podataka u jeziku Java: četiri se odnose na celobrojne vrednosti, dva su vrednosti sa decimalnim zarezom, jedan je za smeštanje pojedinačnih karaktera i jedan je za logičke (bulove) podatke, koji mogu imati vrednost `true` ili `false`. Sledi prikaz primera deklaracija i inicijalizacija promenljivih:

```
int chairs = 12;
char grade = 'A';
boolean cancelJob = false;
double nationalIncome = 23863494965745.78;
float hourlyRate = 12.50f; // dodajte slovo f na kraj
                           //float literals
long totalCars = 46372836483921; // dodajte slovo l na kraj
// of long literals
```

Poslednja dva literala u prethodnoj listi završavaju slovima `f` i `l` da bi se ukazalo da se želi smeštanje podataka korišćenjem `float` i `long` tipa podataka, respektivno. Tip podataka `double` zadovoljava potrebe u većini izračunavanja koja zahtevaju realne brojeve.

Svaki primitivni tip podataka zauzima određenu količinu memorije i ima opseg vrednosti koje može da sadrži. Sledeća tabela sadrži određene karakteristike Java tipova podataka.

OSNOVNI TIP	VELIČINA	MINIMUM	MAKSIMUM	KLASA
<code>byte</code>	8 bitova	-128	127	<code>Byte</code>
<code>short</code>	16 bitova	-32768	32767	<code>Short</code>
<code>int</code>	32 bita	-2 147 483 648	2 147 483 647	<code>Integer</code>
<code>long</code>	64 bita	-9 223 372 036 854 775 808	9 223 372 036 854 775 807	<code>Long</code>

OSNOVNI TIP	VELIČINA	MINIMUM	MAKSIMUM	KLASA
float	32 bita	Realan broj obične tačnosti; videti Java specifikaciju na http://bit.ly/9nlwjh Realan broj obične tačnosti; videti Java specifikaciju na http://bit.ly/9nlwjh	Float	
double	64 bita		Realan broj duple tačnosti; videti Java specifikaciju na http://bit.ly/9nlwjh Realan broj duple tačnosti; videti Java specifikaciju na http://bit.ly/9nlwjh	Double
char	16 bitova	Unicode 0	Unicode 2 na 16. vrednost	Character
boolean	-	false (nije minimum)	true (nije maksimum)	Boolean

Da li ste primetili da `char` tip podataka koristi dva bajta memorije za skladištenje podataka? To omogućava da skladištite skupove karaktera koji sadrže mnogo više simbola od tradicionalnih alfabeta, pošto jedan bajt može da prikaže samo 256 različitih karaktera, dok dva bajta mogu da predstavljaju 65.536 karaktera.

Ukoliko je neophodno da skladištite veoma velike brojeve, imajte u vidu da Java sadrži `BigDecimal` klasu, ali to nije osnovni tip podataka.

DOMET PROMENLJIVIH

Ukoliko deklarišete promenljivu unutar metoda ili bloka koda ograđenog velikim zagradama, promenljiva je lokalnog dometa (na primer, `tempCounter` promenljiva u listingu 3-1 je lokalna). To znači da će promenljiva biti vidljiva samo u kodu unutar `stopEngine()` metoda. Lokalna promenljiva se može koristiti u metodu samo nakon deklarisanja promenljive, i to jedino u bloku u kome je deklarisan. Na primer, promenljiva deklarisan unutar `for` petlje ne može se koristiti izvan petlje, čak i u istom metodu.

Kada metod okonča izvršavanje, sve lokalno definisane promenljive osnovnih tipova automatski se uklanjaju iz stek memorije. Ukoliko je promenljiva ukazivala na instancu nekog objekta (na primer, `car1` na slici 3-1), *Garbage Collector* (GC) za Java jezik uklanja odgovarajuću instancu objekta iz hip memorije, ali se to ne dešava odmah. GC periodično pristupa hip memoriji i uklanja sve objekte koji ne referenciraju promenljive.

Ukoliko nekoj promenljivoj mora da se pristupa iz više metoda klase, deklariše se na nivou klase. U listingu 3-1 prikazana je klasa `Car`, u kojoj su `color` i `numberOfDoors` promenljive definisane na nivou klase, odnosno to su *promenljive članice* klase. Ove promenljive postoje sve dok postoji instanca `Car` objekta u memoriji. Njih mogu da dele i koriste više puta metodi klase, a mogu čak biti vidljive i iz eksternih klasa (o nivoima pristupa biće više reči u Poglavlju 7). Postoje određene razlike u prosleđivanju osnovnih promenljivih i onih koje ukazuju na instance objekta. Pročitajte odeljak „Prosleđivanje po vrednosti ili referenci“, koji se nalazi u sledećem poglavlju.

NAPOMENA Ukoliko je promenljiva deklarirana korišćenjem `static` kvalifikatora (videti Poglavlje 4), deliće je sve instance određene klase. Promenljive instance (bez ključne reči `static`) skladište različite vrednosti za svaki objekat, odnosno instancu.

KLASE OMOTAČI I AUTOMATSKO KONVERTOVANJE U OBJEKTE I OSNOVNE TIPOVE

Svi osnovni tipovi podataka imaju odgovarajuće klase omotače (eng. wrapper classes), koje sadrže korisne metode za rad sa odgovarajućim tipovima podataka. Ove klase imaju dve svrhe:

1. Sadrže određeni broj korisnih funkcija za rad sa osnovnim tipovima podataka. Na primer, klasa `Integer` sadrži korisne metode, kao što su konverzija `String` tipa u `int` vrednost, pretvaranje `int` vrednosti u `float` vrednost i druge. `Integer` klasa omogućava i da definišete minimalne i maksimalne vrednosti određenog tipa.
2. Neke Java kolekcije ne mogu da skladište vrednosti osnovnih tipova podataka (kao što je `ArrayList`), tako da osnovni tipovi moraju da budu prevedeni u objekte - na primer:

```
ArrayList myLotteryNumbers = new ArrayList();
myLotteryNumbers.add(new Integer(6));
myLotteryNumbers.add(new Integer(15));
```

Java jezik ima mogućnost automatskog konvertovanja u objekte (eng. autoboxing), pa se automatski kreira nova instanca za svaki osnovni tip podataka. Jednostavno, možete da napišete `myLotteryNumbers.add(6)` i vrednost osnovnog tipa `6` automatski se konvertuje u instancu `Integer` klase.

Shodno tome, i sledeća linija je validna:

```
int luckyNumber= myLotteryNumber.get(23);
```

Iako `get(23)` vraća vrednost 24. elementa (budući da brojanje u Java kolekcijama počinje od nule) kao `Integer` objekat; taj objekat se automatski konvertuje u vrednost osnovnog tipa podataka. To je označeno kao otpakivanje (eng. *unboxing*).

KOMENTARI U PROGRAMU

U toku pisanja Java programa treba da dodajete komentare, koji predstavljaju tekst sa objašnjenjima šta program radi. Programi se mnogo češće čitaju, nego što se pišu. Osim toga, drugi programeri će čitati i pokušavati da razumeju vaš kod. Budite predusretljivi i učinite im posao jednostavnijim. Tipičan programer ne voli da piše komentare (nezavisno od toga koji programski jezik koristi).

Predlažemo sledeću jednostavnu tehniku: pišite prvo komentare, pa tek onda kod. Kada napišete program, on će već sadržati komentare. Možete da pišete komentare bilo gde u kodu – pre ili u telu klase, ili u telu metoda.

U Java programskom jeziku postoje tri tipa komentara:

- Blok komentari sadrže više od jedne linije teksta, koje se pišu između simbola `/*` i `*/`. Na primer:

```

/* This method will calculate the cost of shipping, handling,
and all applicable taxes
*/

```

Kompajler ignoriše tekst u komentarima, tako da možete da pišete šta god želite.

- Ukoliko želite da pišete kratak komentar, koji se uklapa u jednu liniju, počnite ovu liniju sa dve kose crte (//). Možete da postavite komentare iza dve kose crte i na kraju linije koda.

```

// Calculate the cost of shipping
int cost = calcShippingCost(); // results depends on country

```

Određeni komentari počinju sa `/**` i završavaju se sa `*/`. Njih koristi specijalni alat javadoc prilikom automatskog ekstrahovanja teksta iz komentara i kreiranja dokumentacije programa. Javadoc alat omogućava i korišćenje specijalne notacije (na primer, `@param`, `@return`, `@see`) koja omogućava kreiranje dokumentacije programa profesionalnog izgleda. Da biste saznali šta javadoc može da generiše, pročitajte na adresi <http://goo.gl/imDMU> tehničko uputstvo kompanije „Oracle“ o pisanju javadoc komentara.

Vaš prvi koristan program

Vreme je da napišete program koji radi nešto mnogo korisnije od prikazivanja poruke „Hello World“. Ovaj program izračunava vrednost poreza. Cilj je prikazivanje načina na koji Java klase komuniciraju, kako se pozivaju metodi i kako se mogu koristiti promenljive.

Prvo je neophodno da se opredelite koje Java klase treba da kreirate za rešavanje problema. Zatim, razmislite o atributima (promenljivim u klasama) i metodima (ponašanju) koje ove klase treba da imaju.

Deklarisanje Tax klase

Pošto planirate da izračunavate vrednost poreza, nije potrebno da budete naučnik da biste utvrdili da je potrebno da kreirate klasu `Tax`. Počnite nazivom klase i velikim zagradama – to je najjednostavnija klasa koju možete da kreirate:

```

class Tax{
}

```

Šta je neophodno da bi ova klasa mogla da određuje vrednost poreza? Definitivno morate da znate koliki je prihod osobe kojoj određujete godišnji porez. Ukupan prihod je dobar „kandidat“ za jedan atribut klase. Atributi u Java jeziku su predstavljeni promenljivim. Izaberite jedan od numeričkih tipova podataka. Ukupan prihod nije uvek ceo broj, tako da možete da koristite `double` tip podataka, jer je reč o broju sa decimalama. Možete da koristite `float` umesto ovog tipa, ali korišćenje `double` tipa omogućava da budete spremni da određujete vrednost i za veće prihode.

```

class Tax{
    double grossIncome;
}

```

Osim toga, neophodno je da znate u kom statusu je osoba; pravila oporezivanja su različita u različitim državama SAD. Sledi nekoliko skraćenica država u SAD: NY, NJ, CT. Upotrebite `String` tip podataka za skladištenje tekstualnih podataka:

```
class Tax{
    double grossIncome;
    String state;
}
```

Dodajte još jedan atribut za izdržavana lica za osobu čiji porez računate. Celobrojni tip (integer) biće odgovarajući – nije moguće da neko izdržava dva i po lica:

```
class Tax{
    double grossIncome;
    String state;
    int dependents;
}
```

Dodavanje metoda u Tax klasu

Promenljive skladište podatke, dok metodi izvršavaju akcije. Vreme je za akcije. Prvi metod `calcTax()` određuje vrednost poreza na osnovu vrednosti ukupnog prohoda, broja izdržavanih lica i države:

Listing 3-3 Tax klasa

```
class Tax{
    double grossIncome;
    String state;
    int dependents;
    public double calcTax() {

        return 234.55;
    }
}
```

Specifikacija `calcTax()` metoda definiše sledeće:

- Bilo koja eksterna klasa može da pristupa ovom metodu (`public`).
- Ovaj metod vraća vrednost tipa `double`.
- Naziv metoda je `calcTax`.

Prazne zagrade nakon naziva metoda ukazuju da metod nema argumente, ili, drugim rečima, nije neophodna ni jedna vrednost izvan `Tax` klase da bi bila izvršena izračunavanja. Zapravo, ova verzija `calcTax()` metoda ne koristi čak ni vrednosti iz promenljivih klase za izračunavanje poreza. Uvek vraća vrednost 234,55, koja je uneta u sam kod.

Kako da procenite da li metod treba da vrati vrednost? Ukoliko vaš metod izvršava određena izračunavanja i mora da vrati određenu vrednost u program iz koga se poziva, tada će imati povratnu vrednost. Ukoliko metod direktno modifikuje promenljive klase ili jednostavno negde prikazuje podatke (monitor, disk, server), ne mora da vraća nikakvu vrednost. Vi i dalje morate da deklarirate da „ne vraća vrednost“ u specifikaciji metoda, koristeći posebnu ključnu reč `void`:

```
public void printAnnualTaxReturn() {
    //Određeni kod unosi se ovde
}
```

Koristeći `return` naredbu Java jezika, metod može da vrati programu iz koga se poziva podatke koji se nalaze u promenljivoj - na primer:

```
return calculatedTax;
```

Ako deklarirate povratni tip u specifikaciji metoda, ali zaboravite da ukucate `return` naredbu u telu metoda, Java kompajler prijavljuje grešku.

Deklarisanje druge klase `TestTax`

`Tax` klasa omogućava određivanje načina na koji se računa porez, ali u realnim aplikacijama imaćete brojne klase koje predstavljaju različite tokove podataka u ovom procesu. Na primer, možda je neophodno da kreirate klasu `Customer`. Zavisno od tipa zaposlenja ili prihoda, računovode koriste brojne druge različite forme za određivanje poreza, a svaka forma može da se prikaže posebnom klasom: `Form1040`, `Form1099` i tako redom.

Svaka od ovih klasa predstavlja određeni entitet, ali ni jedna nije izvršni program; to znači da ni jedna neće sadržati `main()` metod. Neophodno je da kreirate još jednu klasu za startovanje aplikacije i kreiranje instanci drugih klasa. Nju ćemo nazvati `TestTax`.

Klasa `TestTax` treba da ima mogućnost izvršavanja sledećih akcija:

- Kreiranje instance `Tax` klase
- Dodeljivanje podataka korisnika (ukupan prihod, država, izdržavana lica) promenljivim klase `Tax`.
- Izvršavanje metoda `calcTax()`
- Prikazivanje rezultata izvršavanja na ekranu

Klasa `TestTax` je smeštena u posebnu datoteku pod nazivom `TestTax.java`.

Listing 3-4 Klasa `TestTax`

```
class TestTax{
    public static void main(String[] args){
        Tax t = new Tax(); // kreiranje instance

        // dodeljivanje vrednosti članovima klase
        t.grossIncome= 50000;
        t.dependents= 2;
        t.state= „NJ“;

        double yourTax = t.calcTax(); //određivanje poreza

        // Prikazivanje rezultata
```

```
        System.out.println(„Your tax is “ + yourTax);
    }
}
```

U prethodnom kodu deklarirali ste promenljivu `t` tipa `Tax`. Metod `main()` je početna tačka programa za izračunavanje poreza. Ovaj metod kreira instancu klase `Tax`, a promenljiva `t` ukazuje na mesto u memoriji vašeg računara gde je kreiran `Tax` objekat. Od tog trenutka, ukoliko želite da referencirate ovaj objekat, koristite promenljivu `t`. Pogledajte još jednom sliku 3-1 – na njoj je prikazana slična situacija kao ona u prethodnom kodu.

Sledeće tri linije omogućavaju dodeljivanje vrednosti poljima `Tax` objekta:

```
t.grossIncome= 50000;
t.dependents= 2;
t.state= „NJ“;
```

Nakon toga, možete da izračunate porez za objekat prikazan promenljivom `t` izvršavanjem metoda `calcTax()`, a rezultat izvršavanja ovog metoda biće dodeljen promenljivoj `yourTax`. Metod `calcTax()` i dalje vraća vrednost definisanu u kodu, ali to ćete promeniti u odeljku „Probajte sami“ u ovoj lekciji. Poslednja linija služi za prikazivanje rezultata u sistemskoj konzoli.

U ovom trenutku imate dve klase koje međusobno komuniciraju - `TestTax` i `Tax`. Klasa `TestTax` kreira instancu `Tax` klase, inicijalizuje njene promenljive, a zatim izvršava njen metod

`calcTax()`, koji vraća vrednost klasi `TextTax`.

Uslovna naredba if

Java jezik sadrži `if` naredbu koja utvrđuje da li je određeni uslov ispunjen ili nije. Na osnovu ispunjenosti uslova određuje se redosled izvršavanja programa.

U sledećem kodu, ukoliko uslovni izraz (`totalOrderPrice > 100`) ima vrednost `true` (tačno, istinito), izvršava se kod između velikih zagrada; u suprotnom, izvršava se kod koji se nalazi iza `else` naredbe:

```
if (totalOrderPrice > 100){
    System.out.println(„You'll get a 20% discount“);
}
else{
    System.out.println(„Order books for more than a“ +
    „ $100 to get a 20% discount“);
}
```

Pošto ovaj kod sadrži samo jednu naredbu u `if` i `else` klauzulama, navođenje velikih zagrada nije neophodno, ali one čine kod preglednijim i sprečavaju pojavu bagova koji se teško otkrivaju ukoliko kasnije morate da dodate još koda u `if` naredbi.

Naredba switch

Naredba switch je alternativa za if. Obeležje case iza switch uslova (taxCode) se proverava i program nastavlja izvršavanje jedne od sledećih case klauzula:

```
int taxCode=someObject.getTaxCode(grossIncome);
switch (taxCode){
    case 0:
        System.out.println(„Tax Exempt“);
        break;
    case 1:
        System.out.println(„Low Tax Bracket“);
        break;
    case 2:
        System.out.println(„High Tax Bracket“);
        break;
    default:
        System.out.println(„Wrong Tax Bracket“);
}
// Određeni kod unosi se ovde
```

Prethodni kod izvršava samo jedan od println() metoda, a zatim nastavlja izvršavanje koda koji se nalazi iza zatvorene velike zagrade (ukoliko on postoji). Ne zaboravite da postavite naredbu break na kraju svake case naredbe, tako da program nakon obrade case naredbe nastavlja izvršavanje iza switch naredbe; u suprotnom, nastavlja se izvršavanje koda i štampa se više od jedne linije, čak i kada taxCode može da ima samo jednu vrednost. Na primer, sledeći kod štampa „Tax Exempt“ i „Low Tax Bracket“ tekst, čak i kada taxCode promenljiva ima vrednost nula:

```
switch (taxCode){
case 0:
    System.out.println(„Tax Exempt“);
case 1:
    System.out.println(„Low Tax Bracket“);
    break;
case 2:
    System.out.println(„High Tax Bracket“);
    break;
default:
    System.out.println(„Wrong Tax Bracket“);
}
```

Počevši od Java 7 verzije, možete da koristite String vrednost u case izrazu:

```
switch (yourState){
case „NY“:
    System.out.println(„Taxing by NY law“);
    break;
case „CA“:
```

```
System.out.println(„Taxing by CA law“);
break;
case „FL“:
    System.out.println(„Taxing by FL law“);
break;
default:
    System.out.println(„Wrong state“);
```

„Nasleđivanje“

U objektno-orijentisanim jezicima „nasleđivanje“ podrazumeva mogućnost da se nova klasa definiše na osnovu postojeće (ne od početka).

Zamislite da klasa `Tax` određuje `tax` svojstvo za sve, osim za državu New Jersey, u kojoj su uvedene nove obrazovne poreske olakšice. Ukoliko imate dete na koledžu, možete da dobijete dodatnu poresku olakšicu od 500 dolara. U ovom slučaju neophodno je ili da promenite `calcTax()` metod u `Tax` klasi da biste uveli specijalni slučaj za New Jersey ili da kreirate novu klasu na osnovu `Tax` klase i dodate ovu novu funkcionalnost u nju.

Svaka osoba nasleđuje određene karakteristike od svojih roditelja. Sličan mehanizam postoji i u Java programskom jeziku. Specijalna ključna reč `extends` se koristi za ukazivanje da je neka klasa izvedena iz druge klase:

```
class NJTax extends Tax{
}
```

Klasa `NJTax` ima sve karakteristike `Tax` klase, a vi možete da dodate nove attribute i metode u nju. Pri ovakvoj definiciji, `Tax` klasa je *superklasa* (natklasa), dok je `NJTax` potklasa (izvedena klasa). Mogu se primenjivati i pojmovi „predak“ i „potomak“, respektivno. Ova nova klasa ima pristup svim promenljivim i metodima natklase, ukoliko nemaju privatni nivo pristupa ili nivo pristupa na nivou paketa, o čemu će biti reči u Poglavlju 5.

Proširimo funkcionalnost klase `Tax` u `NJTax` klasi. Druga klasa sadrži `adjustForStudents()` metod:

Listing 3-5 Klasa `NJTax`

```
class NJTax extends Tax{

    double adjustForStudents (double stateTax){
        double adjustedTax = stateTax - 500;
        return adjustedTax;
    }
}
```

Da biste koristili ovaj novi metod, `TestTax` klasa treba da instancira `NJTax` klasu, umesto `Tax` klase, kao što je to prikazano u listingu 3-4.

```
NJTax t= new NJTax();
```

Sada možete da izvršite metode koji su definisani u `Tax` klasi, kao i one iz `NJTax` klase, koristeći `t` promenljivu - na primer:

```
NJTax t= new NJTax();
double yourTax = t.calcTax();
double totalTax = t.adjustForStudents(yourTax);
```

Dodali smo novu funkcionalnost u program za određivanje poreza bez promene koda `Tax` klase. U prethodnom delu koda pokazano je i kako možete da prosledite rezultate iz jednog metoda u drugi. Vrednost promenljive `yourTax` je izračunata u `calcTax()` metodi, a zatim prosledena u `adjustForStudents()` metod kao argument.

Predefinisane metode

Još jedan značajan pojam u objektno-orientisanom programiranju je predefinisane metode. Zamislite klasu `Tax` koja sadrži 20 metoda - većina ih je primenljiva za sve države, ali postoji jedan metod koji nije validan za New Jersey. Umesto da modifikujete ovaj metod u natklasi, možete da kreirate drugi metod u izvedenoj klasi, *sa istim nazivom i listom argumenata* (to se naziva i specifikacija). Ukoliko izvedena klasa sadrži metod sa istom specifikacijom, predefiniše (potiskuje) odgovarajući metod natklase.

Predefinisane metode postaje korisno u sledećim situacijama:

- Izvorni kod natklase nije raspoloživ, ali je i dalje neophodna promena funkcionalnosti.
- Originalna verzija metoda je i dalje validna u nekim situacijama, ali ne želite da metod menjate.
- Koristite predefinisane metode da biste omogućili polimorfizam, što će biti opisano u Poglavlju 7.

Imaćete priliku da predefinisane metode isprobate u odeljku „Probajte sami“. U Poglavlju 4 biće reči o preopterećenju metoda, koje se razlikuje od predefinisanja.

DODATNI MATERIJALI

Osnove o uklanjanju objekata u Java jeziku možete pronaći na stranici na adresi <http://www.oracle.com/webfolder/technetwork/tutorials/obe/java/gc01/index.html>.

PROBAJTE SAMI

Kreirajte pomoću Eclipse alata aplikaciju za određivanje poreza opisanu u ovoj lekciji, a zatim je modifikujte tako da zamenite vrednost unetu u kodu `calcTax()` metodom sa određenim izračunavanjima. Nakon što to uradite, na osnovu `Tax` klase napravite izvedenu klasu i predefinišite `calcTax()` metod.

Zahtevi

Za potrebe ove lekcije morate da imate instalirano Eclipse integrisano razvojno okruženje.

NAPOMENA Možete da preuzmete kod i resurse za ovu vežbu „Probajte sami“ na web stranici knjige na adresi www.wrox.com/go/javaprogramming24hr2e. Materijali se nalaze u Lesson3.zip datoteci.

Savet

Ova lekcija sadrži samo osnovne instrukcije o Java jezičkim konstrukcijama. Mrežno dostupno uputstvo može da bude od koristi prilikom izvršavanja ovog i budućih zadataka. Ono je raspoloživo na stranici na adresi <http://download.oracle.com/javase/tutorial/java/index.html>.

Korak po korak

1. U Eclipse alatu kreirajte novi projekat pod nazivom Lesson 3.
2. Kreirajte novu `Tax` klasu (File→New→Class). Unesite kod prikazan u listingu 3-3.
3. Kreirajte drugu klasu `TestTax` i unesite kod prikazan u listingu 3-4.
4. Snimite ove klase, pa startujte `TestTax` (kliknite desnim tasterom miša, pa selektujte Run As→Java Application). U konzoli treba da bude prikazan tekst „Your tax is \$234.55“.
5. Zamenite vrednost postavljenu u kod određenim proračunom poreza. Recimo, ukoliko je ukupan prihod manji od 30.000 dolara, državni porez je 5 odsto. Ukoliko je ukupan prihod veći od 30.000 dolara, za porez se izdvaja 6 odsto. Modifikujte kod `calcTax` metoda na sledeći način: startujte program nekoliko puta, modifikujući vrednosti promenljivih `Tax` klase. Proverite da li je u konzoli prikazana vrednost poreza ispravno određena:

```
public double calcTax() {
    double stateTax=0;
    if (grossIncome < 30000) {
        stateTax=grossIncome*0.05;
    }
    else{
        stateTax= grossIncome*0.06;
    }
    return stateTax;
}
```

6. Kreirajte `NJTax` klasu prikazanu u listingu 3-5.
7. Promenite funkcionalnost `calcTax()` metoda u `NJTax` klasi. Nova verzija `calcTax()` metoda treba da umanji porez za 500 dolara pre prikazivanja vrednosti.
8. Modifikujte kod `TestTax` klase da biste instancirali `NJTax` klasu, umesto `Tax` klase. Pratite da li je poreski odbitak od 500 dolara ispravno izračunat.

Da biste dobili datoteke probne baze, možete da preuzmete Chapter 3 fajl sa web stranice knjige na adresi www.wrox.com/go/javaprogram24hr2e.

SAUET Selektujte video materijale posvećene Poglavlju 3 na stranici na adresi www.wrox.com/go/javaprogram24hr2e. Moći ćete da sa te web stranice preuzmete izvorni kod i resurse posvećene ovoj lekciji.
