Pridružite se diskusiji na p2p.wrox.com

PREVOD DRUGOG IZDANJA

Java® 8 PROGRAMIRANJE

Yakov Fain

kompjuter biblioteka

Java® 8

PROGRAMIRANJE

PREVOD DRUGOG IZDANJA

Yakov Fain

Wiley Publishing, Inc.

Izdavač:

kompjuter biblioteka

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 Janaćković

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

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 Janaćković]. - 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

Sadržaj

LEKCIJA 3

LEKCIJA 4

LEKCIJA 5

LEKCIJA 11

LEKCIJA 12

LEKCIJA 13

LEKCIJA 14

LEKCIJA 15

LEKCIJA 16

LEKCIJA 17

LEKCIJA 22

LEKCIJA 23

LEKCIJA 24

LEKCIJA 25

LEKCIJA 26

LEKCIJA 27

LEKCIJA 33

LEKCIJA 34

LEKCIJA 35

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 promoviše 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 intervjue 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 intervjue za Java tehničke poslove svih onih koji su zainteresovani da se prijave za posao Java programera.

KAKO JE STRUKTUIRANA 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 propratnim 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 mateijral, 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 preuzećete 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 konfigurišete 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 višeplatformski 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 čitoacima 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 adesi 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 prosleđene 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 Džejmsa Bonda (videti listing 3-2). Možete da kažete da je klasa `JamesBondCar` *potklasa* `Car` klase, odnosno, koristeći Java sintaksu, `JamesBondCar` klasa „nasleđuje“ (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
        }
}
```

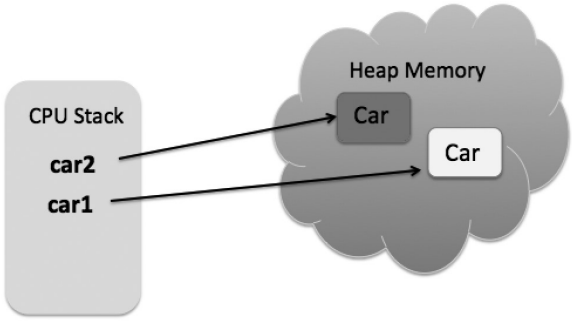
Kao što možete da naslutite na osnovu naziva metoda, automobil Džejmsa Bonda ne samo da se vozi po zemlji, već i ispod vode, a zatim ponovo izranja na površinu. Međutim, čak i nakon definisnja svih atributa i metoda, `JamesBondCar` klasu ne možete da odmah koristite. Java klasa je nalik nekoj vrsti šablona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

Kreiranje objekata, poznatih kao instance, na osnovu klasa 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“), deklarišete 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 predstavaju 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 deklarisana (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()` metodu.

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 deklarisati `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. Deklarišite promenljivu `final` i odmah je inicijalizujte, kao što je prethodno prikazano.

Osnovni tipovi podataka

Prilikom deklarisanja klase kreirate novi tip podataka i možete da deklarišete 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
`byte`	8 bitova	-128	127	`Byte`
`short`	16 bitova	-32768	32767	`Short`
`int`	32 bita	-2 147 483 648	2 147 483 647	`Integer`
`long`	64 bita	-9 223 372 036 854 775 808	9 223 372 036 854 775 807	`Long`

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 skladište 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 predstave 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 deklarisana. Na primer, promenljiva deklarisana unutar for petlje ne može se koristiti izvan for 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 deklarisana 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 korisni 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 deklarišete 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 deklarišete 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, računovođe 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(); // kreranje 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 deklarisali 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 `TextTax` 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 atribute 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()` metodu, a zatim prosleđena u `adjustForStudents()` metod kao argument.

Predefinisanje metoda

Još jedan značajan pojam u objektno-orijentisanom programiranju je predefinisanje metoda. 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.

Predefinisanje metoda 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 predefinisanje metoda da biste omogućili polimorfizam, što će biti opisano u Poglavlju 7.

Imaćete priliku da predefinisanje metoda 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/javaprog24hr2e. 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/javaprog24hr2e.

SAVET Selektujte video materijale posvećene Poglavlju 3 na stranici na adresi www.wrox.com/go/javaprog24hr2e. Moći ćete da sa te web stranice preuzmete izvorni kod i resurse posvećene ovoj lekciji.