Rogers Cadenhead

OSMO IZDANJE

Pokriva **Java 11 i 12**

NAUČITE SAMOSTALNO

Java™

za **21 DAN**

kompjuter biblioteka

SAMS

Rogers Cadenhead

NAUČITE SAMOSTALNO

Java™

PREVOD VIII IZDANJA

za **21 DAN**

kompjuter biblioteka

Izdavač:

kompjuter biblioteka

Obalskih radnika 4a, Beograd

Tel: 011/2520272

e-mail: kombib@gmail.com

internet: www.kombib.rs

Urednik: Mihailo J. Šolajić

Za izdavača, direktor:
Mihailo J. Šolajić

Autor: Rogers Cadenhead

Prevod: Biljana Tešić

Lektura: Miloš Jevtović

Slog : Zvonko Aleksić

Znak Kompjuter biblioteke:
Miloš Milosavljević

Štampa: „Pekograf“, Zemun

Tiraž: 500

Godina izdanja: 2020.

Broj knjige: 525

Izdanje: Prvo

ISBN: 978-86-7310-548-2

Sams Teach Yourself Java in 21 Days, Eighth Edition

by Rogers Cadenhead
ISBN: 978-0-672-33795-6

Uvod

Neke revolucije su iznenadile svet. Na primer, Slack, Linux, Popajev sendvič sa piletinom i muzičko takmičenje „*The Masked Singer*“ su neočekivano postali poznati.

Međutim, izuzetan uspeh Java programskog jezika nikoga nije iznenadio. Java je bila izvor velikih očekivanja još od svog predstavljanja pre 25 godina. Kada je predstavljena u veb pregledačima, javnost je pozdravila dolazak novog jezika.

Suosnivač kompanije „*Sun Microsystems*“ Bill Joy izjavio je: „Java predstavlja krajnji rezultat skoro petnaestogodišnjeg nastojanja da se kreiraju bolji programski jezik i okruženje za izradu jednostavnijeg i pouzdanijeg softvera.“

Kompaniju „Sun“, koja je Javu kreirala 1991. godine i prvi put je objavila u javnosti četiri godine kasnije, kupila je 2010. godine kompanija „Oracle“, koja je posvećena razvoju Jave od svojih najranijih godina i dalje podržava ovaj jezik i kreira nove verzije. Sada postoji i implementacija otvorenog koda.

Java jezik je u velikoj meri ispunio očekivanja javnosti. On je moćan deo softverskog razvoja i, baš kao istoimeni napitak, jedna vrsta Jave drži programere budnim noćima, a druga vrsta omogućava programerima da se bolje odmore nakon što razviju svoj softver.

Java je prvobitno predstavljena kao tehnologija za unapređenje veb sajtova pomoću programa koji se pokreću u pregledačima. Danas je veća verovatnoća da će se Java naći na serverima, pokrećući dinamične aplikacije u „oblaku“ koje su potpomognute relacionim bazama podataka na nekim od najvećih sajtova na Vebu i na milionima Android mobilnih telefona i tablet računara na kojima se mogu pokrenuti popularne aplikacije, kao što su Subway Surfers i Instagram.

U svakom novom izdanju Jave poboljšane su njene mogućnosti kao programskog jezika opšte namene za širok opseg okruženja. Java se koristi za desktop aplikacije, internet servere, mobilne uređaje i mnoga druga okruženja.

U svom 13. izdanju Java 12 jezik „sazreo“ je u vrhunsku konkurenciju ostalim programskim jezicima opšte namene, kao što su C++, Python i JavaScript.

Možda ste već upoznali alatke za programiranje Jave, kao što su Eclipse, NetBeans i IntelliJ IDEA. One omogućavaju razvoj funkcionalnih Java programa, a možete da koristite i „Oracleov“ Java Development Kit i Open Source OpenJDK. Ova dva kompleta, koja su na Internetu dostupna besplatno sa različitim uslovima licenciranja, obezbeđuju alatke komandne linije za pisanje, kompajliranje i testiranje Java programa. Još jedna besplatna alatka kompanije „Apache“ NetBeans je integrisano razvojno okruženje za kreiranje Java programa. Ovu alatku možete da preuzmete sa linka https://netbeans.apache.org.

U ovoj knjizi upoznajete sve aspekte razvoja Java softvera, jer se u njoj koriste najnovija verzija Java jezika i najbolje raspoložive tehnike u verziji Java Standard Edition koja se najčešće koristi. Programi se pripremaju i testiraju pomoću Apache NetBeansa, tako da možete brzo pokazati veštine kojima ste ovladali u svakoj lekciji.

Kada pročitate ovu knjigu, shvatićete zašto je Java postala najčešće korišćeni programski jezik na našoj planeti.

Kako je knjiga organizovana

U ovoj knjizi naučićete Java jezik, a takođe ćete saznati kako da ga koristite za kreiranje aplikacija u svakom računarskom okruženju. Kada pročitate celu knjigu, dobro ćete razumeti Javu i Java biblioteke klasa. Koristeći svoje nove veštine, moći ćete da razvijete sopstvene programe za zadatke, kao što su veb servisi, povezivanje baze podataka, obrada XML-a i mrežno programiranje.

Takođe ćete naučiti da kreirate nekoliko programa u svakoj lekciji u kojima se prikazuju predstavljene teme. Izvorni kod svih ovih programa dostupan je na zvaničnom veb sajtu knjige, na adresi www.java21dais.com, zajedno sa ostalim dodatnim materijalima, kao što su odgovori na pitanja čitalaca.

Ova knjiga „pokriva“ Java jezik i njegove biblioteke klasa u 21 lekciji koje su organizovane u tri dela, od kojih svaki „pokriva“ široku oblast razvoja Java programa.

U prvom delu učite o samom jeziku Java:

- Lekcija 1 „pokriva“ osnove - šta je Java jezik, zašto bi trebalo da ga naučite i kako da kreirate softver, koristeći moćan stil razvoja koji se naziva objektno-orijentisano programiranje. U ovoj lekciji ćete kreirati svoju prvu Java aplikaciju.
- U Lekciji 2 ćete „zaroniti“ u osnovne Java gradivne blokove - tipove podataka, promenljive i izraze.

- U Lekciji 3 detaljno je objašnjeno kako da koristite objekte u Javi - kako da ih kreirate, kako da koristite njihove promenljive, kako da pozovete njihove metode i kako da uporedite te objekte.
- U Lekciji 4 Java programe ćete poboljšati pomoću uslova, a koristićete i nizove i petlje.
- U Lekciji 5 u potpunosti je istraženo kreiranje klasa, tj. osnovni gradivni blokovi bilo kojeg Java programa.
- U Lekciji 6 ćete otkriti više informacija o interfejsima i paketima koji su korisni za grupisanje klasa i organizovanje hijerarhije klasa.
- Lekcija 7 „pokriva“ tri moćne karakteristike Jave - izuzetke koji omogućavaju da istražujete greške, programske niti koje omogućavaju da istovremeno pokrenete različite delove programa i asertacije, tj. tehniku koja program čini pouzdanijim.

Drugi deo je posvećen najkorisnijim Oracle klasama za upotrebu u Java programima:

- U Lekciji 8 predstavljene su strukture podataka koje možete da koristite kao alternativu za znakovne nizove i nizove - liste nizova, stekove, mape, heš mape i skupove bitova. Takođe je opisana petlja `for` koja podatke čini lakšim za upotrebu.
- U Lekciji 9 započinjemo istraživanje vizuelnog programiranja u pet lekcija. Naučićete kako da kreirate grafički korisnički interfejs, koristeći Swing klase za interfejse, grafiku i korisnički unos.
- Lekcija 10 „pokriva“ više desetina komponenata interfejsa koje možete da koristite u Java programu, uključujući dugmad, tekstualna polja, klizače, pomeranje oblasti sa tekstom i ikone.
- U Lekciji 11 objašnjeno je kako da učinite da korisnički interfejs izgleda fantastično pomoću *raspoređivača (managers layouts)*, tj. skupa klasa koje određuju kako su komponente na interfejsu raspoređene.
- U Lekciji 12 završavamo razmatranje Swing klasa pomoću rukovanja događajima koje programu omogućava da reaguje na klikove miša i druge interakcije korisnika.
- U Lekciji 13 naučićete da crtate oblike i znakove na komponentama korisničkog interfejsa.
- U Lekciji 14 završavamo istraživanje Swinga predstavljanjem korisnih ugnežđenih klasa i sofisticiranog raspoređivača.

U trećem delu prelazimo na napredne teme:

- U Lekciji 15 obezbeđen je kompletan uvod u lambda izraze, koji se takođe nazivaju *zatvaranja* (closures). Lambda izrazi omogućavaju da prvi put upotrebite novi tip Java kodiranja koje se zove *funkcionalno programiranje*. Interne klase se istražuju detaljnije, jer se odnose na zatvaranja.

- U Lekciji 16 se razmatraju ulazi i izlazi u kojima se koriste *strimovi*, tj. skup klasa koje omogućavaju pristup datotekama i mreži i drugo sofisticirano rukovanje podacima.
- U Lekciji 17 proširićete svoje znanje o strimovima da biste mogli da napišete programe koji komuniciraju na Internetu pomoću HTTP-a, uključujući socket programiranje, bafere, kanale i rukovanje URL-ovima.
- U Lekciji 18 je prikazano kako da povežete relacione baze podataka, koristeći Java Database Connectivity (JDBC). Naučićete kako da iskoristite mogućnosti baze podataka Derby, tj. baze podataka otvorenog koda koja je uključena u Javu.
- U Lekciji 19 ćete naučiti kako da čitate i pišete RSS dokumente pomoću XML Object Modela (XOM), tj. biblioteke Java klase otvorenog koda. RSS feedovi, koji su jedan od najpopularnijih XML dijalekata, milionima ljudi omogućavaju da prate ažuriranja veb sajtova i ostalog novog veb sadržaja.
- U Lekciji 20 istražujemo kako se pišu klijenti veb usluga pomoću još jedne biblioteke otvorenog koda Apache XML-RPC.
- U lekciji 21 su spojeni svi delovi, pokazujući kako možete kreirati igru (slagalicu) koja se zove `Banko`. „Zaronićete“ u izvorni kod kompletne Java aplikacije da biste mogli da testirate proces koji ćete izvršiti kada budete kreirali sopstvene programe.

Ko treba da čita ovu knjigu?

U ovoj knjizi Java jezik je namenjen za tri grupe čitalaca:

- za novajlije koje su relativni početnici u programiranju
- za ljude koji poznaju starije verzije Jave
- za iskusne programere koji koriste druge programske jezike, kao što su Visual C++, JavaScript i Python

Kada pročitate celu knjigu, moći ćete da eksperimentišete bilo kojim aspektom Java jezika. Takođe ćete moći da započnete sopstvene ambiciozne programske projekte, kako na Internetu, tako i izvan njega.

Ako ste novi u programiranju ili nikad niste napisali neki program, možda ćete se zapitati da li je ovo prava knjiga za vas. Pošto su svi koncepti u ovoj knjizi ilustrovani pomoću radnih programa, moći ćete da ovladate svim lekcijama, bez obzira koliko je bogato vaše iskustvo. Ako znate šta su promenljive i petlje, imaćete koristi od ove knjige. Za vašu želju da pročitate ovu knjigu mogući razlozi su sledeći:

- ako ste imali neke časove o osnovi programiranja u školi, shvatili šta je programiranje i čuli da je Java moćna, odlična i jednostavna za učenje
- ako već nekoliko godina programirate na nekom drugom jeziku, a stalno čujete pohvale za Javu i želite da proverite da li će ona ispuniti vaša očekivanja
- ako ste čuli da je Java odlična za veb aplikacije i Android programe

Ako još niste upoznali objektno-orijentisano programiranje, tj. stil programiranja koji se koristi u Javi, nemojte se obeshrabriti. U ovoj knjizi se pretpostavlja da nemate znanja o objektno-orijentisanom dizajnu. Dobićete priliku da naučite ovu razvojnu metodologiju dok učite Javu.

Ako ste potpuni početnik u programiranju, ova knjiga će vam se možda činiti komplikovanom. Međutim, Java je dobar jezik za početak programiranja, pa ako ste strpljivi i ako pregledate sve primere, još uvek možete da naučite Javu i da započnete kreiranje sopstvenih programa.

Konvencije upotrebljene u ovoj knjizi

NAPOMENA Napomena predstavlja interesantan, ponekad i tehnički, deo informacija koje se odnose na aktuelnu temu.

SAVET U odeljku Savet pružamo vam savete ili prikazujemo jednostavniji metod za izvršavanje zadatka.

PAŽNJA! U odeljku Pažnja! upozoravamo vas na potencijalne probleme i objašnjavamo kako da ih izbegnete.

qweryTekst koji unesete i tekst koji se prikazuje na ekranu predstavljen je fontom `monospace`:

```
Monospace izgleda ovako. Hi, Mom!
```

Ovaj font predstavlja izgled teksta na ekranu. Čuvari mesta za promenljive i izraze prikazuju se u fontu *`monospace italic`*.

Na kraju svake lekcije nalazi se nekoliko posebnih funkcija: odgovori na najčešće postavljena pitanja o temi koja je predstavljena, kviz za proveru vašeg znanja o materiji obrađenoj u konkretnoj lekciji, dve vežbe koje možete sami da uradite i praktična pitanja u slučaju da se pripremate za sticanje Java sertifikata. Rešenja za vežbe i odgovori na pitanja o sertifikaciji se nalaze na zvaničnom veb sajtu knjige.

Postanite član Kompjuter biblioteke

Kupovinom jedne naše knjige stekli ste pravo da postanete član Kompjuter biblioteke. Kao član možete da kupujete knjige u pretplati sa 40% popusta i učestvujete u akcijama kada ostvarujete popuste na sva naša izdanja. Potrebno je samo da se prijavite preko formulara na našem sajtu.
Link za prijavu: http://bit.ly/2TxekSa

Skenirajte QR kod
registrujte knjigu
i osvojite nagradu

DEO I

Java jezik

LEKCIJA 1

Početak rada u Javi

Java je izuzetno uspešna u svojim nastojanjima da obuhvati mnoštvo različitih domena, tako da možete da koristite aplikacioni server, mobilni telefon i interplanetarnu navigaciju, da vršite naučno programiranje, da pišete softver, itd...

autor Java jezika James Gosling

Kada je Java programski jezik predstavljen javnosti, pre više od 20 godina, on je bio inventivna „igračka" za veb pregledače, koja je imala potencijal da postane mnogo bolja.

Reč *„potencijal"* je kompliment koji ima svoj rok trajanja. Pre ili kasnije, potencijal se mora realizovati da ga ne bi zamenile nove reči i izrazi, kao što su*„lenština"*, *„neverica"*, *„gubitak prostora"* ili *„veliko razočaranje za vašu majku i mene"*.

Dok razvijate vaše veštine tokom čitanja ove knjige, bićete u odličnoj poziciji da procenite da li je Java jezik ispunio sva vaša očekivanja.

Takođe ćete postati Java programer sa mnogo potencijala.

Java jezik

Od svoje prve javne primene kao Jave 1.0, 1996. godine, Java programski jezik ispunio je mnoga očekivanja koja su pratila njegov „dolazak“. Više od 15 miliona programera naučilo je ovaj jezik i koristi ga na mestima, kao što su Google, NASA, IBM, ServiceNow i Netflix. Java jezik je osnovni deo akademskog programa u odeljenjima za računarske nauke širom sveta. On je prvi put korišćen za kreiranje jednostavnih programa na veb stranicama, a danas se može naći na sledećim mestima (i mnogim drugim):

- u serverima u „oblaku“
- u relacionim bazama podataka
- u orbitalnim teleskopima
- u čitačima za elektronske knjige
- u mobilnim telefonima

Iako je Java i dalje korisna u veb aplikacijama i na serverima, njene „ambicije“ se sada šire i izvan Veba. Ona je jedan od najpopularnijih programskih jezika opšte namene.

Istorija jezika

Priča o Java jeziku je dobro poznata. Tim programera radio je na interaktivnom TV projektu u kompaniji „Sun Microsystems“ tokom 1990-ih godina. Jedan od članova tog tima bio je James Gosling, koji je bio nezadovoljan objektno-orijentisanim programskim jezikom C++, razvijenim deceniju ranije kao proširenje jezika C.

Da bi se pozabavio onim zbog čega je postao nezadovoljan, Gosling se zatvorio u svoju kancelariju i kreirao novi jezik koji je bio pogodan za njegov projekat.

Iako je taj interaktivni TV projekat bio neuspešan, Goslingov jezik je neočekivano počeo da se primenjuje u novom mediju World Wide Webu, koji je istovremeno postao popularan.

Java jezik se prvi put pojavio u javnosti kao „alpha version 1.0a2“ 23. marta 1995. godine. Iako je većina karakteristika ovog jezika bila primitivna, u poređenju sa karakteristikama C++ pogramskog jezika (i Java jezika danas), on je imao odličnu aplikaciju, tj. specijalne Java programe, pod nazivom apleti, koji su se mogli pokrenuti kao deo veb stranica u Netscape Navigatoru, najpopularnijem veb pregledaču u to vreme.

Ova funkcija, tj. prvi interaktivni program koji je dostupan na Vebu, skrenula je toliko pažnje na novi jezik da je nekoliko stotina hiljada programera naučilo Java jezik u prvih šest meseci nakon njegove pojave.

Nakon što su Java programi koji se pokreću na veb stranicama prestali da budu novina, sveukupne prednosti jezika postale su jasne i svi ti programeri su se „zadržali“ u Javi. Danas postoji više profesionalnih Java programera nego profesionalnih C++ programera.

Kompanija „Sun Microsystems“ kontrolisala je razvoj Java jezika, od njegovog početka, do 2010. godine, kada ju je kupio gigant baze podataka i poslovni softverski gigant „Oracle“ za 7,4 milijarde dolara. Kompanija „Oracle“, koja je dugogodišnji korisnik Java jezika u sopstvenim proizvodima, izuzetno je posvećena podršci tom jeziku i nastavlja da povećava njegove mogućnosti u svakom novom izdanju.

Uvod u Java jezik

Java je objektno-orijentisan, platformski neutralan i bezbedan jezik koji je dizajniran tako da se lakše uči i teže zloupotrebljava nego C++ programski jezik.

Objektno-orijentisano programiranje (OOP - Object-oriented rogramming) je metodologija razvoja softvera u kojoj se program konceptualizuje kao grupa objekata koji funkcionišu zajedno. Objekti su kreirani za obrasce koji se nazivaju *klase* i sadrže podatke i kod neophodan za upotrebu tih podataka. Java je izuzetno objektno-orijentisan jezik, kao što ćete videti kasnije u ovoj lekciji kada budete kreirali svoju prvu klasu i koristili je za kreiranje objekata.

Platformska neutralnost je mogućnost programa da se pokreće bez modifikacija u različitim računarskim okruženjima. Java programi se transformišu u format zvani *bajtkod*, koji može pokrenuti svaki računar ili uređaj opremljen Java virtuelnom mašinom (JVM – Java Virtual Machine). Java program možete da kreirate na Windows 10 uređaju koji se pokreće baš kao bajtkod na Linux veb serveru, na Apple Macu koji koristi OS 10.14 i na telefonu Samsung Galaxy S10. Sve dok platforma ima JVM, ona može pokrenuti bajtkod.

Java je dizajnirana tako da bude lakša za učenje od C++ jezika, pre svega na sledeće načine:

- Java automatski dodeljuje i oslobađa memoriju, pa programeri ne moraju da obavljaju ovaj naporan posao koji je podložan greškama.
- Java ne sadrži pokazivače, tj. moćnu funkciju za iskusne programere koja se lako može zloupotrebiti i može izazvati veliku bezbednosnu ranjivost.
- Java uključuje samo pojedinačno nasleđivanje klasa u objektno-orijentisanom programiranju.
- Java se ne mora kompajlirati na svakoj platformi izvršavanjem istog bajtkoda.

Nedostatak pokazivača i prisustvo automatskog upravljanja memorijom su dva ključna elementa bezbednosti Jave.

Odabir programerskih alatki

Sada, kada vam je Java predstavljena kao „posmatraču“, vreme je da neke od tih koncepata reprodukujete i da kreirate svoj prvi Java program.

Dok ne stignete do 21. lekcije ove knjige, dobro ćete upoznati Javine mogućnostima, uključujući grafiku, XML obradu, mikroservise i razvoj baze podataka. Napisaćete programe koji se mogu pokrenuti na računarima, na veb serverima, na mobilnim telefonima i u drugim računarskim okruženjima.

Pre nego što nastavite dalje, na računaru morate imati softver koji može da se koristi za uređivanje, pripremu i pokretanje Java programa koji koriste najsavremenije verzije Java jezika.

Nekoliko popularnih integrisanih razvojnih okruženja (IDE-a) podržavaju Java verziju 1213, uključujući Apache NetBeans i Eclipse otvorenog koda i komercijalni IntelliJ IDE. Java 13 će biti podržana u vreme kada budete čitali ovu knjigu.

Ako učite da koristite ove alatke istovremeno dok učite i Javu, to može biti pomalo zastrašujući zadatak. Većina IDE-a fokusirana je prevashodno na iskusne programere koji žele da budu produktivniji, a ne na nove ljude koji kreću svojim prvim nesigurnim koracima ka novom jeziku.

Najjednostavnija alatka za razvoj Jave je Java Development Kit (JDK), koja je besplatna i može se preuzeti sa Java SE Downloads stranice, na adresi www.oracle.com/technetwork/java/javase/downloads.

Kad god kompanija „Oracle“ izda novu verziju Jave, obezbeđuje i besplatni JDK koji je dostupan na Vebu za podršku toj verziji. JDK za najnoviju verziju Jave, koja je u vreme pisanja ovog teksta bila Java SE 12.0.2 Oracle JDK, trebalo bi da bude prvi na stranici za preuzimanje.

Mana razvoja Java programa samo pomoću JDK-a je ta što je JDK skup alatki komandne linije bez grafičkog korisničkog interfejsa za uređivanje programa, za pripremu programa za pokretanje, za pakovanje programa za implementaciju ili za testiranje. Komandna linija je, jednostavno, odzivnik za unos tekstualnih komandi. Dostupna je u Windowsu kao program Command Prompt, a na Macu kao program Terminal.

Kao alternativa, na adresi https://netbeans.apache.org postoji odličan besplatan IDE za Java programere, koji se zove Apache NetBeans. Iako je NetBeans sofisticirani IDE, može lako da se nauči kako da se koristi za osnove kreiranja i testiranja Java programa.

Pošto je NetBeans za većinu ljudi jednostavniji za upotrebu od JDK komandne linije, on je korišćen u ovoj knjizi.

Ako na računaru još nemate Java programsku alatku i želite da isprobate NetBeans, možete saznati kako da započnete korišćenje integrisanog razvojnog okruženja pomoću softvera u Dodatku A, „Korišćenje NetBeans integrisanog razvojnog okruženja“. U tom dodatku je opisano kako da preuzmete i instalirate IDE i kako da ga koristite za kreiranje uzorka Java programa da biste se uverili da taj program funkcioniše.

Kada na računaru imate Java programsku alatku koja podržava najnoviju verziju Jave, spremni ste da „zaronite“ u ovaj programski jezik.

Ako još nemate programsku alatku na računaru, sada je vreme da je instalirate.

> **SAVET** Za informacije o drugim IDE-ima za Javu koji su pomenuti u ovom odeljku posetite IDEA sajt na adresi www.jetbrains.com/idea i Eclipse sajt na adresi www.eclipse.org.

Objektno-orijentisano programiranje

Najveći izazov za novog Java programera je učenje objektno-orijentisanog programiranja tokom učenja Java jezika.

Iako ovo može zvučati zastrašujuće ukoliko nikada ranije niste programirali u ovom stilu, zamislite to kao „popust dva za jedan” za vaš mozak. Objektno-orijentisano programiranje naučićete dok učite Javu. Ne postoji drugi način za učenje Java jezika.

Objektno-orijentisano programiranje predstavlja pristup za izradu računarskih programa, koji „imitira“ kako se objekti sastavljaju u fizičkom svetu.

Korišćenjem ovog stila programiranja možete da kreirate programe koji se mogu upotrebiti više puta i koji su pouzdani i razumljivi. Da biste došli do te „tačke“, prvo morate da istražite kako Java ugrađuje principe objektno-orijentisanog programiranja (često opisanog skraćenicom OOP).

Ako već poznajete objektno-orijentisanim programiranje, veći deo ove lekcije će za vas biti samo pregled tog programiranja. Čak i ako preskočite uvodni materijal, trebalo bi da kreirate primer programa da biste stekli neko iskustvo u razvoju, kompajliranju i pokretanju Java programa.

Postoji mnogo načina za konceptualizaciju računarskog programa. Jedan od načina je da zamislite program kao niz instrukcija koje se izvršavaju redom; ovo se zove *proceduralno programiranj*e. Raniji programski jezik BASIC je proceduralni.

Proceduralni programski jezici odražavaju način na koji računar izvršava instrukcije, tako da su programi koje pišete prilagođeni načinu na koji računar funkcioniše. Proceduralni programer mora prvo da nauči kako da podeli problem na niz jednostavnih koraka koji se izvršavaju redom.

Objektno-orijentisano programiranje „posmatra“ računarski program iz drugačijeg ugla, fokusirajući se na zadatak zbog kojeg je on kreiran, a ne na način na koji računar upravlja zadacima.

U objektno-orijentisanom programiranju računarski program se konceptualizuje kao skup objekata koji zajedno izvršavaju neki zadatak. Svaki objekat je poseban deo programa, koji komunicira sa drugim delovima programa na visokokontrolisani način.

Za primer objektno-orijentisanog dizajna u realnom životu razmotrite kućni stereo-sistem. Većina sistema je izgrađena spajanjem gomile različitih objekata, koji se uobičajeno nazivaju *komponente.* Ako ste se iz kupovine vratili sa stereo-sistemom, možda ćete doneti kući ove različite objekte:

- komponente zvučnika koje reprodukuju zvukove srednje i visoke frekvencije
- komponentu Subwoofer koja reprodukuje zvukove niske bas frekvencije
- komponentu tjunera koja prima radio-difuzne signale
- komponentu CD plejera koja čita audio-podatke sa CD-a
- komponentu gramofona koja čita audio-podatke sa gramofonskih ploča

Ove komponente su dizajnirane tako da međusobno funkcionišu pomoću standardnih priključaka za ulaz i izlaz. Čak i ako ste kupili zvučnike, Subwoofer, tjuner, CD plejer i gramofon pet pet različitih proizvođača, možete da ih kombinujete da biste formirali stereo-sistem ukoliko svaka komponenta ima standardne priključke.

Objektno-orijentisano programiranje funkcioniše po istom principu: sastavljate program kreiranjem novih objekata koje povezujete međusobno i sa postojećim objektima. Ovi objekti mogu da budu oni koje kreirate ili oni koje obezbeđuju Oracle, Google, Apache Project ili neki drugi softverski programer. Svi objekti su sastavni delovi većeg programa i kombinuju se na standardni način. Svaki objekat ima određenu „ulogu“ u većem programu.

Objekat je samostalna komponenta računarskog programa koja predstavlja srodnu grupu funkcija, a dizajniran je za izvršavanje određenih zadataka. Ako jedan objekat ima previše posla, njegov posao bi mogao biti podeljen na dva ili više objekata.

Objekti i klase

Objektno-orijentisano programiranje je zasnovano na opažanju da se u fizičkom svetu veliki objekti sastoje od mnogobrojnih manjih objekata.

Mogućnost kombinovanja objekata samo je jedan od aspekata objektno-orijentisanog programiranja. Drugi aspekt je upotreba klasa.

Klasa je obrazac koji se koristi za kreiranje objekata. Svaki objekat kreiran iz iste klase ima slične karakteristike i funkcije.

Klase predstavljaju sve aspekte skupa objekata. Kada pišete program na objektno-orijentisanom jeziku, ne definišete pojedinačne objekte, već definišete klase i koristite ih za kreiranje objekata.

Ako ste na Java jeziku napisali program za umrežavanje računara, mogli biste da kreirate klasu `Router`, koja opisuje funkcije svih internet rutera. Ovi uređaji imaju sledeće zajedničke karakteristike:

- Povezuju se na Ethernet port računara.
- Šalju i primaju informacije.
- Komuniciraju sa internet serverima.

Klasa `Router` služi kao apstraktni model koncepta takvih uređaja. Da biste konkretno mogli da manipulišete programom, potreban vam je objekat. Morate da koristite klasu `Router` da biste kreirali objekat `Router`. Proces kreiranja objekta iz klase zove se *instanciranje*, zbog čega se objekti nazivaju *instance*.

Klasa `Router` može da se koristi za kreiranje različitih objekata `Router` u programu, od kojih svaki ima različite funkcije - na primer:

- Neki objekti funkcionišu kao modem velike brzine, dok drugi ne.
- Imaju sigurnosni zaštitni zid.
- Podržavaju različite protokole za komunikaciju.

Iako se ova dva `Router` objekta razlikuju, još uvek imaju dovoljno zajedničkog da budu prepoznatljivi kao usko povezani objekti.

Evo još jednog primera: pomoću Jave možete da kreirate klasu koja će predstavljati svu komandnu dugmad, tj. okvire na koje se može kliknuti i koji se prikazuju u prozorima i ostalim delovima grafičkog korisničkog interfejsa.

Kada je razvijena klasa `Button`, ona može da definiše sledeće karakteristike:

- tekst prikazan na dugmetu
- veličinu dugmeta
- ponašanje kada se klikne na to dugme
- aspekte izgleda dugmeta - na primer, da li dugme ima 3D senku

Nakon što definišete klasu `Button`, možete da kreirate primere tog dugmeta - drugim rečima, možete da kreirate objekte `Button`. Svi objekti poprimaju osnovne karakteristike dugmeta, baš kao što je definisano klasom. Međutim, svaki od objekata može da ima drugačiji izgled i malo drugačije ponašanje, u zavisnosti od toga šta konkretni objekat treba da izvrši.

Kada je kreirana klasa `Button`, ne morate ponovo da pišete kod za svako dugme koje želite da koristite u vašim programima. Možete ponovo da koristite klasu `Button` da biste, po potrebi, kreirali različite vrste dugmadi u svakom programu koji razvijate.

Kada pišete Java program, vi dizajnirate i konstruišete skup klasa. Kada se program izvršava, iz tih klasa se kreiraju objekti koji se koriste prema potrebi. Vaš zadatak kao Java programera je da kreirate odgovarajući skup klasa da biste postigli ono što vaš program treba da izvrši.

Srećom, ne morate da započnete pisanje Java programa „od nule“. Java jezik uključuje biblioteku Java Class, sa više od 4.400 klasa koje implementiraju većinu neophodnih funkcija. Ove klase su instalirane zajedno sa programerskim alatkama, kao što su NetBeans ili JDK.

Kada govorimo o programiranju na Java jeziku, govorimo, u stvari, o korišćenju ove biblioteke klasa i standardnih ključnih reči i operatora definisanih u Javi.

Biblioteka klasa izvršava brojne zadatke, kao što su matematičke funkcije, obrada teksta, grafičko prikazivanje, interakcije sa korisnicima i umrežavanje. Korišćenje ovih klasa se ne razlikuje od korišćenja Java klasa koje sami kreirate.

Za komplikovane Java programe možete da kreirate skup novih klasa koje formiraju sopstvenu biblioteku klasa, koju možete da distribuirate za upotrebu u drugim programima.

Ponovna upotreba je jedna od osnovnih prednosti objektno-orijentisanog programiranja.

> **NAPOMENA** U biblioteci jedne od Javinih standardnih klasa Java Class Library `JButton` u paketu `javax.swing` obuhvata sve funkcije ovog hipotetičkog primera `Button`, zajedno sa još mnogo elemenata. U Lekciji 9, „Kreiranje grafičkog korisničkog interfejsa“, imaćete priliku da kreirate objekte iz ove klase.

Atributi i ponašanje

Java klasa se sastoji od dve vrste podataka: atributa i ponašanja. Obe vrste su prisutne u projektu `MarsRobot`, koji ćete sada implementirati kao klasu. Ovaj projekat jednostavne simulacije vozila za istraživanje planeta inspirisan je misijom „Mars Exploration Rovers“, tokom koje je korišćen NASA program Jet Propulsion Laboratory za istraživanje geologije planete Mars.

Pre nego što kreirate program, morate da naučite kako su objektno-orijentisani programi dizajnirani u Javi. Možda će vam biti teško da razumete ove koncepte dok ih upoznajete, ali naučićete ih pomoću vežbi u ovoj knjizi.

Atributi klasa objekata

Atributi su podaci po kojima razlikujemo jedan objekat od drugog. Oni se mogu koristiti za određivanje izgleda, stanja i drugih kvaliteta objekata koji pripadaju toj klasi.

Vozilo za istraživanje Marsa može da ima sledeće atribute:

- **status** - istraživanje, premeštanje, povratak kući
- **brzina** - meri se u miljama na sat
- **temperatura** - izmerena u stepenima Farenhajta

U klasi su atributi definisani *promenljivim* koje su mesta za skladištenje informacija u računarskom programu. *Promenljive instance* su atributi koji imaju vrednosti koje se razlikuju, u zavisnosti od objekta.

Promenljiva instance definiše atribut jednog određenog objekta. Klasa objekta definiše vrstu atributa, a svaka instanca skladišti sopstvenu vrednost za taj atribut. Promenljive instance se nazivaju i *objektne promenljive.*

Svaki atribut klase ima jednu odgovarajuću promenljivu. Taj atribut objekta možete da izmenite kada promenite vrednosti promenljive.

Na primer, klasa `MarsRobot` definiše promenljivu instance `speed`. Klasa mora biti promenljiva instance, jer svaki robot se kreće nekom svojom brzinom. Vrednost promenljive instance robota `speed` može se promeniti da bi se robot kretao brže ili sporije.

Promenljivim instance može da se dodeli vrednost kada se objekat kreira, koja će ostati konstantna tokom čitavog „životnog veka“ objekta. Ovim promenljivim mogu da se dodele i različite vrednosti, jer se objekat koristi u pokrenutom programu.

Za neke promenljive koje ne moraju biti različite u svakom objektu ima smisla imati jednu vrednost koju koriste svi objekti iz te klase. Ti se atributi zovu *promenljive klase*.

Promenljiva klase definiše atribut čitave klase. Ona se odnosi na samu klasu i na sve njene instance, tako da se skladišti samo jedna vrednost, bez obzira koliko objekata te klase je kreirano u programu.

Primer promenljive klase za `MarsRobot` klasu bi bila promenljiva `topSpeed`, koja skladišti maksimalnu brzinu kojom bilo koji robot može da se kreće. Ako je promenljiva instance kreirana za skladištenje brzine, svaki objekat može da ima drugačiju vrednost za ovu promenljivu, što može izazvati probleme, jer ni jedan robot ne može da premaši tu vrednost.

Upotreba promenljive klase sprečava ovaj problem, jer svi objekti te klase automatski imaju istu vrednost. Svaki objekat `MarsRobot` ima pristup toj promenljivoj.

Ponašanje klase objekata

Ponašanje se odnosi na zadatke koje klasa objekata može da izvrši za sebe i za druge objekte. Ono se može koristiti za promenu atributa objekta, primanje informacija od drugih objekata i slanje poruka drugim objektima, pri čemu se od njih traži da izvršavaju zadatke.

Robot Mars može da:

- proverava trenutnu temperaturu
- započne istraživanje područja
- akceleriše ili uspori svoju brzinu
- prijavi svoju trenutnu lokaciju

Ponašanje klase objekata implementira se korišćenjem metoda.

Metod je grupa iskaza u klasi. Metodi se koriste za izvršavanje određenih zadataka u objektima metoda i u drugim objektima. Mogu da se upoređuju sa funkcijama i potprogramima u drugim programskim jezicima. Dobro dizajnirani metod izvršava samo jedan zadatak.

Objekti međusobno komuniciraju pomoću metoda. Metod je poput jednog objekta koji daje komandu drugom objektu. Klasa ili objekat mogu pozivati metode u druge objekte zbog više razloga, uključujući sledeće:

- da prijave promenu u drugom objektu
- da ukažu drugom objektu da treba da nešto promeni na sebi
- da zatraže od drugog objekta da nešto uradi

Na primer, dva robota Mars mogu da koriste metode da bi izveštavali o svom položaju i izbegli sudare, a jedan robot može da „kaže“ drugom da se zaustavi kako bi mogao bezbedno da prođe.

Baš kao što postoje promenljive instance i klase, takođe postoje metodi instance i klase. Metodi instance, koji se obično nazivaju samo *metodi*, koriste se u objektu klase. Metod koji menja pojedinačni objekat mora da bude metod instance. Metodi klase odnose se na samu klasu.

Kreiranje klase

Do sada ste teoretski učili o objektno-orijentisanom programiranju. Da biste videli klase, objekte, atribute i ponašanje u akciji, sada ćete da razvijete klasu `MarsRobot`, da kreirate objekte iz te klase i da ih koristite u pokrenutom programu.

> **NAPOMENA** Glavna svrha ovog projekta je istraživanje objektno-orijentisanog programiranja. Više informacija o Java sintaksi programiranja ćete saznati u Lekciji 2, „Osnove programiranja“.

Pre nego što počnete da koristite `MarsRobot`, potrebno vam je malo pripreme.

U ovoj knjizi se koristi NetBeans kao primarna programska alatka za kreiranje Java programa. NetBeans organizuje Java klase u projektima. Bilo bi korisno da imate projekat za skladištenje klasa. Pokrenite NetBeans, a ako to već niste učinili, kreirajte projekat, tako što ćete preduzeti sledeće radnje:

1. Izaberite komandu menija File, New Project. Biće prikazan dijaloški okvir New Project.
2. U oknu Categories izaberite stavku Java with Ant.
3. U oknu Projects izaberite Java Application i kliknite na Next. Otvoriće se novi dijaloški okvir New Java Application (ako je aktivirana podrška za Java SE).
4. Ako se od vas zatraži da aktivirate podršku za Java SE, kliknite na Activate. Kada je aktivacija završena, biće prikazan dijaloški okvir New Java Application.
5. U tekstualno polje Project Name unesite naziv projekta (na primer, `Java21`). Polje Project Folder će biti ažurirano kada upišete naziv projekta. Zapišite naziv ove fascikle, jer u njoj u računaru mogu biti smešteni vaši Java programi.
6. Poništite polje za potvrdu Create Main Class.
7. Kliknite na Finish.

Novi projekat je kreiran. Možete da ga koristite u čitavoj knjizi.

Ako ste već kreirali projekat, on će verovatno biti otvoren u NetBeansu. Ako nije otvoren, izaberite komandu menija File, Open Recent Project da biste ga pronašli i izabrali.

Nova klasa koju ćete kreirati biće dodata u trenutni projekat.

Da biste započeli kreiranje svoje prve klase, prvo kreirajte novi program:

1. Izaberite komandu menija File, New File. Otvoriće se dijaloški okvir New File.
2. U oknu Categories izaberite stavku Java.
3. U oknu File Types izaberite Empty Java File i kliknite na Next. Otvoriće se dijaloški okvir Empty Java File.
4. U tekstualno polje Class Name ukucajte `MarsRobot`.
5. U polje Package ukucajte `com.java21days`. Datoteka koju ste kreirali prikazuje se u polju Created Field i ne može se menjati. Ova datoteka ima naziv `MarsRobot.java`.
6. Kliknite na Finish.

Uređivač izvornog koda NetBeans otvara se prazan. Popunite ga kodom koji je naveden u listingu 1.1. Kada završite unošenje koda, sačuvajte datoteku pomoću komande menija File, Save. Datoteka `MarsRobot.java` je sada sačuvana.

> **NAPOMENA** Ne upisujte brojeve na početku svakog reda u listingu. Oni nisu deo programa. Dodati su da bi se pojedine linije mogle opisati u ovoj knjizi.

LISTING 1.1 Kompletan tekst datoteke MarsRobot.java

```
 1: package com.java21days;
 2:
 3: class MarsRobot {
 4:     String status;
 5:     int speed;
 6:     float temperature;
 7:
 8:     void checkTemperature() {
 9:         if (temperature < -80) {
10:             status = "returning home";
11:             speed = 5;
12:         }
13:     }
14:
15:     void showAttributes() {
16:         System.out.println("Status: " + status);
17:         System.out.println("Speed: " + speed);
18:         System.out.println("Temperature: " + temperature);
19:     }
20: }
```

Kada sačuvate ovu datoteku, ako nema grešaka, NetBeans automatski kreira klasu `MarsRobot`. Ovaj proces se naziva *kompajliranje* klase i uključuje upotrebu alatke koja se zove *kompajler*. Kompajler pretvara linije izvornog koda u bajtkod koji pokreće JVM.

Iskaz `package` u liniji 1 listinga 1.1 smešta klasu u *paket* koji grupiše srodne klase u Javi. Paket ima naziv `com.java21days`.

Iskaz `class` u liniji 3 definiše i imenuje klasu `MarsRobot`. Sve što se nalazi između početne zagrade ({) u liniji 3 i završne zagrade (}) u liniji 20 je deo ove klase.

Klasa `MarsRobot` sadrži tri promenljive instance i dva metoda instance.

Promenljive instance su definisane u linijama 4–6:

```
String status;
int speed;
float temperature;
```

Promenljive se zovu `status`, `speed` i `temperature`. Svaka promenljiva se koristi za skladištenje različitih vrsta informacija:

- `status` skladišti objekat `String`, tj. grupu slova, brojeve, interpunkcijske znakove i druge znakove.
- `speed` skladišti `int`, tj. vrednost celog broja.
- `temperature` skladišti `float`, tj. broj sa pokretnom tačkom.

Objekti `String` se kreiraju iz klase `String` koja je deo biblioteke Java Class.

> **NAPOMENA** Kao što ste možda primetili, u ovom programu klasa može da koristi objekat `String` kao promenljivu instance.

Prvi metod instance u klasi `MarsRobot` definisan je u linijama 8–13:

```
void checkTemperature() {
    if (temperature < -80) {
        status = "returning home";
        speed = 5;
    }
}
```

Metod je definisan na sličan način kao klasa. Na početku definicije se nalazi iskaz koji imenuje metod, identifikuje vrstu informacija koje metod generiše i definiše druge elemente.

Metod `checkTemperature()` se nalazi u početnoj zagradi u liniji 8 listinga 1.1 i u završnoj zagradi u liniji 13. On se može pozvati u objektu `MarsRobot` da bi bila utvrđena temperatura tog objekta.

Pomoću ovog metoda se proverava da li vrednost promenljive objekta `temperature` ima vrednost manju od -80. Ako ima, menjaju se još dve promenljive instance:

- Promenljiva `status` se menja u tekst „returning home“, što ukazuje da je temperatura previše niska, a robot se vraća u svoju bazu.
- Vrednost brzine je promenjena u vrednost `5` (verovatno je ovo najveća brzina kojom robot može da se kreće).

Drugi metod instance `showAttributes()` definisan je u linijama 15–19:

```
void showAttributes() {
    System.out.println("Status: " + status);
    System.out.println("Speed: " + speed);
    System.out.println("Temperature: " + temperature);
}
```

Ovaj metod poziva metod `System.out.println()` da bi bile prikazane vrednosti tri promenljive instance, zajedno sa nekim tekstom u kojem je objašnjeno šta svaka vrednost predstavlja.

Ako još uvek niste sačuvali ovaj program, izaberite komandu menija File, Save. Ako datoteka nije promenjena od kada ste je prethodno sačuvali, ova komanda će biti onemogućena.

Pokretanje programa

Čak i ako ste pravilno uneli program `MarsRobot` u listingu 1.1 i uspešno ga kompajlirali u klasi, ne možete ništa učiniti sa njim. Klasa koju ste kreirali definiše objekat `MarsRobot`, ali ona, zapravo, ne kreira ni jedan objekat.

Postoje dva načina na koji klasa `MarsRobot` može da se koristi:

- Napišite zaseban Java program koji kreira objekat koji pripada toj klasi.
- Dodajte poseban metod klase **`main()`** u klasu `MarsRobot`, tako da ona može da se pokrene kao aplikacija, pa kreirajte objekat te klase u tom metodu.

Za ovu vežbu je izabrana prva opcija.

Listing 1.2 sadrži izvorni kod za `MarsApplication`, tj. za Java klasu koja kreira objekat `MarsRobot`, postavlja njegove promenljive instance i poziva metode. Prateći iste korake kao i ranije, kreirajte novu Java datoteku u NetBeansu u paketu `com.java21days` i dodajte joj naziv `MarsApplication`.

Da biste započeli kreiranje ove druge klase, sledite u NetBeansu ove korake:

1. U meniju izaberite File, New File. Otvoriće se dijaloški okvir New File.
2. U oknu Categories izaberite Java.

3. U oknu File Types izaberite Empty Java File i kliknite na Next. Otvoriće se dijaloški okvirEmpty Java File.
4. U tekstualnom poljuClass Name upišite `MarsApplication`.
5. U polje `Package` upišite `com.java21days` (ako već nije upisano). Datoteka koju ste kreirali prikazuje se u polju Create File i ima naziv `MarsApplication.java`.
6. Kliknite na Finish.

Nova datoteka je kreirana. U uređivaču izvornog koda NetBeansa unesite kod koji je prikazan u listingu 1.2.

LISTING 1.2 Kompletan tekst datoteke MarsApplication.java

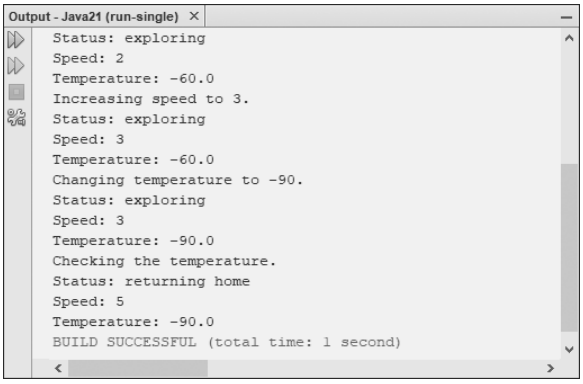
```
 1: package com.java21days;
 2:
 3: class MarsApplication {
 4:     public static void main(String[] arguments) {
 5:         MarsRobot spirit = new MarsRobot();
 6:         spirit.status = "exploring";
 7:         spirit.speed = 2;
 8:         spirit.temperature = -60;
 9:
10:         spirit.showAttributes();
11:         System.out.println("Increasing speed to 3.");
12:         spirit.speed = 3;
13:         spirit.showAttributes();
14:         System.out.println("Changing temperature to -90.");
15:         spirit.temperature = -90;
16:         spirit.showAttributes();
17:         System.out.println("Checking the temperature.");
18:         spirit.checkTemperature();
19:         spirit.showAttributes();
20:     }
21: }
```

Kada izaberete File, Save da biste sačuvali datoteku, NetBeans se automatski kompajlira u klasi `MarsApplication`, koja sadrži bajtkod za pokretanje JVM-a.

> **SAVET** Ako naiđete na probleme prilikom kompajliranja ili pokretanja bilo kog programa u ovoj knjizi, možete da pronađete kopiju datoteke izvornog koda i druge povezane datoteke na zvaničnom veb sajtu knjige na adresi www.java21days.com.

Nakon što kompajlirate aplikaciju, pokrenite program, tako što ćete izabrati komandu menija Run, Run File. Izlaz koji prikazuje klasa `MarsApplication` pojavljuje se u izlaznom oknu u NetBeansu, kao na slici 1.1.

Slika 1.1
Izlaz klase MarsApplication



Koristeći listing 1.2 kao vodič, možete videti sledeće stavke koje se izvršavaju u metodu klasemain() ove aplikacije:

- **Linija 4** – Kreira se i imenuje metod `main()`. Svi metodi `main()` izgledaju baš kao ovaj metod, kao što ćete videti u Lekciji 5, „Kreiranje klasa i metoda“. Za sada je najvažnije da obratite pažnju na rezervisanu reč `static`, koja označava da je metod koji koriste svi objekti `MarsRobot` metod klase.
- **Linija 5** – Kreira se novi objekat `MarsRobot` pomoću klase kao obrasca. Dat mu je naziv `spirit`.
- **Linije 6-8** – Promenljivim instance objekta `spirit` su dodeljene vrednosti: `status` je postavljen na vrednost „`exploring`“, `speed` na vrednost `2`, a `temperature` na vrednost `–60`.
- **Linija 10** – Metod `showAttributes()` objekta `spirit` se poziva i prikazuje trenutne vrednosti promenljivih instance `status`, `speed` i `temperature`.
- **Linija 11** – Kada se pozove metod `System.out.println()`, biće prikazan tekst u zagradama na ekranu.
- **Linija 12** – Promenljiva instance `speed` postavljena je na vrednost `3` pre nego što se atributi ponovo prikažu.
- **Linija 15** – Promenljiva instance `temperature` je postavljena na vrednost `-90` pre nego što se atributi prikažu treći put.
- **Linija 18** – Poziva se metod `checkTemperature()` objekta `spirit`. Pomoću njega se proverava da li je promenljiva instance `temperature` manja od `-80`. Ako je manja od -80, promenljivim instance `status` i `speed` se dodeljuju nove vrednosti.

NAPOMENA Ako zbog nečega ne možete da koristite NetBeans ili neki drugi IDE za pisanje Java programa, pa morate da koristite komplet Java Development Kit, možete saznati kako se instalira u dodatku D, „Upotreba kompleta Java Development Kit“.

Organizovanje klase i ponašanje klase

Uvod u objektno-orijentisano programiranje na Java jeziku zahteva razmatranje još dva koncepta: nasleđivanja i paketa. Oba koncepta su mehanizmi za organizovanje klasa.

Nasleđivanje

Nasleđivanje, koje predstavlja jedan od najvažnijih koncepata objektno-orijentisanog programiranja, ima direktan uticaj na način kako dizajnirate i pišete Java klase.

Nasleđivanje je mehanizam koji omogućava jednoj klasi da nasledi ponašanje i atribute druge klase. Pomoću njega klasa automatski preuzima funkcionalnost postojeće klase. Nova klasa mora da definiše po čemu se razlikuje od postojeće.

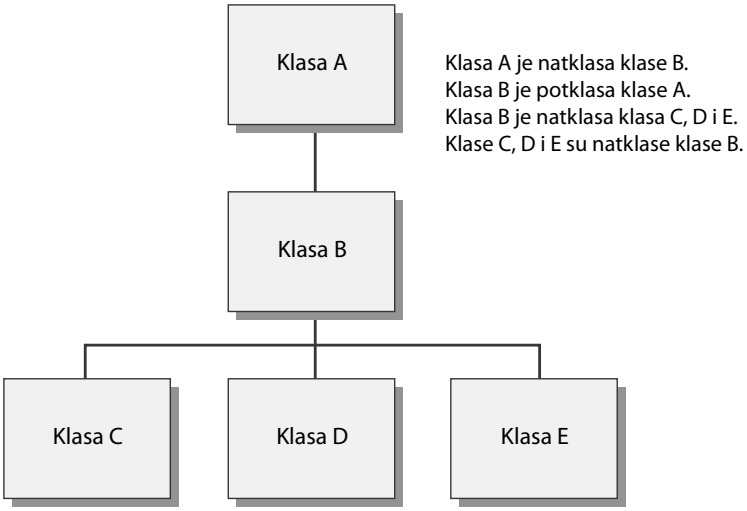
Kada je reč o nasleđivanju, sve klase, uključujući one koje kreirate i one u biblioteci Java Class, raspoređene su u hijerarhiji. Klasa koja nasleđuje druge klase naziva se *potklasa*. Klasa od koje se nasleđuje naziva se *natklasa*. Klasa može imati samo jednu natklasu, ali neograničen broj potklasa. Potklase nasleđuju sve atribute i ponašanje svojih natklasa.

U praktičnom smislu to znači da ako natklasa ima ponašanje i atribute koji su neophodni vašoj klasi, ne morate ponovo da definišite ponašanje, niti da kopirate kod da bi vaša klasa nasledila isto ponašanje i atribute. Vaša klasa u hijerarhiji automatski nasleđuje te elemente od svoje natklase, natklasa ih nasleđuje od svoje natklase i tako dalje. Vaša klasa postaje kombinacija sopstvenih funkcija i svih funkcija klasa koje su iznad nje u hijerarhiji.

Ovaj način nasleđivanja se može uporediti sa načinom na koji ste od roditelja nasledili osobine, poput visine, boje kose ili boje očiju. Neke od osobina ste nasledili od svojih roditelja, a oni su ih nasledili od svojih roditelja, itd.

Na slici 1.2 je prikazano kako je uređena hijerarhija klasa.

Slika 1.2
Hijerarhija klasa



Na vrhu hirerhije Java klasa je klasa `Object`. Ona sadrži ponašanje i atribute koji su neophodni za objekat u jeziku.

Sve klase nasleđuju od ove natklase. `Object` je opšta klasa u hijerarhiji.

Ona definiše ponašanje koje je nasleđeno od svih klasa u biblioteci Java Class Library.

Svaka klasa dalje u hijerarhiji postaje više prilagođena specifičnoj nameni.

Hijerarhija klasa definiše apstraktne koncepte na vrhu hijerarhije. Ti koncepti postaju konkretniji u dodatnim potklasama.

Kada kreirate novu klasu u Javi, možda ćete želeti da ona ima sve funkcije postojeće klase, bez nekih dodataka ili modifikacija koje ste sami kreirali. Na primer, poželećete novu verziju dugmeta (`Button`) koje će generisati zvuk kada se klikne na njega. To dugme bi se moglo nazvati `AudioButton`.

Da biste primili sve funkcije dugmeta (`Button`), a da ne morate ponovo da ga kreirate, možete da definišete svoju novu klasu `AudioButton` kao potklasu klase`Button`.

Zahvaljujući nasleđivanju, vaša klasa automatski nasleđuje ponašanje i atribute koji su definisani u klasi `Button`, ali i ponašanje i atribute koji su definisani u natklasi klase `Button`. Treba da brinete samo o stavkama po kojima se vaša nova klasa razlikuje od same klase `Button`, tj. o *potklasiranju.*

Potklasiranje je kreiranje nove klase koja nasleđuje ponašanje i atribute od postojeće. U potklasi je neophodan samo kod koji ukazuje na razlike u ponašanju i atributima između potklase i njenih natklasa.

Ako vaša klasa definiše potpuno novo ponašanje i nije potklasa druge klase, može naslediti direktno od klase `Object`.

Ako kreirate klasu koja ne proširuje natklasu, Java „pretpostavlja“ da nova klasa nasleđuje direktno od klase `Object`. Klasa `MarsRobot` koju ste kreirali ranije u ovoj lekciji nije odredila natklasu, tako da je ona potklasa klase `Object`.

Kreiranje hijerarhije klasa

Ako kreirate veliki skup klasa, logično je da vaše klase naslede od postojeće hijerarhije klasa i da same formiraju hijerarhiju. Ovo obezbeđuje vašim klasama nekoliko prednosti:

- Funkcionalnost koja je zajednička za više klasa može se svrstati u natklasu, pri čemu se omogućava da ona bude deo svih klasa koje se nalaze ispod natklase u hijerarhiji.
- Promene natklase se automatski odražavaju na sve njene potklase, na potklase potklasa, itd. Nije potrebno menjati ili ponovo kompajlirati niže klase, jer one dobijaju nove informacije pomoću nasleđivanja.

Na primer, zamislite da ste kreirali Java klasu za implementaciju svih funkcija istraživačkog robota (za ovo nije potrebno mnogo mašte).

Klasa `MarsRobot` je kreirana i uspešno funkcioniše. Vaš šef u agenciji NASA zahteva da kreirate Java klasu koja se zove `MercuryRobot`.

Ove dve vrste robota imaju slične karakteristike. Oba su istraživački roboti koji funkcionišu u neprijateljskim okruženjima i vrše istraživanje. Oni prate trenutnu temperaturu i brzinu.

Možda ćete odlučiti da prvo otvorite datoteku izvornog koda `MarsRobot.java`, da je kopirate u novu izvornu datoteku `MercuryRobot.java`, a zatim da izvršite neophodne izmene da bi novi robot obavio svoj posao. Ovo je loša odluka.

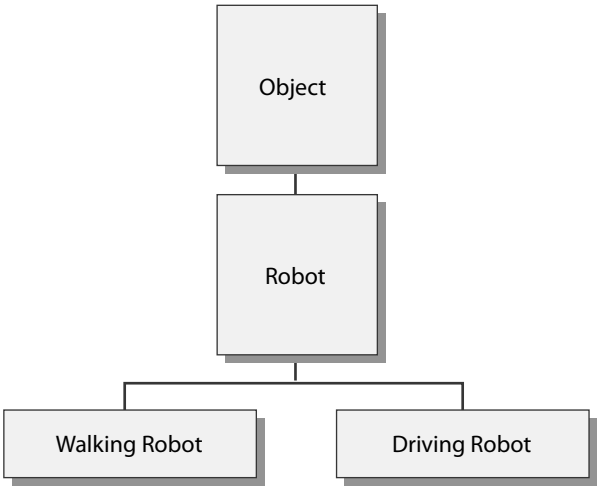
Mnogo je bolje da utvrdite zajedničku funkcionalnost klasa `MercuryRobot` i `MarsRobot` i da je organizujete u opštu hijerarhiju klasa. Ovo bi moglo da zahteva mnogo posla samo za klase `MarsRobot` i `MercuryRobot`, ali šta treba da učinite ako planirate da kasnije dodate klase `MoonRobot`, `UnderseaRobot` i `DesertRobot`? Faktorisanje uobičajenog ponašanja u natklasi koja se može upotrebiti više puta značajno smanjuje ukupan obim posla koji morate da obavite.

Da biste dizajnirali hijerarhiju klasa koja bi mogla poslužiti u tu svrhu, počnite dizajniranje sa vrha klase `Object`, koja je „vrhunac“ svih Java klasa.

Opšta klasa kojoj ti roboti pripadaju mogla bi se zvati `Robot`. Generalno, robot se može definisati kao autonomni uređaj za istraživanje. U klasi `Robot` definišite samo ponašanje koje omogućava da nešto postane uređaj, da se samokontroliše i da bude dizajnirano za istraživanje.

Ispod klase `Robot` mogu da budu klase `WalkingRobot` i `DrivingRobot`. Očigledno je da se ove dve klase, tj. roboti, razlikuju po tome što jedan robot hoda, a drugi se kreće pomoću točkova. Ponašanje robota koji hodaju može uključivati savijanje da bi nešto pokupili, saginjanje, trčanje i slično. Roboti koji se kreću pomoću točkova ponašali bi se drugačije. Na slici 1.3 je prikazana osnovna hijerarhija klase Robot.

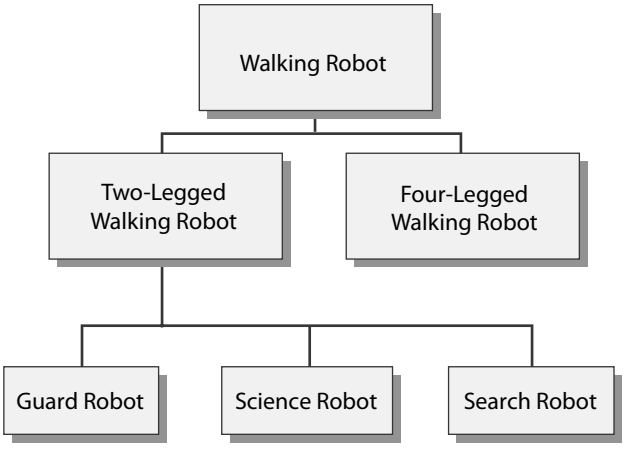
Slika 1.3
Osnovna hijerarhija klaseRobot



Sada hijerarhija može postati još određenija.

Klasa `WalkingRobot` može da ima nekoliko klasa: `ScienceRobot`, `GuardRobot`, `SearchRobot` i tako dalje. Kao alternativu, možete faktorisati još više funkcija i imati srednje klase za robote `TwoLegged` i `FourLegged`, sa različitim ponašanjem za svakog od njih (pogledajte sliku 1.4).

Slika 1.4
Dvonožni i četvoronožni roboti koji hodaju



Konačno je hijerarhija izvršena i imate mesto za `MarsRobot`. To može biti potklasa klase `ScienceRobot`, koja je potklasa klase `WalkingRobot`, koja je potklasa klase `Robot`, a ona je potklasa klase `Object`.

Gde pripadaju atributi kao što su `status`, `temperature` i `speed`? Pripadaju mestu na kojem se najprirodnije uklapaju u hijerarhiju klasa. Pošto svi roboti moraju da prate temperaturu svoje okoline, logično je da definišete temperaturu kao promenljivu instance u klasi `Robot`. Sve potklase će imati i tu promenljivu instance. Imajte na umu da morate da definišete ponašanje i atribut samo jednom u hijerarhiji, a automatski ih nasleđuje svaka potklasa.

> **NAPOMENA** Dizajn efikasne hijerarhije klasa uključuje mnogo planiranja i revizija. Dok pokušavate da unesete atribute i ponašanje u hijerarhiju, verovatno ćete pronaći razloge zbog kojih ćete premestiti neke klase na različita mesta u hijerarhiji. Cilj je da se smanji broj funkcija koje se ponavljaju (i reduntantni kod).

Nasleđivanje u akciji

Nasleđivanje u Javi funkcioniše mnogo jednostavnije nego u realnom svetu. Nisu potrebni testamenti ili sudovi prilikom nasleđivanja od „roditelja“.

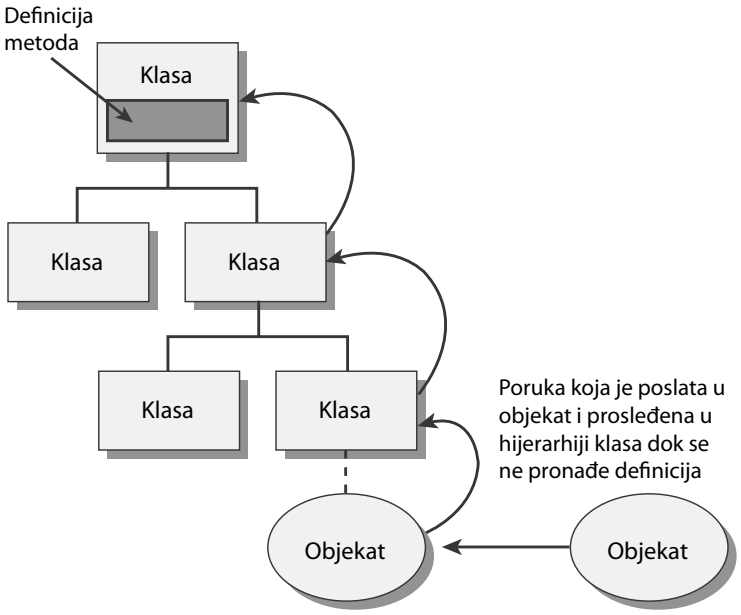
Kada kreirate novi objekat, Java prati svaku promenljivu definisanu za taj objekat i svaku promenljivu definisanu za svaku natklasu objekta. Dakle, sve klase se kombinuju da bi formirale obrazac za trenutni objekat, a svaki objekat dodaje informacije koje odgovaraju njegovoj situaciji.

Metodi funkcionišu na sličan način. Novi objekat ima pristup svim nazivima metoda svoje klase i pretklase. Ovo se utvrđuje dinamično kada se metod koristi u pokrenutom programu. Ako pozovete metod objekta, JVM prvo proverava klasu objekta za taj metod. Ako metod nije pronađen, JVM ga traži u natklasi te klase i tako dalje, sve dok se ne pronađe definicija metoda. Način na koji su metodi raspoređeni u hijerarhiji klasa je prikazan na slici 1.5.

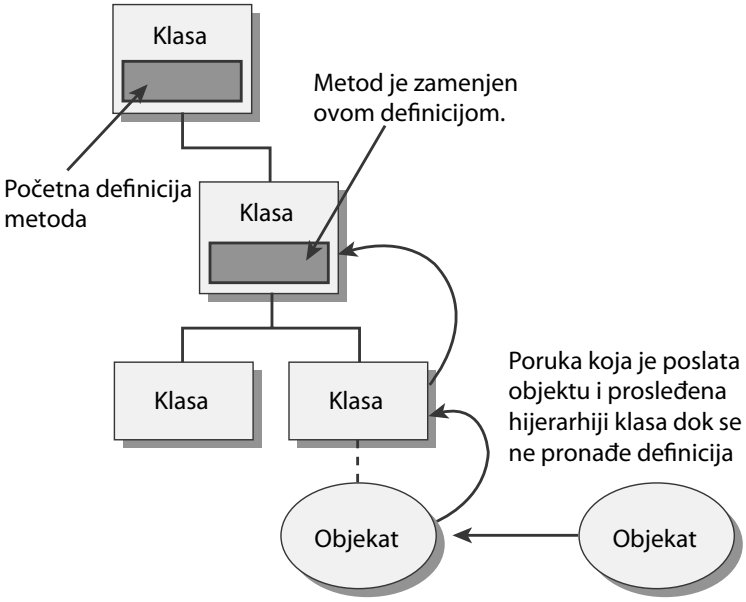
Situacija počinje da se komplikuje kada potklasa definiše metod koji odgovara nazivu i drugim aspektima definisanim u pretklasi. U ovom slučaju se prvo koristi definicija metoda (koja počinje od dna hijerarhije i „kreće“ se prema gore).

Zbog toga, možete da kreirate metod u potklasi koji sprečava upotrebu metoda u natklasi. Da biste to učinili, dodelite metodu isti naziv, tip povratka i argumente kao metodu u natklasi. Ovaj postupak, koji je prikazan na slici 1.6, zove se *redefinisanje*.

Slika 1.5 Kako su metodi smešteni u hijerarhiji klasa



Slika 1.6 Redefinisanje metoda



NAPOMENA Javin oblik nasleđivanja zove se *pojedinačno nasleđivanje*, zato što svaka Java klasa može imati samo jednu natklasu, iako konkretna natklasa može imati više potklasa.

U drugim objektno-orijentisanim programskim jezicima, kao što je C ++, klase mogu imati više natklasa i nasleđuju kombinovane promenljive i metode iz svih tih natklasa. To se naziva *višestruko nasleđivanje*. Java čini nasleđivanje jednostavnijim, tako što omogućava samo pojedinačno nasleđivanje.

Paketi

Programi koje ste kreirali tokom ove lekcije smešteni su u paket `com.java21days`. Paketi omogućavaju da grupišete srodne klase i interfejse. Oni eliminišu potencijalne „sukobe“ prilikom imenovanja klasa.

Klase u Javi mogu se pozivati pomoću kratkog naziva, kao što je `Object`, ili pomoću punog naziva paketa, kao što je `java.lang.Object`.

Vaše Java klase mogu ukazati na bilo koju klasu u paketu `java.lang` samo pomoću svojih kratkih naziva. Paket `java.lang` obezbeđuje osnovne jezičke funkcije, kao što su upravljanje znakovnim nizovima i matematičke operacije. Da biste koristili klase iz bilo kojeg drugog paketa, morate da ih referencirate eksplicitnim korišćenjem punog naziva paketa ili korišćenjem komande `import` za određeni paket. Upotreba komande `import` ukazuje Java kompajleru da treba da proveri taj paket prilikom pretvaranja kratkog naziva klase u pun naziv.

Postoji klasa `Color` u paketu `java.awt` u biblioteci Java Class. Ako želite da koristite ovu klasu u svom programu, možete da je referencirate pomoću punog naziva `java.awt.Color`.

Možete da olakšate referenciranje korišćenjem iskaza `import` u programu:

```
import java.awt.Color;
```

To omogućava da se klasa jednostavno referencira kao `Color`.

U iskazu `import` možete da koristite zvezdicu da biste naveli sve klase u paketu:

```
import java.awt.*;
```

Upotreba zvezdice kao znaka zamene omogućava proveru svake klase u paketu, pa se `Color` i sve ostale klase u paketu `java.avt` mogu referencirati pomoću svojih kraćih naziva.

NAPOMENA Mnogi Java programeri izbegavaju upotrebu iskaza `import` sa zvezdicom (*), pa koriste niz iskaza `import` koji identifikuju svaku klasu koja se koristi u programu. NetBeans to može učiniti automatski. Kada napišete liniju koda koja prvo koristi klasu kratkog naziva, NetBeans uređivač označava grešku na levoj ivici te linije. Kliknite na grešku da biste videli iskačući meni koji sadrži komandu za izvoz klase. Ako odaberete komandu, dodaje se iskaz `import`.

Paket za klasu se određuje iskazom `package`. Mnoge klase koje ćete kreirati u ovoj knjizi su smeštene u paketu `com.java21days`, kao što je:

```
package com.java21days;
```

Ovaj iskaz mora da bude prva linija programa. Kada je ta linija izostavljena, klasa će pripasti neimenovanom paketu, koji se zove *podrazumevani paket*.

Rezime

Ako ste se u ovoj lekciji prvi put susreli sa objektno-orijentisanim programiranjem, verovatno vam se ono čini teoretskim i pomalo komplikovanom. Mnogi ljudi su imali istu reakciju kada su se prvi put suočili sa „misterioznom“ metodologijom koja se zove OOP.

Budući da je vaš mozak prvi put pretrpan objektno-orijentisanim programskim konceptima i terminologijom, mogli biste se zabrinuti da u njemu neće biti prostora za ostatak Java jezika. Britanci imaju dva korisna saveta za situacije poput ove:

- Ne paničite!
- Budite mirni i nastavite dalje!

U ovom trenutku trebalo bi da imate osnovno znanje o klasama, objektima, atributima i ponašanjima. Takođe bi trebalo da poznajete promenljive i metode instance. Njih ćete iskoristiti u sledećoj lekciji.

Ostali aspekti objektno-orijentisanog programiranja, kao što su nasleđivanje i paketi, detaljnije su razmotreni u narednim lekcijama.

U čitavoj knjizi ćete koristiti objektno-orijentisano programiranje. Ne postoji drugi način pisanja programa na Java jeziku.

Nakon što pročitate prvih sedam lekcija, znaćete da koristite objekte, klase, nasleđivanje i sve druge aspekte metodologije.

Pitanja i odgovori

Pitanje - Metodi su funkcije koje su definisane unutar klasa. Ako metodi izgledaju kao funkcije i ponašaju se poput njih, zašto se ne zovu funkcije?

Odgovor – U nekim objektno-orijentisanim programskim jezicima metodi se zovu funkcije. U programskom jeziku C++ se zovu funkcije članova. Drugi objektno-orijentisani programski jezici razlikuju funkcije unutar i izvan „tela“ klase ili objekta, jer je u tim jezicima upotreba posebnih termina važna da biste razumeli kako se svaka funkcija izvršava. Pošto je razlika relevantna u drugim jezicima i pošto je termin *metod* sada u uobičajenoj upotrebi u objektno-orijentisanoj terminologiji, Java koristi i taj termin. Metodi se u funkcionalnim programskim jezicima zovu funkcije.

Pitanje – U čemu je razlika između promenljivih instance i metoda i njihovih „parnjaka“, promenljivih klase i metoda?

Odgovor – Skoro sve što radite u Java programu podrazumeva instance (koje se zovu i objekti), a ne klase. Međutim, neke skriptove ponašanja i atribute bolje je skladištiti u klasi, nego u objektu.

Na primer, klasa `Math` u paketu `java.lang` uključuje promenljivu klase `PI` koja sadrži približnu vrednost broja pi. Ova vrednost se ne menja, tako da nema razloga zbog kojih bi različitim objektima te klase bila potrebna pojedinačna kopija `PI` promenljive. Sa druge strane, svaki objekat `String` sadrži metod `length()` koji otkriva broj znakova u tom znakovnom nizu (`Stringu`). Ova vrednost može biti različita za svaki objekat te klase, tako da mora biti metod instance.

Promenljive klase zauzimaju memoriju dok Java program ne završi izvršavanje, tako da ih treba pažljivo koristiti. Ako promenljiva klase referencira objekat, taj objekat će takođe ostati u memoriji. Ovo je čest problem zbog kojeg program zauzima previše memorije i izvršava se sporo.

Pitanje - Kada Java klasa uvozi čitav paket, da li se povećava kompajlirana veličina te klase?

Odgovor - Ne. Upotreba termina uvoz u Javi je pogrešna. Rezervisana reč import ne dodaje kod jedne klase ili paketa u klasu koju kreirate. Umesto toga, lakše je referencirati unutar druge klase.

Jedina svrha uvoza je da skrati nazive klasa kada se koriste u Java iskazima. Bilo bi zamorno da u vašem kodu uvek morate da referencirate pune nazive klase, kao što su javax.swing.JButton i java.util.Random, umesto naziva JButton i Random.

Kviz

Pregledajte materiju u ovoj lekciji pomoću kviza koji se sastoji od tri pitanja.

Pitanja

1. Kako se drugačije naziva klasa?

   A. objekat

   B. obrazac

   C. instanca

2. Kada kreirate potklasu, šta morate definisati o klasi?

   A. Ništa. Sve je već definisano.

   B. elemente koje se razlikuju od elemenatanjene natklase

   C. sve o njoj

3. Šta predstavlja metod instance klase?

   A. atribute klase

   B. ponašanje klase

   C. ponašanje objekta koji je kreiran iz klase

Odgovori

1. B. Klasa je apstraktni obrazac koji se koristi za kreiranje objekata koji su slični jedni drugima.

2. B. Vi definišete kako se potklasa razlikuje od njene natklase. Elementi koji su slični su već automatski definisani zbog nasleđivanja. Odgovor A je tehnički tačan, ali ako su svi elementi u potklasi identični elementima u natklasi, nema potrebe kreirati potklase.

3. C. Metodi instance odnose se na ponašanje određenog objekta. Metodi klase odnose se na ponašanje svih objekata koji pripadaju toj klasi.

Sertifikaciona praksa

Sledeće pitanje je vrsta pitanja koje biste mogli očekivati da će vam biti postavljeno na testu za sticanje sertifikacije za Java programiranje. Odgovorite na ovo pitanje, ali nemojte ponovo da pregledate ovu lekciju.

Koji od sledećih iskaza je tačan?

A. Svi objekti koji su kreirani iz iste klase moraju da budu identični.

B. Svi objekti koji su kreirani iz iste klase mogu da imaju različite atribute.

C. Objekat nasleđuje atribute i ponašanje iz klase koja se koristi za njegovo kreiranje.

D. Klasa nasleđuje atribute i ponašanje iz svoje potklase.

Odgovor je dostupan na veb sajtu knjige, na adresi www.java21days.com. Posetite stranicu Lesson 1 i kliknite na link Certification Practice.

Vežbe

Da biste proširili vaše znanje o temama koje su obrađene u ovoj lekciji, probajte da uradite sledeće vežbe:

1. U metodu `main()` klase `MarsRobot` kreirajte drugi `MarsRobot` robot, pod nazivom `opportunity`, postavite njegove promenljive instance i prikažite ih.
2. Kreirajte hijerarhiju nasleđivanja za delove šahovske garniture. Odlučite gde boje promenljivih instance `startPosition`, `forwardMovement` i `sideMovement` treba da budu definisane u hijerarhiji.

Ako su vam potrebna, rešenja za vežbe možete da pronađete na veb sajtu knjige, na adresi www.java21days.com.