Prof. dr Slobodan Đukanović

JAVA
I
OBJEKTNO-ORIJENTISANO PROGRAMIRANJE

Recenzenti:
Prof. dr Igor Đurović
Prof. dr Vesna Popović-Bugarin
dr Goran Šuković

Izdavač:
Narodna knjiga, Podgorica
Miba books, Beograd

Za izdavača:
Miodrag Minić

Tehnička obrada:
Slobodan Đukanović

Dizajn korica:
Aleksandar Petrović

Godina izdanja:
2021.

Plasman za Crnu Goru:
+382 20 232 230, +382 69 025 535
e-mail: narodnaknjiga@t-com.me
web: https://narodna-knjiga.com

Plasman za Srbiju:
+381 11 713 8036, +381 65 278 3448
e-mail: mibabooks@gmail.com
web: https://mibabooks.rs

Štampa:
Neven, Beograd

Tiraž:
400 primeraka

CIP - Каталогизација у публикацији
Национална библиотека Црне Горе, Цетиње

ISBN 978-9940-25-151-2
COBISS.CG-ID 18078980

Slobodan Đukanović

JAVA
I
OBJEKTNO-ORIJENTISANO PROGRAMIRANJE

Narodna knjiga
Miba books

Sadržaj

Predgovor

Ovaj udžbenik je nastao kao rezultat dugogodišnjeg rada autora u okviru predmeta *Objektno-orijentisani dizajn softvera* koji se izučava na specijalističkim i master studijama Elektrotehničkog fakulteta Univerziteta Crne Gore. Udžbenik je samodovoljan i namenjen je širem krugu čitalaca. Mogu ga koristiti i početnici u programiranju, ali poznavanje osnovnih pojmova iz objektno-orijentisanog programiranja i programskih jezika C/C++ ili C# znatno olakšava savladavanje materije. Stoga je udžbenik prevashodno namenjen studentima koji imaju predznanje iz programiranja, programerima koji su se odlučili da izuče programski jezik Java, kao i ambicioznijim srednjoškolcima.

Poreklo Jave i sled događaja koji su doveli do pojave Jave opisani su u prvom poglavlju. Drugo poglavlje daje pregled osnova rada u Javi, tj. jezičkih elemenata (primitivnih tipova podataka, operacija, kontrole toka programa, rada sa metodama) i rada sa Java programima (kreiranje, kompajliranje i izvršavanje). Klasa, kao osnovna jedinica objektno-orijentisanog programiranja, je detaljno obrađena u trećem poglavlju. Čitaocima koji nemaju predznanje iz objektno-orijentisanog programiranja ovo poglavlje predstavlja dobar uvod u tematiku. Referencijski tipovi podataka u Javi (nizovi, prebrojivi tipovi, liste i stringovi) su obrađeni u četvrtom i petom poglavlju, dok su kolekcije obrađene u desetom poglavlju. Nasleđivanje i polimorfizam, kao ključni koncepti objektno-orijentisanog programiranja, su razmatrani u šestom i sedmom poglavlju. Ova dva poglavlje, kao i treće, imaju opšti značaj, jer se ovi koncepti sreću u svim objektno-orijentisanom jezicima. Obrada izuzetaka, koja je u Javi u potpunosti objektno orijentisana, je opisana u osmom poglavlju. Deveto poglavlje obuhvata rad sa tokovima podataka, fajlovima i opisuje način serijalizacije objekata u Javi. Višenitno programiranje je obrađeno u jedanaestom poglavlju.

Mnogi autori u svojim udžbenicima idu u jednu od dve krajnosti. Jedna krajnost je odvajanje teorijskog od praktičnog dela, što rezultuje suvoparnom teorijom kasnije ilustrovanom dugačkim programima. Uloga i značaj teorijskih koncepata u takvim programima mogu biti nedokučivi. Druga krajnost su prečeste ilustracije teorije kroz veoma kratke izvode iz programa, koje nije lako pretočiti u čitav projekat i na osnovu kojih nije lako steći širu sliku o predmetnom konceptu. Ovaj udžbenik se može staviti između ove dve krajnosti - svaki uvedeni koncept se odmah ilustruje na zaokruženom primeru. Kroz dugogodišnje autorovo iskustvo u domenu obrazovanja na polju programiranja, ovaj pristup se pokazao kao najefikasniji.

Svi programi u ovom udžbeniku potpuni su u smislu da mogu da se izvršavaju na računaru i mogu se preuzeti sa Web strane `https://slobodan.ucg.ac.me/books/javaOOP/` u formi zip arhive. Na istoj strani mogu se preuzeti i delovi pojedinih poglavlja udžbenika u pdf formatu.

Značajan trud je uložen da bi se eliminisale greške, ali se jedan broj sigurno potkrao. Stoga pozivam čitaoce da obrate pažnju na ispravnost sadržaja udžbenika, jer ću ih za svaku novootkrivenu grešku (pravopisnu, logičku ili sintaksnu) nagraditi šoljicom kafe. Zašto baš kafe? Jezik Java je dobio ime po indonežanskom ostrvu Javi, velikom proizvođaču kafe. Logotip ovog jezika je upravo šoljica kafe, nezvanični zaštitni znak programera. Pronađene greške možete uputiti autoru elektronskom poštom na adresu `slobdj@ucg.ac.me`, kao i sve sugestije koje mogu poboljšati ovaj materijal. Unapred im se radujem.

Čitaocima želim uspešno savladavanje programskog jezika Java i koncepata objektno-orijentisanog programiranja, i nadam se da će im ovaj udžbenik biti od koristi. Konačno, ne zaboravite tajnu učenja programiranja – programiranje se najbolje uči programirajući!

`Slobodan Đukanović`

Podgorica, jul 2021.

Oznake

Radi bolje preglednosti materijala, u knjizi koristimo fontove Calibri za tekst i `Consolas` za programski kôd, konzolne naredbe, nazive fajlova i njihovih putanja.

Radi bolje čitljivosti programskog kôda, koristimo različite boje, slično kako to rade integrisana razvojna okruženja za Javu:

- `zelenu za komentare`
- **`tamno ljubičastu (plus boldovano) za Javine ključne reči`**
- `plavu za tekst unutar stringa`
- `crnu za ostali programski tekst`

Kraći segmenti programskog kôda u tekstu, koji se koriste kao ilustrativni primeri za uvedeni pojam, označeni su crnom bojom.

Ispis programa ima plavu pozadinu:

```
Ovo je ispis našeg programa
```

Važne napomene i zapažanja imaju zelenu pozadinu:

> Ovo je važna napomena.

Ukoliko program sadrži više klasa, definicije klasa ćemo razdvajati isprekidanom linijom:

```
------------------------------------------------------------------------
```

1 Uvod u Java programiranje

1.1 Poreklo Jave

Java je srodnik jezika C++, koji je direktni potomak jezika C. Veći deo svojih osobina Java je preuzela iz ova dva jezika. Iz C-a, Java je preuzela sintaksu, dok je mnoge objektno-orijentisane (OO) koncepte preuzela iz C++. Generalno govoreći, svaka inovacija programskog jezika nastaje iz potrebe da se reši neki fundamentalni problem koji nije rešiv u postojećim jezicima. Nastanak Jave je pravi primerak tog pravila.

Navedimo redosled događaja koji su doveli do pojave Jave.

1.1.1 Programski jezik C: Nastanak modernog programiranja

Pojava jezika C uzdrmala je računarski svet. C je izazvao revoluciju u pogledu jednostavnosti pristupa hardveru. U dugom periodu bio je oslonac razvoja operativnih sistema i drugih sistemskih programa i kao takav je važan činilac tekuće tehnološke revolucije. Pored toga, našao je veliku primenu i kod mikroprocesorskih sistema.

Pre C-a, programeri su obično birali jezik namenjen određenoj aplikaciji. Na primer, BASIC je bio jednostavan jezik čija je svrha bila učenje programiranja, tj. prvi korak u savladavanju moćnijih jezika tog doba kakvi su bili FORTRAN ili ALGOL. FORTRAN je imao upotrebu u naučno-istraživačkom radu. COBOL je bio usmeren na rešavanje problema u ekonomiji, uključujući rad sa bazama podataka. Sa druge strane, programiranje u asembleru je davalo veliku slobodu programerima i rezultovalo je izuzetno efikasnim programima, ali je ovaj pristup teži od viših programskih jezika. Takođe, otklanjanje grešaka u asembleru je mukotrpan posao.

Do sedamdesetih godina XX veka, upotreba naredbe `goto` bila je vrlo česta u jezicima tog doba, što je narušavalo strukturnost programa i otežavalo njihovo održavanje. U ovom periodu, prestaje potreba za korišćenjem ove naredbe, odnosno princip strukturnog programiranja postaje programerski standard. Nametnula se težnja da se kreiraju programski moduli (funkcije, rutine, potprogrami) koje je moguće koristiti u drugim programima. Sa druge strane, računarski hardver je postao uveliko dostupan programerima, što je donelo slobodu u eksperimentisanju i omogućilo programerima da počnu da prave svoje alate. Sve je vapilo za novim jezikom koji bi predstavljao kvalitativan skok u aktuelnom programiranju. Jezik C je bio upravo to.

Programski jezik C je stvorio Dennis Ritchie. C je nastao iz jezika B (kreator Ken Thompson), a ovaj je nastao iz jezika BCPL (kreator Martin Richards). Tokom više godina, standard jezika C je bio onaj koji se koristio na UNIX-u, i koji je opisan u knjizi *The C programming language* (Prentice Hall, 1978), čiji su autori Brian Kernighan i Denis Ritchie. Jezik C je standardizovan od strane ANSI instituta 1989.

Smatra se da nastanak C-a predstavlja početak modernog doba u programskim jezicima. To nije samo programski, već i *programerski* jezik. Za razliku od jezika pre pojave C-a, koji su obično nastajali kao rezultat akademskih eksperimenata, C su zamislili, kreirali i razvili programeri, pa zato on održava programerski pristup programiranju. C su stvorili ljudi koji su ga i koristili. Time je ovaj jezik vrlo brzo pronašao svoje mesto među programerima.

1.1.2 C++: Korak više

Početkom osamdesetih godina, programeri su široj javnosti otkrili mogućnosti računara, pa su zahtevi korisnika računara postajali sve veći, a samim tim je i rastao obim posla poveravan računarima. Broj i veličina projekata na kojima su programerima radili su rasli i došli do tačke kada strukturirano programiranje ne može da izađe u susret svim zahtevima. Naime, pisanje, testiranje i ispravljanje programa je postao mukotrpan posao zbog čestih izmena programa. Često se dešavalo da je mala promena u specifikaciji softvera iziskivala znatne promene u kôdu nekog modula ili njegovo ponovno pisanje. Zatim, veze tog modula sa ostalim modulima programa su bile veoma složene, pa je promena jednog modula zahtevala značajnu promenu čitavog programa. Problem je, dalje, predstavljalo i testiranje novog softvera, odnosno otklanjanje grešaka koje bi se pojavile kao posledica menjanja pojedinih modula.

Odgovor na softversku krizu predstavljalo je OO programiranje. Iako se kao koncept pojavilo znatno pre softverske krize (jezik Simula je bio prvi OO jezik), sa njenom pojavom uočena je neophodnost ovog koncepta. Naime, američka komunikaciona kompanija AT&T (American Telephone and Telegraph) je trebala da početkom osamdesetih godina XX veka realizuje određeni projekat. Uočeno je da realizacija ovog projekta zahteva značajne prepravke postojećih programa. Bjarne Stroustrup je uočio da korišćenjem strukturiranog programiranja nije moguće rešiti ovaj problem. Stoga je 1979. kreirao OO verziju programskog jezika C, koji je prvobitno nazvan *C sa klasama*, a 1983. je preinačen u C++. Prvi C++ kompajler se pojavio 1986, a standardizovan je 1997. Ovo je danas vrlo popularan programski jezik.

C++ proširuje C dodajući mu OO osobine. Pošto se temelji na C-u, C++ nasleđuje sve njegove osobine i prednosti, što i jeste razlog uspeha ovog jezika. C++ nije stvoren kao potpuno nov programski jezik, već sa ciljem da se poboljša jezik koji se pokazao izuzetno uspešnim.

1.2 Nastanak Jave

OO programiranje je krajem osamdesetih i početkom devedesetih uzelo maha, i činilo se da je pronađen savršen programski jezik koji će izaći u susret svakom programerskom izazovu. Ipak, kao mnogo puta ranije, javila se potreba da se računarski jezici pomaknu za još jedan stepen u svojoj evoluciji. U to doba, Internet već poprima svoj sadašnji oblik, što će začeti novu revoluciju na polju programiranja.

Javu je prvobitno, 1991. godine, koncipirao James Gosling sa svojim kolegama iz korporacije Sun Microsystems, Inc. Prvo ime jezika bilo je Oak, koje je 1995. godine preinačeno u Java. Iznenađujuće, ali glavni pokretač razvoja Jave nije bio Internet, već potreba za jezikom nezavisnim od računarske platforme, koji bi se mogao koristiti za programiranje različitih elektronskih uređaja u domaćinstvu (mikrotalasne pećnice, frižideri i slično). Kao kontroleri ovih uređaja koriste se različiti procesori, pa je upotreba jezika C i C++ za njihovo programiranje ograničena, jer su ovi jezici, kao i većina drugih, usmereni na određeni procesor. Iako se C++ program može prevesti za bilo koji procesor, za to je potreban kompajler namenjen konkretnom procesoru. Kompajleri su skupi i sporo se prave, što implicira potrebu za lakšim i jeftinijim rešenjem. Java je rođena upravo iz pokušaja Gosling-a i saradnika da kreiraju prenosivi jezik koji ne zavisi od platforme i čiji se kôd može izvršavati na različitim procesorima i u različitim okruženjima.

U to doba, pojavljuje se ključni faktor za razvoj Jave, World Wide Web (WWW). Da se WWW pojavio u neko drugo vreme, pretpostavka je da bi se Java koristila samo za programiranje kućne elektronike. Međutim, sa pojavom WWW-a, Java se probija u prvi plan računarskih jezika, jer su za WWW bili potrebni prenosivi programi. WWW i Internet su povezali mnoštvo različitih računara i operativnih sistema, pa je potreba za prenosivim programima postala urgentna. Isti problem zbog koga je Java prvobitno smišljena se pojavio i na Internetu, samo u mnogo većem obimu. Ovime se fokus primene Jave, sa kućne elektronike, premešta na programiranje za Internet. Dakle, iako je početni motiv razvoja Jave bio kreiranje jezika nezavisnog od platforme, za njegov uspeh na visokom nivou najzaslužniji je Internet.

Nije slučajno to što Java većinu svojih osobina duguje jezicima C i C++. Autori Jave su znali da će ova dva široko rasprostranjena jezika uticati da programeri vrlo brzo prihvate Javu. Sa C i C++ Java ima zajedničke one osobine koje su ova dva jezika učinile ekstremno popularnim. Javu su oblikovali, testirali i unapređivali profesionalni programeri. To je jezik utemeljen na potrebama i iskustvima osoba koje su ga stvorile, zbog čega on predstavlja jezik programera.

Sličnost između Jave i jezika C++ navodi na pogrešnu pretpostavku da Java predstavlja C++ verziju za Internet. Java se od jezika C++ razlikuje i po filozofiji i u praktičnom smislu. Iako je Java nastala pod uticajem jezika C++, ona ne predstavlja poboljšanu verziju tog jezika,

već je napravljena da reši drugačiji skup problema od onih kojima je namenjen C++. Oba jezika će postojati uporedo još mnogo godina.

1.3 Java i Internet: Bezbednost i prenosivost

Pojavom Jave značajno su poboljšani bezbednost i prenosivost programa na Internetu. Objekti koji kruže Internetom se ugrubo mogu podeliti na pasivne podatke (npr. elektronska pošta) i dinamičke, aktivne programe. Sa stanovišta bezbednosti, dinamički programi, iako mogu biti vrlo korisni, predstavljaju veliki rizik.

Java je značajno uticala na poboljšanje bezbednosti Interneta. U vezi s tim, pomenimo Java aplete, kao posebnu vrstu Java programa namenjenih distribuciji preko Interneta i automatskom izvršavanju u Web pretraživačima koji podržavaju Javu. Preuzimanje Java apleta je potpuno bezbedno. Java obezbeđuje zaštitu tako što izvršenje programa ograničava na svoje okruženje, ne dozvoljavajući mu pristup drugim resursima računara. Mogućnost preuzimanja apleta bez bojazni od štete za mnoge je predstavljala najveću novinu koju je Java donela.

Za programe koji treba da se dinamički preuzimaju na različitim platformama povezanim sa Internetom, mora postojati mogućnost generisanja prenosivog izvršnog kôda. Isti mehanizam kojim se ostvaruje bezbednost izvršavanja apleta omogućava i njihovu prenosivost. Probleme bezbednosti i prenosivosti Java rešava vrlo elegantno i efikasno, što će biti objašnjeno u nastavku.

1.4 Bajt kôd

Java obezbeđuje bezbednost i prenosivost tako što Java prevodilac ne generiše izvršni kôd, već tzv. bajt kôd (eng. *bytecode*), optimizovan skup instrukcija kojeg u trenutku izvršavanja interpretira Javin izvršni sistem poznatiji kao Javina virtuelna mašina (eng. *Java virtual machine*, JVM). Dakle, JVM je interpreter bajt kôda.

Prevođenje Java programa u bajt kôd omogućuje lakše izvršavanje u različitim okruženjima, jer je za različite platforme potrebno napraviti samo različite virtuelne mašine. Iako se Javine virtuelne mašine razlikuju na različitim platformama, sve one razumeju isti Javin bajt kôd, tj. mogu da izvrše svaki Javin program. Kada bi se Java programi direktno prevodili u izvršni kôd, onda bi morale postojati različite verzije programa za svaki tip procesora, što nas vraća na početni problem prenosivosti. Izvršavanje bajt kôda pomoću JVM-a predstavlja najjednostavniji način pravljenja prenosivih programa.

Činjenica da JVM, a ne neposredno procesor izvršava Java programe čini te programe bezbednijim. Svojim delovanjem, tj. upravljanjem izvršenja programa u potpunosti, JVM

sprečava da program koji se izvršava utiče na okolni sistem. Bezbednost je dodatno povećana određenim ograničenjima koja postoje u Javi.

Po pravilu, programi prevedeni u izvršni oblik se brže izvršavaju od programa koji se interpretiraju. Međutim, u Javi, ova razlika u vremenu izvršavanja nije velika. Razlog tome je činjenica da je bajt kôd u velikoj meri optimizovan. Iako je Java izvorno namenjena interpretiranju, tehnički ne postoji prepreka da se, radi bržeg izvršavanja, bajt kôd prevede u izvršni kôd. Zato je firma Sun ubrzo posle objavljivanja Jave ponudila svoj JIT (Just In Time) prevodilac pod imenom HotSpot. Kada JVM uključuje JIT prevodilac, tokom izvršavanja se određeni delovi bajt kôda prevode u izvršni kôd. Ipak, nije moguće ceo Java program prevesti u izvršni kôd, zbog različitih provera koje Java sprovodi tokom izvršavanja programa. Zato Java prevodi kôd po potrebi, tokom izvršavanja, i to samo one delove koji će bitno ubrzati izvršavanje. Preostali deo kôda se i dalje interpretira. Ovaj način "prevođenja po potrebi" pruža bolje performanse izvršavanja, dok se bezbednost i prenosivost programa ne smanjuju, jer izvršavanjem i dalje upravlja JVM.

1.5 Osobine Jave

Pored prenosivosti i bezbednosti, Javu karakteriše niz drugih osobina:

- jednostavnost
- objektna orijentisanost
- robustnost
- nezavisnost od platforme
- interpretiranost
- visoka efikasnost
- višenitnost
- distribuiranost
- dinamičnost.

U nastavku ukratko opisujemo svaku od ovih osobina.

Jednostavnost: Java je koncipirana tako da programeri mogu lako da je nauče i efikasno koriste. Pod uslovom da imate određeno iskustvo u programiranju i da poznajete osnovne pojmove OO programiranja, učenje Jave neće biti težak zadatak. Najpoželjnije bi bilo da znate jezik C++ ili C#, sa kojih bi prelazak na Javu bio relativno bezbolan.

Objektna orijentisanost: Iako je bila pod velikim uticajem svojih prethodnika, Java kôd nije dizajniran da bude kompatibilan sa bilo kojim drugim jezikom. Autori Jave su imali odrešene ruke u kreiranju Jave, što je za rezultat imalo čist, upotrebljiv, pragmatičan pristup objektima. Java je slobodno pozajmljivala principe iz mnogih objektnih softverskih okruženja koja su nastala tokom prethodnih decenija. Objektni model Jave je jednostavan

i lako se proširuje, dok primitivni tipovi nisu realizovani kao objekti radi boljih performansi izvršavanja.

Robusnost: Pri projektovanju Jave jedan od prioriteta je bila sposobnost da se napravi robustan program. U cilju pouzdanosti izvršenja programa, Java ograničava programera u nekoliko ključnih aspekata, primoravajući ga da ispravlja greške u ranoj fazi. Sa druge strane, Java nas oslobađa brige oko najčešćih grešaka. Pošto je Java strogo tipiziran jezik, ona proverava programski kôd u trenutku njegovog prevođenja, ali i tokom izvršavanja. Mnoge neuhvatljive greške, one koje se javljaju u situacijama koje je teško simulirati, u Javi se ne mogu ni napraviti. Prisetimo se dva glavna razloga otkazivanja programa: greške pri upravljanju memorijom i loša obrada izuzetaka. U tradicionalnim programskim jezicima, upravljanje memorijom nije jednostavan posao. Na primer, u jezicima C i C++, programer mora da vodi računa o zauzimanju i oslobađanju dinamički dodeljene memorije, što je čest izvor fatalnih grešaka, jer programeri zaboravljaju da oslobode (delociraju) dodeljenu memoriju ili delociraju memoriju koja se još uvek koristi u programu. Ovi problemi kod Jave ne postoje, jer ona sama dodeljuje i oslobađa memoriju. Java obezbeđuje OO obradu izuzetaka - sve greške koje nastaju tokom izvršenja Java programa se istim programom mogu obraditi.

Nezavisnost od platforme: Osnovni problem pri dizajnu Jave bilo je stvaranje prenosivog kôda koji će trajno raditi. Za program koji je napisan danas i koji radi, niko ne može garantovati da će raditi i sutra, čak i na istoj mašini. Operativni sistemi se stalno poboljšavaju, kao i procesori, a kada se sve iskombinuje sa promenama u osnovnim resursima sistema, može se dogoditi da program više ne radi kako treba. Slogan dizajnera Jave bio je "Napiši jednom, izvršavaj bilo gde, bilo kad i zauvek", što je najvećim delom i postignuto.

Interpretiranost i visoka efikasnost: O ovome je bilo reči kada smo govorili o bajt kôdu. Da ponovimo, bajt kôd se može izvršavati na svakom računaru koji ima JVM. U kombinaciji sa JIT prevodiocem, delovi bajt kôda se prevode u mašinski kôd, čime se postižu bolje performanse. S obzirom da JVM upravlja celokupnim procesom izvršavanja, ovakvo izvršavanje ne umanjuje nezavisnost kôda od platforme.

Višenitnost: Java podržava višenitno programiranje, koje omogućava da program istovremeno radi više stvari. JVM ima elegantno i potpuno rešenje za sinhronizovanje više procesa. Višenitnost u Javi je vrlo važna da bi se izašlo u susret realnim zahtevima pravljenja interaktivnih mrežnih programa.

Distribuiranost: Java je posebno namenjena distribuiranom okruženju Interneta, jer lako rukuje protokolima TCP/IP. Pristupanje Internet adresi se ne razlikuje bitno od pristupanja fajlu. Java podržava i daljinsko izvršavanje procedura (eng. *Remote Method Invocation*), što znači da Java program može da izvršava procedure koje se nalaze na bilo kojoj mrežnoj adresi.