Laslo Kraus

PROGRAMSKI JEZIK

Java

sa rešenim zadacima

(Java SE 13)

AKADEMSKA MISAO
Beograd, 2019

Laslo Kraus

PROGRAMSKI JEZIK JAVA SA REŠENIM ZADACIMA
Treće, prerađeno izdanje

Recenzenti
Dr Igor Tartalja
Dr Đorđe Đurđević

Izdaje i štampa
AKADEMSKA MISAO
Bulevar kralja Aleksandra 73, Beograd

Lektor
Anđelka Kovačević

Dizajn naslovne strane
Zorica Marković, akademski slikar

Tiraž
300 primeraka

ISBN 978-86-7466-807-8

Dušanki i Katarini

Predgovor

Ova knjiga predstavlja udžbenik za programski jezik *Java* za široki krug čitalaca. Knjigu mogu da koriste i početnici u programiranju, ali poznavanje osnovnih pojmova iz objektno-orijentisanog programiranja i programskih jezika C/C++ ili C# znatno olakšava da se savlada materija iz ove knjige.

Programski jezik *Java* izložen je u obimu koji može da zadovoljava i naprednije neprofesionalne programere. Od ogromne standardne biblioteke tipova (klasa), koja prati jezik *Java*, objašnjeni su samo delovi koji su potrebni za efikasno programiranje pri rešavanju većine problema relativno visoke složenosti.

Jedini način da se nauči neki programski jezik jeste da se pišu programi na njemu. Ova knjiga je u potpunosti podređena tom osnovnom načelu.

Uvodno poglavlje 1 sadrži informacije korisne pre svega za početnike: osnovne pojmove objektno-orijentisanog programiranja, glavne osobine programskog jezika *Java* i načine obrade programa u tekstualnom i grafičkom okruženju.

U poglavlju 2 obrađeni su podaci koji su predmet obrade u programima. Objašnjen je pojam tipa podataka, načini definisanja i korišćenja skalarnih brojčanih, znakovnih, logičkih i tekstualnih podataka. Da bi što pre mogli da se pišu potpuni programi u ovom poglavlju su obrađene i ulazna i izlazna konverzija brojčanih podataka. Naime, programi koji ne čitaju početne podatke i ispisuju rezultate nemaju nikakvu upotrebnu vrednost.

Poglavlje 3 posvećeno je operatorima i načinima sastavljanja izraza pomoću njih. Tu su obrađeni operatori jezika *Java* koji čitaocu s prosečnim matematičkim obrazovanjem ne bi trebalo da budu strani. Prikazivanje nekih specifičnijih operatora je odloženo za kasnije. Kao proširenje skupa operatora dat je i pregled važnijih standardnih bibliotečkih funkcija koje postoje u jeziku *Java*. Odabrane su funkcije koje se odnose na opštepoznate pojmove iz matematike i na obradu već objašnjenih tipova podataka.

U poglavlju 4 prikazane su naredbe koje predstavljaju jedinične obrade u programima. Pored prostih naredbi obrađene su složene naredbe jezika *Java* koje omogućavaju sastavljanje složenih programa na efikasan i pregledan način: uslovno izvršavanje (selekcije), višestruko izvršavanje (ciklusi) delova programa i upravljačke naredbe (skokovi).

Iz matematike dobro poznati nizovni tipovi: vektori, matrice i višedimenzionalni nizovi obrađeni su u poglavlju 5.

Izučavanje objektno-orijentisanog programiranja počinje u poglavlju 6 o klasama. Klase su složeni tipovi podataka koji programerima omogućuju modeliranje objekata iz realnog života. Sastoje se od polja čije vrednosti čine stanja primeraka klasa i metoda koje definišu

moguće načine korišćenja tih polja. Primerci klasa su podaci klasnih tipova i nazivaju se objektima. Jedan od važnih elemenata ispravnog rada programa je pravilna inicijalizacija objekata.

Programski jezik *Java* omogućava grupisanje srodnih klasa u pakete. U poglavlju 7 obrađeni su paketi: njihovo formiranje i korišćenje. Prikazani su i važniji elementi dva najčešće korićena standardna paketa.

Poglavlje 8 posvećeno je sledećem fundamentalnom konceptu objektno-orijentisanog programiranja, nasleđivanju. U grupama podataka koje su predstavljene klasama mogu da se uoče podgrupe sa nekim specifičnijim osobinama. Te podgrupe, predstavljene potklasama, nasleđuju osobine širih grupa, natklasa. Član podgrupe uvek može da se koristi i kao član opštije nadgrupe.

Interfejsi su specijalni apstraktni tipovi koji samo definišu određene osobine koje klase koje ih ostvaruju treba da poseduju, a ne i način kako se te osobine ostvaruju. Obrađeni su u poglavlju 9.

U poglavlju 10 obrađeni su ugnežđeni tipovi, klase i interfejsi koji su definisani unutar drugih klasa i interfejsa. Tu spadaju i lokalne klase i njihove specijalne vrste: bezimene klase i lambda izrazi.

Jezik *Java* omogućava da rukovanje izuzecima (greškama) vremenski ne opterećuje program sve dok se izuzetak ne pojavi. Kada se izuzetak pojavi, kontrola se automatski predaje jednom od rukovalaca izuzecima koje je definisao programer. Način rukovanja izuzecima prikazan je u poglavlju 11.

U poglavlju 12 objašnjeni su generički tipovi i metode. Generičke klase i metode omogućavaju pisanje klasa i metoda za obradu podataka, čiji tipovi nisu poznati u vreme njihovog pisanja, na bezbedan način sa stanovišta provera ispravnog korišćenja tipova podataka. Generički interfejsi, na sličan način, opisuju očekivane osobine budućih klasa u uslovima kada tipovi pojedinih korišćenih podataka nisu poznati. Na kraju ovog poglavlja su ukratko objašnjene i dve najčešće korišćene standardne generičke klase za rad sa zbirkama podataka.

Nabrajanja su specijalni tipovi podataka čije se jedine vrednosti (objekti) navode u definiciji nabrajanja. Njima je posvećeno poglavlje 13.

U poglavlju 14 izloženi su osnovni koncepti konkurentne obrade, obrade koja se sastoji od istovremenog odvijanja više tokova kontrole, više istovremenih obrada. Ti tokovi kontrola nazivaju se niti, čija je korektna sinhronizacija osnova dobrog programa s konkurentnom obradom.

Savremeni stil komunikacije korisnika s programima je preko prozora na grafičkoj korisničkoj površi (*GUI*). Rad s prozorima ne sadrži nijedan novi element jezika *Java*, već samo korišćenje gotovih standardnih bibliotečkih klasa (tipova). Ipak, zbog široke upotrebe prozora u današnjim programima, u poglavlju 15 prikazani su osnovni elementi izrade programa korišćenjem grafičke korisničke površi.

Jezik *Java* vrlo je složen. Nisu svi detalji neophodni svakome, a naročito ne početnicima. Odeljci, ponegde cela poglavlja, u knjizi koji mogu da se preskoče u prvom čitanju, bilo zbog složenosti, bilo zbog manjeg značaja, obeleženi su sa ∆.

(∆ Pasusi unutar odeljaka koji mogu da se preskoče u prvom čitanju obeleženi su na ovde prikazan način. ∆)

Odeljci i pasusi, uglavnom, mogu se preskakati nezavisno. Ako se preskoči jedan odeljak ili pasus, to ne znači da mora da se preskoči i neki kasniji, obeleženi odeljak ili pasus.

Rukovodeći se već istaknutim načelom o učenju jezika, ova knjiga, pored manjih primera u toku izlaganja, na kraju svakog poglavlja sadrži nekoliko rešenih zadataka. Više zadataka može da se nađe u autorovoj zbirci *Rešeni zadaci iz programskog jezika Java* (videti [8]). Kroz zadatke u ovoj knjizi i u zbirci skreće se pažnja na neke specifičnosti jezika *Java* i daju se ideje za elegantno i efikasno rešavanje nekih problema.

Ova knjiga je više nego udžbenik za programski jezik *Java*. Kroz rešene zadatke prikazane su i primene osnovnih elemenata objektno-orijentisanog programiranja, a na primerima koji se uopšteno sreću u računarskoj tehnici: obradi redova, stekova, listi, tekstova, datoteka itd. Posebna pažnja posvećena je i inženjerskim aspektima programiranja. Nije dovoljno da program rešava zadati problem, već treba da bude i „lep", razumljiv, da se lako održava, da troši što manje memorije i da bude što efikasniji.

Svi programi koji se nalaze u ovoj knjizi potpuni su u smislu da mogu da se izvršavaju na računaru. Provereni su na nekoliko računara korišćenjem nekoliko različitih prevodilaca za jezik *Java*.

Izvorni tekstovi svih programa iz ove knjige mogu da se preuzmu sa Interneta, sa adrese `home.etf.rs/~kraus/knjige/`. Na istoj adresi objaviće se ispravke eventualnih, naknadno otkrivenih grešaka u knjizi. Svoja zapažanja čitaoci mogu da upute autoru elektronskom poštom na adresu `kraus@etf.rs`.

Laslo Kraus

Beograd, *septembar 2019.*

Sadržaj

1 Uvod

1.1 O programskom jeziku *Java*

Programski jezik *Java* viši je programski jezik opšte namene koji u potpunosti podržava paradigmu objektno-orijentisanog programiranja.

Java je više od programskog jezika koji ima određenu sintaksu i semantiku. Ona je pravo programsko okruženje koje pomoću ogromne biblioteke standardnih programskih modula podržava obrade bez kojih danas ne može da se zamisli nijedan ozbiljan program. Tu spadaju komunikacija sa korisnikom preko prozora, višenitna obrada, rad preko *Interneta*, troslojna arhitektura rada s bazama podataka itd. Standardi programskih jezika do pojave jezika *Java* te elemente nisu obuhvatali. Svaki proizvođač je uvodio nestandardna rešenja za njih, ograničavajući time prenosivost programa s jedne na drugu platformu.

Programski jezik *Java* je srodnik jezika *C++* koji je direktni potomak jezika *C*.

Pogramski jezik *C* pojavio se početkom 1970-tih godina i u osnovama je promenio dotadašnji, nesistematizovani, stil programiranja. Kroz podršku za *strukturirano programiranje* obezbedio je da mogu da se napišu, razumeju i održavaju znatno veći programski sistemi nego pre. Pošto se u centru pažnje nalaze postupci koji se primenjuju na podatke, ta tehnika programiranja naziva se i *proceduralno programiranje*.

Pošto su jezik *C* projektovali programeri koji su i sami pisali programe, drugi programeri su ga rado prihvatili i uskoro je postao jedan od najrasprostranjenijih programskih jezika, koji se koristi i danas. Za autora jezika *C* smatra se Denis Riči (*Dennis Ritchie*) iz Belovih laboratorija (*Bell Laboratories*).

Početkom 1980-tih godina računarski programi narasli su do takvih veličina da su se tehnike proceduralnog programiranja pokazale neefikasnim. Rešenje se našlo u paradigmi *objektno-orijentisanog programiranja*, kod koje se u centru pažnje nalaze podaci (objekti) na koje se primenjuju neki postupci. Od programskih jezika koji su projektovani da podržavaju nove tehnike programiranja najveći uspeh imala je nadogradnja jezika *C*, koja je danas poznata pod imenom *C++* (++ je u jeziku *C* operator povećavanja!). Preko 95% jezika *C* usvojeno je bez izmena i u jeziku *C++*.

C++ je hibridni jezik u smislu što, pored novijeg objektno-orijentisanog programiranja, podžava i starije proceduralno programiranje.

Za autora jezika *C++* smatra se Bjarn Strostrup (*Bjarne Stroustrup*).

Početkom 1990-tih godina pojavila se potreba za programima koji su u prevedenom obliku nezavisni od platforme, tj. mogu da se izvršavaju na bilo kom računaru bilo kog proizvođača. Radilo se prvenstveno o programima za potrebe industrijske elektronike za upravljanje kućnim aparatima.

Dotad postojeći viši programski jezici bili su, bar teorijski, prenosivi na nivou izvornog teksta programa. Taj tekst morao je zasebno da se prevede u izvodljivi oblik za svaku vrstu računara. To bi u uslovima naraslog broja različitih platformi tražilo izradu velikog broja prevodilaca sa višeg programskog jezika na mašinski jezik računara. Trebalo je projektovati programski jezik koji omogućava da prevedeni oblik programa bude nezavisan od platforme na kojoj se izvršava.

Zbog toga je grupa programera u firmi *Sun Microsystems, Inc.* s Džejmsom Gazlingom (*James Gosling*) na čelu započela projektovanje novog programskog jezika koji je trebalo da bude lak za korišćenje, pouzdan i nezavisan od platforme na nivou prevedenog oblika. Taj jezik se u početku zvao *Oak* (hrast), a ime *Java* dobio je 1995. godine.

Nekako baš u vreme razvijanja jezika *Java* počelo je intenzivno širenje *Interneta* i posebno njegove usluge *World Wide Web* za pružanje najraznovrsnijih informacija, organizovanih u takozvane „veb stranice", sa specijalnih servera do proizvoljnih klijenata širom sveta. Veb stranice su u početku bile statične: sadržavale su samo tekstove i slike (u početku nepokretne, kasnije i pokretne). Za dinamični sadržaj, koji podrazumeva i interakciju s klijentima, trebalo je u veb stranice ugraditi male programe. Ti programi, pošto bi se izvršavali na računaru klijenta, morali su biti nezavisni od platforme i bezbedni da ne naprave štetu na klijentskom računaru.

Programski jezik *Java* imao je upravo te osobine, pa je današnju popularnost stekao kroz takozvane aplete (*applets*), male programe koji se ugrađuju u veb stranice. U međuvremenu apleti kao dinamičan sadržaj veb stranica prevaziđeni su, ali i nove tehnologije programiranja za *Internet* u velikoj meri se oslanjaju na jezik *Java*.

Kao programski jezik opšte namene, *Java* se koristi i za izradu samostalnih programa koji se nazivaju *aplikacije*.

Programski jezik *Java* nema zvaničan standard. Kao *de facto* standard uzima se specifikacija koju je u početku publikovala firma *Sun Microsystems, Inc*., a u novije vreme firma *Oracle*, koja je kupila firmu *Sun*.

Jezik je još u intenzivnom razvoju i publikovan je veći broj izdanja. Prvo izdanje je bila verzija 1.0, ali je ubrzo zamenjena verzijom 1.1 koja nije donela bitne izmene. Verzija 1.2 donela je značajne izmene i od tada se jezik skraćeno zvala *Java* 2, a punim imenom *Java 2 Platform Standard Edition* (*J2SE 1.2*). Sledile su verzije 1.3 i 1.4 koje su donele samo manje izmene. Verzija 1.5 donela je ogromne promene, pa da bi se to naglasilo, ta verzija je proglašena verzijom 5 (*J2SE 5*). Počev od naredne verzije počele su da se koriste oznake oblika *Java SE x*. Aktuelna verzija u vreme pisanja ove knjige je *Java SE 13*.

1.2 Uvod u objektno-orijentisano programiranje

U svakom računarskom programu mogu da se uoče dve grupe elemenata: naredbe i podaci. Naredbe određuju šta se radi, a podaci čime se radi. Organizacija programa može da bude orijentisana ka jednoj od ove dve grupe elemenata.

Klasičan stil programiranja okrenut je prema postupcima i naziva se **proceduralno programiranje** (*procedural programming*). Po tom modelu program se sastoji od niza uzastopnih koraka. Logičke celine koraka mogu da se ostvaruju u obliku modula koji se nazivaju potprogrami, procedure ili funkcije. Složena obrada ostvaruje se kombinovanjem takvih modula.

Mana ovakvog pristupa programiranju je veliki stepen povezanosti delova složenog sistema. To otežava održavanje i unapređivanje sistema. Da bi se unele nove mogućnosti često je potrebno preraditi vrlo veliki deo već gotovih delova programa.

Dva programska jezika koji podržavaju proceduralno programiranje i koji su u današnje vreme vrlo rašireni jesu *Pascal* i *C*. Programski jezik *C++* takođe podržava proceduralno programiranje, ali njegovo glavno polje primene jeste u objektno-orijentisanom programiranju.

Savremen stil programiranja okrenut je prema podacima i naziva se **objektno-orijentisano programiranje** (*OOP – object-oriented programming*). Po tom modelu program se sastoji od **objekata** (*objects*) koji imaju neka moguća stanja i ponašanja. Stanja predstavljaju vrednosti objekata, koje vremenom mogu da se menjaju. Ponašanja predstavljaju pravila menjanja stanja, reakcije na uticaje okoline i načine uticanja na okolinu. Celokupna obrada ostvaruje se u obliku međusobnih delovanja objekata u programu.

Svi objekti u programu grupišu se po svojim osobinama. Grupe objekata sa sličnim osobinama čine **klase** (*classes*). Objekti iste klase imaju ista moguća stanja i ista ponašanja. Objekti date klase predstavljaju pojedinačne primerke svojih klasa.

Pojam klase i objekta najlakše se shvata kroz primere iz svakodnevnog života. *Pas* (klasa) je životinja sa svima poznatim opštim osobinama i ponašanjem. *Lesi* (objekat) je jedan tačno određeni pas s konkretnim osobinama (boja, težina, starost) i ponašanjem koje se uklapa u ponašanje svih pasa. Drugi primer: *automobil* je opšte ime (klasa) za sva prevozna sredstva te vrste, dok je *automobil BG720AX* jedan tačno određeni automobil (objekat).

Klase su tipovi podataka prema definiciji tipova podataka u programiranju, jer određuju moguće vrednosti svojih primeraka i moguće radnje nad tim primercima. Po tome, objekti su podaci klasnih tipova. Za razliku od prostih tipova podataka, kao što su celi brojevi ili realni brojevi, klase su složeni tipovi podataka.

Stanja objekata predstavljaju se podacima unutar objekata koji nazivaju se **polja** (*fields*). Polja mogu da budu podaci prostih i složenih tipova. U proste tipove podataka spadaju celi brojevi, realni brojevi itd., a u složene tipove nizovi i klase.

Ponašanje objekata ostvaruje se postupcima unutar klasa koji se nazivaju **metode** (*methods*). Metode odgovaraju funkcijama i procedurama u klasičnim programskim jezicima.

Polja i metode klasa zajedničkim imenom nazivaju se **članovi** (*members*) klasa.

Prednosti objektno-orijentisanog programiranja ogledaju se u tome što je pri menjanju mogućnosti programskog sistema potrebno prerađivati samo mali deo već gotovog programa. Ako se prošire mogućnosti neke klase nije potrebno promeniti i deo programa koji koristi tu klasu. Naravno, pod uslovom da se ništa ne promeni u načinu korišćenja mogućnosti klase koje su postojale i pre promene. Takođe, ako se u programski sistem uvode nove klase, deo programa koji koristi samo stare klase ne treba promeniti.

Najviše zastupljeni jezici za objektno-orijentisano programiranje su *C++*, *Java* i, u novije vreme, *C#*.

Objektno-orijentisano programiranje zasniva se na pet osnovnih principa: apstrakciji, učaurivanju, nasleđivanju, polimorfizmu i ponovnom korišćenju koda.

1.2.1 Apstrakcija

Pod apstrakcijom (*abstraction*) podrazumeva se zanemarivanje nebitnih detalja složenih objekata, u zavisnosti od trenutnih potreba.

Na primer, da bi se koristio televizor nije potrebno poznavati od čega se sastoji i kako radi televizor. Za korisnika je dovoljno da zna za postojanje same kutije televizora i daljinskog upravljača, da zna da priključi izvor napajanja i antenu, da stavi bateriju u daljinski upravljač i da rukuje televizorom pomoću dugmadi na samom aparatu i na daljinskom upravljaču. Majstoru, pak, koji treba da popravlja televizor, potrebno je da poznaje i komponente koje sadrži televizor.

Drugi primer: da bi se čovek učlanio u biblioteku, od svih njegovih osobina, potrebni su samo lični podaci (ime, adresa, broj lične karte itd.), a pri korišćenju lifta, njegova težina.

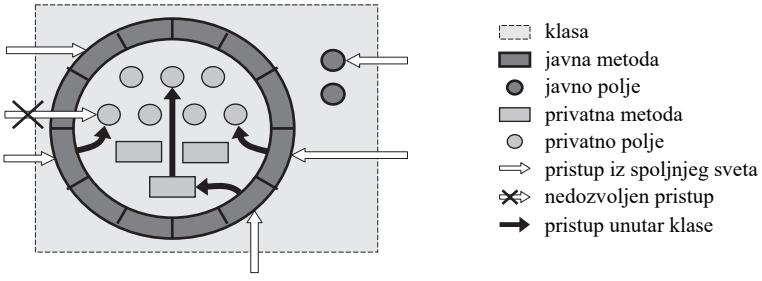
U programiranju apstrakcija se ogleda u odabiru parametara pri modelovanju predmeta ili pojmova koji su bitni za planiranu obradu.

1.2.2 Učaurivanje

Učaurivanje (*encapsulation*) u objektno-orijentisanom programiranju predstavlja mehanizam kojim se podaci i postupci u klasama štite od nepoželjnih uticaja okoline.

Na primer, televizor sadrži puno komponenata koje nisu pristupačne korisniku. Kutija uređaja čini zaštitnu površ. Vezu između korisnika i unutrašnjosti čine dugmad za upravljanje i ekran. Preko dugmadi korisnik utiče na rad televizora, a preko ekrana i zvučnika dobija informacije od televizora.

Učaurivanje u programiranju ostvaruje se podelom članova klase na javne i privatne (slika 1.1). **Javni članovi** (*public members*) mogu slobodno da se koriste u bilo kom delu programa, dok su **privatni članovi** (*private members*) nedostupni izvan klase. Polja su najčešće privatna dok su metode većinom javne. Do privatnih polja moguće je doći samo posredstvom javnih metoda, koje mogu da obezbede da sve promene stanja objekata (vrednosti polja) budu u skladu s definicijom klase. Za javna polja ne može da se sprovede kontrola ispravnog korišćenja, pa njih treba maksimalno izbegavati.



Slika 1.1 *– Javni i privatni članovi klase*

Privatne metode, mada su u manjini u odnosu na javne, mogu da budu korisne. One mogu da obezbede pomoćne radnje za potrebe javnih metoda.

Kao primer može se uočiti klasa kalendarskih datuma. Za polja su potrebna tri cela broja koji predstavljaju dan, mesec i godinu. Ako su polja javna ne može da se obezbedi da se u objektu ne pojavi trojka celih brojeva koji ne predstavljaju ispravan datum (na primer, 31.02.2015., pri čemu je svaka komponenta za sebe unutar dozvoljenih granica za datume –