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 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 dve 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. U ovom poglavlju obrađeni su i interfejsi, 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.

U poglavlju 9 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 10.

U poglavlju 11 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 12.

U poglavlju 13 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. Ipak, zbog široke upotrebe prozora u današnjim programima, u poglavlju 14 prikazani su osnovni elementi izrade programa korišćenjem grafičke korisničke površi.

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, *avgust 2015.*

Sadržaj

Predgovor ..... **v**

Sadržaj ..... **ix**

1 Uvod **1**

1.1 O programskom jeziku *Java* 1
1.2 Uvod u objektno-orijentisano programiranje 2
1.2.1 Apstrakcija 4
1.2.2 Učaurivanje 4
1.2.3 Nasleđivanje 5
1.2.4 Polimorfizam 6
1.2.5 Ponovno korišćenje koda 6
1.3 Osobine jezika *Java* 6
1.4 Obrada programa na jeziku *Java* 8
1.4.1 Obrada programa u tekstualnom režimu 8
1.4.2 Obrada programa u grafičkom režimu 10

2 Podaci **13**

2.1 Elementi jezika *Java* 13
2.2 Identifikatori 14
2.3 Tipovi podataka 15
2.4 Brojčani tipovi 17
2.4.1 Celobrojni tipovi podataka 17
2.4.2 Realni tipovi podataka 18
2.5 Znakovni podaci 18
2.6 Logički podaci 20
2.7 Niske 20
2.8 Definisanje podataka 20
2.9 Čitanje i pisanje podataka 22
2.9.1 Čitanje podataka 22
2.9.2 Pisanje podataka 23
2.10 Primeri potpunih programa 27

3 Operatori 33
3.1 Aritmetički operatori 34
3.2 Operatori za dodelu vrednosti 36
3.2.1 Inicijalizacija i dodela vrednosti 37
3.3 Pisanje složenih izraza 37
3.4 Matematičke funkcije 38
3.5 Konverzija tipa 39
3.6 Relacioni operatori 41
3.7 Logički operatori 41
3.8 Uslovni operator 42
3.9 Operatori po bitovima 43
3.10 Operatori za niske 44
3.11 Pregled operatora 45
3.12 Zadaci 46
3.12.1 Površina trougla u ravni 46
3.12.2 Pakovanje i raspakivanje vremena 48

4 Naredbe 51
4.1 Prosta naredba 51
4.2 Sekvenca ili blok: `{}` 52
4.3 Doseg identifikatora 53
4.4 Selekcije 55
4.4.1 Osnovna selekcija: `if-else` 55
4.4.2 Generalizovana selekcija: `if-else-if-else` 57
4.4.3 Selekcija pomoću skretnice: `switch` 58
4.5 Ciklusi 59
4.5.1 Osnovni ciklus s izlazom na vrhu: `while` 59
4.5.2 Generalizovani ciklus s izlazom na vrhu: `for` 60
4.5.3 Ciklus s izlazom na dnu: `do` 62
4.6 Skokovi 63
4.6.1 Označene naredbe 63
4.6.2 Iskakanje iz označene naredbe: `break` 63
4.6.2.1 Selekcija alternativnih grana 64
4.6.2.2 Ciklus s izlazom u sredini 66
4.6.3 Skok na kraj ciklusa: `continue` 66
4.7 Zadaci 67
4.7.1 Kvadratna jednačina 67
4.7.2 Statistički pokazatelji 70
4.7.3 Ispisivanje brojeva u binarnom sistemu 72

5 Nizovi 75
5.1 Definisanje nizova 75
5.2 Inicijalizacija nizova 77
5.3 Pristupanje elementima niza: `[]` 78
5.4 Dohvatanje dužine niza: `length` 78
5.5 Dodela vrednosti nizovnih promenljivih: `=` 79

5.6 Upoređivanje nizovnih promenljivih: `==`, `!=` ... 80
5.7 Nepravilni višedimenzionalni nizovi ... 81
5.8 Obilazak elemenata niza: `for` ... 82
5.9 Sakupljač otpadaka ... 83
5.10 Zadaci ... 84
5.10.1 Skalarni proizvod dva vektora ... 84
5.10.2 Uređivanje nizova brojeva ... 86
5.10.3 Unija skupova ... 90
5.10.4 Transponovanje matrice ... 93
5.10.5 Stepenovanje simetrične matrice ... 95

6 Klase ... 99

6.1 Definisanje klasa ... 100
6.2 Pristupanje članovima klasa ... 100
6.3 Polja klasa ... 101
6.4 Metode klasa ... 102
6.4.1 Definisanje metoda ... 103
6.4.1.1 Povratna vrednost metode ... 103
6.4.1.2 Parametri metode ... 103
6.4.1.3 Telo metode ... 103
6.4.1.4 Skriveni parametar metode ... 104
6.4.1.5 Bočni efekti metoda ... 105
6.4.2 Pozivanje metode ... 106
6.4.3 Rekurzivne metode ... 107
6.4.4 Metode s promenljivim brojem argumenata ... 109
6.4.5 Preopterećivanje imena metoda ... 110
6.5 Konstruktori ... 111
6.6 Objekti ... 113
6.6.1 Definisanje objekata ... 113
6.6.2 Uništavanje objekata ... 115
6.6.3 Tekstualni opis objekata ... 115
6.7 Statički članovi klasa ... 116
6.8 Inicijalizacioni blokovi ... 119
6.8.1 Statički inicijalizacioni blokovi ... 119
6.8.2 Inicijalizacija klasa ... 119
6.8.3 Nestatički inicijalizacioni blokovi ... 120
6.8.4 Stvaranje objekata ... 120
6.9 Glavna metoda ... 122
6.10 Anotacije ... 123
6.11 Dinamičke strukture podataka ... 124
6.12 Dijagrami klasa ... 126
6.13 Zadaci ... 129
6.13.1 Tačake u ravni ... 129
6.13.2 Nizovi tačaka u ravni ... 132
6.13.3 Materijalne tačake u prostoru ... 136

7 Paketi ... 143

7.1 Definisanje paketa ... 143
7.2 Korišćenje članova paketa ... 145
7.3 Potpaketi ... 147
7.4 Uskladištavanje hijerarhije paketa ... 149
7.5 Uvoženje statičkih članova klasa ... 150
7.6 Dijagrami klasa ... 151
7.7 Paket `java.lang` ... 152
7.7.1 Uslužna klasa za matematičke funkcije ... 152
7.7.2 Omotačke klase za proste tipove ... 152
7.7.2.1 Brojčani tipovi ... 152
7.7.2.2 Znakovni tip ... 153
7.7.2.3 Logički tip ... 154
7.7.2.4 Automatsko pakovanje i raspakivanje ... 155
7.7.3 Klase za niske ... 156
7.7.3.1 Klasa za nepromenljive niske ... 156
7.7.3.2 Klasa za promenljive niske ... 158
7.7.4 Klasa za sistemske radnje ... 159
7.8 Paket `java.util` ... 160
7.8.1 Klasa `Random` ... 160
7.8.2 Klasa `Scanner` ... 162
7.9 Zadaci ... 164
7.9.1 Ravan s obojenim krugovima ... 164

8 Nasleđivanje ... 173

8.1 Potklase ... 174
8.1.1 Definisanje potklasa ... 174
8.1.2 Korišćenje članova potklasa ... 177
8.1.3 Sakrivanje članova natklase ... 178
8.1.4 Korišćenje članova izvan potklase ... 181
8.1.5 Konačne metode ... 182
8.1.6 Stvaranje objekata potklasa ... 183
8.2 Kompatibilnost tipova ... 185
8.3 Nizovi ... 187
8.4 Polimorfizam ... 188
8.5 Klasa `Object` ... 191
8.6 Apstraktne klase ... 192
8.7 Interfejsi ... 194
8.7.1 Definisanje interfejsa ... 194
8.7.2 Ostvarivanje interfejsa ... 196
8.7.3 Višestruko nasleđivanje interfejsa ... 198
8.7.4 Podinterfejsi ... 199
8.7.5 Prazni interfejsi ... 201
8.7.6 Funkcijski interfejsi ... 201
8.7.7 Apstraktna klasa protiv interfejsa ... 202
8.8 *Primer složenijeg dijagrama klasa* ... 202

8.9 Zadaci 203
8.9.1 Geometrijske figure u ravni 203
8.9.2 Uređivanje uporedivih objekata 210

9 Ugnežđeni tipovi 219

9.1 Statički ugnežđeni tipovi 221
9.2 Unutrašnje klase 224
9.3 Lokalne klase 225
9.4 Bezimene klase 228
9.5 Lambda izrazi 230
9.6 Zadaci 237
9.6.1 Redovi objekat neograničenog kapaciteta 237
9.6.2 Nizovi celih brojeva 242

10 Izuzeci 245

10.1 Klase za izuzetke 246
10.2 Prijavljivanje izuzetaka 248
10.3 Rukovanje izuzecima 249
10.3.1 Naredba `try` 249
10.3.2 Izvršavanje naredbe `try` 250
10.3.3 Višetipni rukovaoci izuzecima 252
10.4 Kloniranje objekata 253
10.5 Zadaci 256
10.5.1 Vektori zadatog opsega indeksa 256
10.5.2 Izračunavanje određenog integrala 260
10.5.3 Predmeti koji mogu da se kloniraju 269

11 Generički tipovi i metode 277

11.1 Tipovne promenljive 277
11.2 Generičke klase i interfejsi 278
11.2.1 Definisanje generičkih klasa i interfejsa 278
11.2.2 Konkretizacija generičkih tipova 279
11.2.3 Uskladištavanje podataka u generičke objekte 281
11.2.4 Generički nizovi 282
11.2.5 Generičke potklase i ugnežđene klase 283
11.3 Generičke metode 284
11.3.1 Definisanje generičkih metoda 284
11.3.2 Pozivanje generičkih metoda 285
11.3.3 Zaključivanje tipovnih argumenata 286
11.3.4 Generičke metode u generičkim klasama 287
11.3.5 Generički konstruktori 288
11.4 Ograničavanje tipovnih parametara 289
11.5 Džoker tip 291
11.5.1 Upotreba džoker tipa 292
11.5.2 Ograničavanje džoker tipa 293
11.6 Brisanje tipova 295

11.7 Sirovi tipovi ... 298
11.8 Zbirke podataka ... 299
11.8.1 Obilazak elemenata zbirki naprednim `for` ciklusom ... 299
11.8.2 Radni okvir za zbirke ... 301
11.9 Zadaci ... 303
11.9.1 Generički redovi neograničenog kapaciteta ... 303
11.9.2 Napredna klasa redova objekata ... 305
11.9.3 Generički nizovi geometrijskih figura u ravni ... 308

12 Nabrajanja ... 315

12.1 Definisanje nabrajanja ... 315
12.2 Tip nabrajanja ... 316
12.3 Članovi nabrajanja ... 317
12.4 Bezimene klase u nabrajanjima ... 318
12.5 Zadaci ... 319
12.5.1 Špilovi karata za igru ... 319

13 Niti ... 325

13.1 Niti u jeziku *Java* ... 326
13.2 Stvaranje i izvršavanje niti ... 327
13.2.1 Stvaranje objekta niti ... 327
13.2.2 Vrste niti ... 329
13.2.3 Ime i identifikator niti ... 329
13.2.4 Prioritet niti ... 330
13.2.5 Dohvatanje grupe niti ... 330
13.2.6 Tekstualni opis niti ... 330
13.2.7 Pokretanje niti ... 330
13.2.8 Dohvatanje trenutne niti ... 331
13.2.9 Zahtev za prekidanje niti ... 331
13.2.10 Zaustavljanje niti na određeno vreme ... 332
13.2.11 Ustupanje procesora ... 333
13.3 Sinhronizacija niti ... 333
13.3.1 Čekanje da se nit završi ... 333
13.3.2 Sinhronizovani blokovi i metode ... 334
13.3.3 Modifikator `volatile` ... 337
13.3.4 Čekanje na signal ... 337
13.3.5 Slanje signala ... 338
13.4 Grupe niti ... 341
13.5 Zadaci ... 342
13.5.1 Zbirka s konkurentnim izračunavanjem ukupne veličine ... 342
13.5.2 Problem proizvođači/potrošači ... 346

14 Grafička korisnička površ ... 357

14.1 Klase opšte namene ... 358
14.1.1 Položaj i dimenzije – `Point`, `Dimension` i `Rectangle` ... 358
14.1.2 Boja – `Color` ... 359
14.1.3 Vrsta slova – `Font` ... 360

14.2 Komponente – `Component` ... 360
14.3 Događaji ... 363
14.3.1 Klase stanja događaja ... 365
14.3.2 Opšti događaji – grupa `Component` ... 366
14.3.3 Događaji žiže – gupa `Focus` ... 367
14.3.4 Ulazni događaji ... 367
14.3.5 Događaji miša – grupa `Mouse` ... 368
14.3.6 Događaji tastature – grupa `Key` ... 368
14.4 Kontejneri – `Container` ... 369
14.4.1 Događaji kontejnera – grupa `Container` ... 371
14.5 Raspoređivači ... 372
14.5.1 Linijski raspoređivač – `FlowLayout` ... 373
14.5.2 Mrežni raspoređivač – `GridLayout` ... 374
14.5.3 Ivični raspoređivač – `BorderLayout` ... 374
14.5.4 Kartični raspoređivač – `CardLayout` ... 375
14.6 Prozori ... 376
14.6.1 Događaji prozora – grupa `Window` ... 376
14.6.2 Osnovni prozor – `Window` ... 377
14.6.3 Prozor (s okvirom) – `Frame` ... 379
14.6.4 Prozor za dijalog – `Dialog` ... 382
14.7 Ploča – `Panel` ... 386
14.8 Natpis – `Label` ... 386
14.9 Dugme ... 387
14.9.1 Događaji akcije – grupa `Action` ... 387
14.9.2 Klasa `Button` ... 388
14.10 Tekstovi ... 388
14.10.1 Događaji tekstova – grupa `Text` ... 389
14.10.2 Komponenta za tekst – `TextComponent` ... 389
14.10.3 Polje za tekst – `TextField` ... 390
14.10.4 Prostor za tekst – `TekstArea` ... 391
14.11 Komponente sa stavkama ... 392
14.11.1 Događaji stavki – grupa `Item` ... 392
14.11.2 Polje za potvrdu – `Checkbox` ... 393
14.11.3 Grupa polja za potvrdu – `CheckboxGroup` ... 394
14.11.4 Lista – `List` ... 394
14.11.5 Izbor – `Choice` ... 396
14.12 *Primer korišćenja komponenata* ... 397
14.13 Meniji ... 399
14.13.1 Taster prečica – `MenuShortCut` ... 400
14.13.2 Komponenta menija – `MenuComponent` ... 400
14.13.3 Stavka menija – `MenuItem` ... 401
14.13.3.1 Stavka menija s potvrdom – `CheckboxMenuItem` ... 401
14.13.4 Kontejner menija – `MenuContainer` ... 402
14.13.5 Meni – `Menu` ... 402

14.13.6 Traka menija – `MenuBar` 403
14.13.7 Primer programa s menijem 403
14.13.8 Pomoćni meni – `PopupMenu` 405
14.14 Crtanje 408
14.14.1 Grafički kontekst – `Graphics` 408
14.14.2 Crtanje komponenata 411
14.14.3 Platno – `Canvas` 414
14.14.4 Preslikavanje dela *x-y* ravni na komponentu 416
14.15 Klase grafičke korisničke površi 419
14.16 Zadaci 420
14.16.1 Jednostavan kalkulator za cele brojeve 420
14.16.2 Problem proizvođači/potrošači u grafičkom okruženju 429
14.16.3 Crtanje polinoma 435

Literatura 443

Indeks 445

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.

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 dinamični 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 zove *Java* 2, a punim imenom *Java J2SE* (*Java 2 Platform Standard Edition*). Sledile su verzije 1.3 i 1.4 koje su donele manje izmene. Verzija 1.5 donela je ogromne promene, pa da bi se to naglasilo, ta verzija je proglašena verzijom 5 (*Java SE 5*). Aktuelna verzija u vreme pisanja ove knjige je 8.

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 na-

zivaju 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*.

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 BG0720AX* 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.

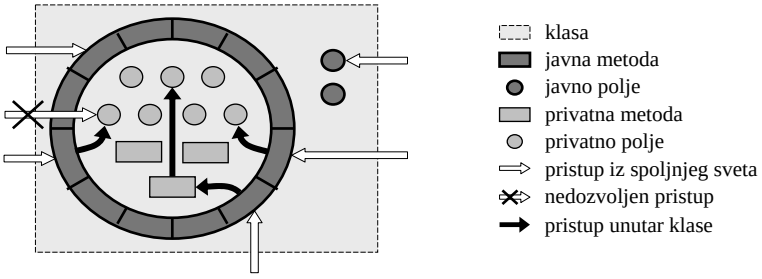
U programiranju apstrakcija se ogleda u odabiru parametara pri modelovanju predmeta ili pojmova koji su bitni za planiranu obradu.

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 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 (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 –