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

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
2.4.3 Grupisanje cifara 19
2.5 Znakovni podaci 19
2.6 Logički podaci 20
2.7 Niske 20
2.8 Definisanje podataka 21
2.9 Čitanje i pisanje podataka 23
2.9.1 Čitanje podataka 23
2.9.2 Pisanje podataka 24
2.10 Primeri potpunih programa 28

3 Operatori 35
3.1 Aritmetički operatori 36
3.2 Operatori za dodelu vrednosti 38
3.2.1 Inicijalizacija i dodela vrednosti 39
3.3 Pisanje složenih izraza 39
3.4 Matematičke funkcije 40
3.5 Konverzija tipa 41
3.6 Relacioni operatori 43
3.7 Logički operatori 43
3.8 Uslovni operator 44
3.9 Operatori nad bitovima 45
3.10 Operatori za niske 46
3.11 Pregled operatora 47
3.12 Zadaci 48
3.12.1 Površina trougla u ravni 48
3.12.2 Pakovanje i raspakivanje vremena 50

4 Naredbe 53
4.1 Prosta naredba 53
4.2 Sekvenca ili blok: `{}` 54
4.3 Doseg identifikatora 55
4.4 Selekcije 57
4.4.1 Osnovna selekcija: `if-else` 57
4.4.2 Generalizovana selekcija: `if-else-if-else` 59
4.4.3 Selekcija pomoću skretnice: `switch` 60
4.5 Ciklusi 61
4.5.1 Osnovni ciklus s izlazom na vrhu: `while` 61
4.5.2 Generalizovani ciklus s izlazom na vrhu: `for` 62
4.5.3 Ciklus s izlazom na dnu: `do` 64
4.6 Skokovi 65
4.6.1 Označene naredbe 65
4.6.2 Iskakanje iz označene naredbe: `break` 65
4.6.2.1 Selekcija alternativnih grana 66
4.6.2.2 Ciklus s izlazom u sredini 68
4.6.3 Skok na kraj ciklusa: `continue` 68
4.6.4 Izmene u selekciji pomoću skretnice <<*preliminarno*>> Δ 69
4.6.4.1 Više konstanti u `case` oznakama Δ 69
4.6.4.2 Sprečavanje prelivanja u naredni odeljak Δ 70
4.6.4.3 `switch` izraz Δ 70
4.7 Zadaci 71
4.7.1 Kvadratna jednačina 71
4.7.2 Statistički pokazatelji 73
4.7.3 Ispisivanje brojeva u binarnom sistemu 75

5 Nizovi ... 79

5.1 Definisanje nizova ... 79
5.2 Inicijalizacija nizova ... 81
5.3 Pristupanje elementima niza: `[]` ... 82
5.4 Dohvatanje dužine niza: `length` ... 82
5.5 Dodela vrednosti nizovnih promenljivih: = ... 83
5.6 Upoređivanje nizovnih promenljivih: ==, `!=` ... 84
5.7 Nepravilni višedimenzionalni nizovi ... 85
5.8 Obilazak elemenata niza: `for` ... 86
5.9 Sakupljač otpadaka ... 87
5.10 Zadaci ... 88
5.10.1 Skalarni proizvod dva vektora ... 88
5.10.2 Uređivanje nizova brojeva ... 90
5.10.3 Unija skupova ... 94
5.10.4 Transponovanje matrice ... 97
5.10.5 Stepenovanje simetrične matrice △ ... 99

6 Klase ... 103

6.1 Definisanje klasa ... 104
6.2 Pristupanje članovima klasa ... 104
6.3 Polja klasa ... 105
6.4 Metode klasa ... 106
6.4.1 Definisanje metoda ... 107
6.4.1.1 Povratna vrednost metode ... 107
6.4.1.2 Parametri metode ... 107
6.4.1.3 Skriveni parametar metode ... 108
6.4.1.4 Telo metode ... 108
6.4.2 Pozivanje metode ... 110
6.4.3 Prenošenje argumenata u metode ... 111
6.4.4 Rekurzivne metode ... 113
6.4.5 Metode s promenljivim brojem argumenata △ ... 114
6.4.6 Preopterećivanje imena metoda ... 115
6.5 Konstruktori ... 116
6.6 Objekti ... 118
6.6.1 Definisanje objekata ... 118
6.6.2 Uništavanje objekata ... 120
6.6.3 Tekstualni opis objekata ... 120
6.7 Dijagrami klasa ... 121
6.8 Statički članovi klasa ... 124
6.9 Inicijalizatorski blokovi ... 127
6.9.1 Statički inicijalizatorski blokovi ... 127
6.9.2 Inicijalizacija klasa ... 127
6.9.3 Nestatički inicijalizatorski blokovi ... 128
6.9.4 Stvaranje objekata ... 128
6.10 Glavna metoda ... 130
6.11 Zabeleške ... 131

6.12 Dinamičke strukture podataka ... 132
6.13 Zadaci ... 134
6.13.1 Tačke u ravni ... 134
6.13.2 Nizovi tačaka u ravni ... 138
6.13.3 Materijalne tačake u prostoru ... 141

7 Paketi ... 149

7.1 Definisanje paketa ... 149
7.2 Korišćenje članova paketa ... 151
7.3 Potpaketi ... 153
7.4 Uskladištavanje hijerarhije paketa ... 155
7.5 Uvoženje statičkih članova klasa ... 156
7.6 Dijagrami klasa ... 157
7.7 Paket `java.lang` ... 158
7.7.1 Uslužna klasa za matematičke funkcije – klasa `Math` ... 158
7.7.2 Omotačke klase za proste tipove ... 158
7.7.2.1 Brojčani tipovi – klase `Byte`, `Short`, `Integer`, `Long`, `Float` i `Double` ... 158
7.7.2.2 Znakovni tip – klasa `Character` ... 160
7.7.2.3 Logički tip – klasa `Boolean` ... 161
7.7.2.4 Automatsko pakovanje i raspakivanje ... 161
7.7.3 Klase za niske ... 162
7.7.3.1 Nepromenljive niske – klasa `String` ... 162
7.7.3.2 Promenljive niske – klasa `StringBuilder` ... 164
7.7.4 Klasa za sistemske radnje – klasa `System` ... 166
7.8 Paket `java.util` ... 167
7.8.1 Generisanje pseudoslučajnih brojeva – klasa `Random` ... 167
7.8.2 Razlaganje teksta na leksičke simbole – klasa `Scanner` ... 169
7.9 Zadaci ... 171
7.9.1 Ravan s obojenim krugovima ... 171

8 Nasleđivanje ... 179

8.1 Potklase ... 180
8.1.1 Definisanje potklasa ... 180
8.1.2 Korišćenje članova potklasa ... 183
8.1.3 Sakrivanje članova natklasa ... 185
8.1.4 Konačne metode ... 188
8.1.5 Stvaranje objekata potklasa ... 189
8.2 Kompatibilnost tipova ... 190
8.3 Nizovi ... 193
8.4 Polimorfizam ... 194
8.5 Klasa `Object` ... 197
8.6 Apstraktne klase ... 198
8.7 Zadaci ... 200
8.7.1 Geometrijske figure u ravni ... 200

9 Interfejsi 209
9.1 Definisanje interfejsa 209
9.2 Ostvarivanje interfejsa 211
9.3 Višestruko nasleđivanje interfejsa 213
9.4 Podinterfejsi 215
9.5 Prazni interfejsi 216
9.6 Funkcijski interfejsi 217
9.7 Apstraktna klasa protiv interfejsa 217
9.8 *Primer složenijeg dijagrama klasa* 218
9.9 Zadaci 219
9.9.1 Uređivanje uporedivih objekata 219

10 Ugnežđeni tipovi 227
10.1 Statički ugnežđeni tipovi 229
10.2 Unutrašnje klase 232
10.3 Lokalne klase 233
10.4 Bezimene klase 236
10.5 Lambda izrazi 238
10.5.1 Definisanje lambda izraza 238
10.5.2 Izračunavanje lambda izraza 239
10.5.3 Ostvarivanje lambda izraza 240
10.5.4 Lambda izrazi kao argumenti metoda 242
10.5.5 Upućivači na metode △ 243
10.5.6 Upućivači na konstruktore △ 245
10.6 Zadaci 246
10.6.1 Redovi objekata neograničenih kapaciteta 246
10.6.2 Nizovi celih brojeva 251

11 Izuzeci 255
11.1 Klase za izuzetke 256
11.2 Prijavljivanje izuzetaka 258
11.3 Rukovanje izuzecima 259
11.3.1 Naredba `try` 259
11.3.2 Izvršavanje naredbe `try` 260
11.3.3 Višetipni rukovaoci izuzecima 262
11.4 Kloniranje objekata 263
11.5 Zadaci 267
11.5.1 Vektori zadatog opsega indeksa 267
11.5.2 Izračunavanje određenog integrala 271
11.5.3 Predmeti koji mogu da se kloniraju 280

12 Generičke stvari 289
12.1 Tipovne promenljive 289
12.2 Generičke klase i interfejsi 290
12.2.1 Definisanje generičkih klasa i interfejsa 290
12.2.2 Konkretizacija generičkih klasa i interfejsa 291

12.2.3 Uskladištavanje podataka u generičke objekte ... 293
12.2.4 Generički nizovi ... 295
12.2.5 Generičke potklase i generičke ugnežđene klase ... 295
12.3 Generičke metode ... 296
12.3.1 Definisanje generičkih metoda ... 296
12.3.2 Pozivanje generičkih metoda ... 297
12.3.3 Zaključivanje tipovnih argumenata ... 298
12.3.4 Generičke metode u generičkim klasama ... 299
12.3.5 Generički konstruktori ... 301
12.4 Ograničavanje tipovnih parametara ... 301
12.5 Džoker tip ... 304
12.5.1 Upotreba džoker tipa ... 304
12.5.2 Ograničavanje džoker tipa ... 305
12.6 Brisanje tipova ... 308
12.7 Sirovi tipovi ... 310
12.8 Zbirke podataka ... 311
12.8.1 Obilazak elemenata zbirki unapređenim `for` ciklusom ... 311
12.8.2 Radni okvir za zbirke – klase `ArrayList<>` i `LinkedList<>` ... 313
12.9 Zadaci ... 315
12.9.1 Generički redovi neograničenih kapaciteta ... 315
12.9.2 Unapređena klasa redova objekata ... 318
12.9.3 Generički nizovi geometrijskih figura u ravni ... 320

13 Nabrajanja ... 327
13.1 Definisanje nabrajanja ... 327
13.2 Tip nabrajanja △ ... 328
13.3 Članovi nabrajanja △ ... 329
13.4 Dijagrami klasa ... 330
13.5 Bezimene klase u nabrajanjima △ ... 330
13.6 Zadaci △ ... 331
13.6.1 Špilovi karata za igru △ ... 331

14 Niti ... 337
14.1 Niti u jeziku *Java* ... 338
14.2 Stvaranje i izvršavanje niti ... 339
14.2.1 Stvaranje objekta niti ... 339
14.2.2 Vrste niti ... 341
14.2.3 Ime i identifikator niti ... 341
14.2.4 Prioritet niti ... 342
14.2.5 Dohvatanje grupe niti ... 342
14.2.6 Tekstualni opis niti ... 342
14.2.7 Pokretanje niti ... 342
14.2.8 Dohvatanje trenutne niti ... 343
14.2.9 Zahtev za prekidanje niti ... 343
14.2.10 Zaustavljanje niti na određeno vreme ... 344
14.2.11 Ustupanje procesora ... 345

14.3 Sinhronizacija niti 345
14.3.1 Čekanje da se nit završi 345
14.3.2 Sinhronizovani blokovi i metode 346
14.3.3 Modifikator `volatile` Δ 349
14.3.4 Čekanje na signal 349
14.3.5 Slanje signala 350
14.4 Grupe niti Δ 353
14.5 Zadaci 354
14.5.1 Zbirka s konkurentnim izračunavanjem ukupne veličine 354
14.5.2 Problem proizvođači/potrošači 358

15 Grafička korisnička površ 369

15.1 Klase opšte namene 370
15.1.1 Položaj i dimenzije – klase `Point`, `Dimension` i `Rectangle` 370
15.1.2 Boja – klasa `Color` 371
15.1.3 Vrsta slova – klasa `Font` 372
15.2 Komponente – klasa `Component` 372
15.3 Događaji 375
15.3.1 Klase stanja događaja 377
15.3.2 Opšti događaji – grupa `Component` 378
15.3.3 Događaji žiže – gupa `Focus` 379
15.3.4 Ulazni događaji 379
15.3.5 Događaji miša – grupa `Mouse` 380
15.3.6 Događaji tastature – grupa `Key` 380
15.4 Kontejneri – klasa `Container` 381
15.4.1 Događaji kontejnera – grupa `Container` 383
15.5 Raspoređivači 384
15.5.1 Linijski raspoređivač – klasa `FlowLayout` 385
15.5.2 Mrežni raspoređivač – klasa `GridLayout` 386
15.5.3 Ivični raspoređivač – klasa `BorderLayout` 386
15.5.4 Kartični raspoređivač – klasa `CardLayout` 387
15.6 Prozori 388
15.6.1 Događaji prozora – grupa `Window` 388
15.6.2 Osnovni prozor – klasa `Window` 389
15.6.3 Prozor (s okvirom) – klasa `Frame` 391
15.6.4 Prozor za dijalog – klasa `Dialog` 394
15.7 Ploča – klasa `Panel` 398
15.8 Natpis – klasa `Label` 398
15.9 Dugme 399
15.9.1 Događaji akcije – grupa `Action` 399
15.9.2 Klasa `Button` 400
15.10 Tekstovi 400
15.10.1 Događaji tekstova – grupa `Text` 401
15.10.2 Komponenta za tekst – klasa `TextComponent` 401
15.10.3 Polje za tekst – klasa `TextField` 402
15.10.4 Prostor za tekst – klasa `TekstArea` 403

15.11 Komponente sa stavkama....404
15.11.1 Događaji stavki – grupa `Item`....404
15.11.2 Polje za potvrdu – klasa `Checkbox`....405
15.11.3 Grupa polja za potvrdu – klasa `CheckboxGroup`....406
15.11.4 Lista – klasa `List`....406
15.11.5 Izbor – klasa `Choice`....408
15.12 *Primer korišćenja komponenata*....409
15.13 Meniji....411
15.13.1 Taster prečica – klasa `MenuShortCut`....412
15.13.2 Komponenta menija – klasa `MenuComponent`....412
15.13.3 Stavka menija – klasa `MenuItem`....413
15.13.3.1 Stavka menija s potvrdom – klasa `CheckboxMenuItem`....413
15.13.4 Kontejner menija – klasa `MenuContainer`....414
15.13.5 Meni – `Menu`....414
15.13.6 Traka menija – klasa `MenuBar`....415
15.13.7 *Primer programa s menije*m....415
15.13.8 Pomoćni meni – klasa `PopupMenu`....417
15.14 Crtanje....420
15.14.1 Grafički kontekst – klasa `Graphics`....420
15.14.2 Crtanje komponenata....423
15.14.3 Platno – klasa `Canvas`....426
15.14.4 Preslikavanje dela *x-y* ravni na komponentu....428
15.15 Klase grafičke korisničke površi....431
15.16 Zadaci....432
15.16.1 Jednostavan kalkulator za cele brojeve....432
15.16.2 Problem proizvođači/potrošači u grafičkom okruženju....441
15.16.3 Crtanje polinoma....447

Literatura....455

Indeks....457

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.

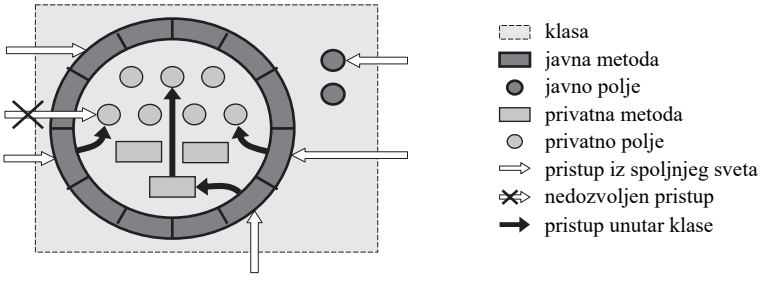
U programiranju apstrakcija se ogleda u odabiru parametara pri modelovanju predmeta ili pojmova koji su bitni za planiranu obradu.

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 –