Objektno orjentisano programiranje:

Nizovi i složeni tipovi podataka

[Prilagođeno od prof. dr Gordana Đorđević]

Nizovi

Nizovi su promjenljive koje mogu uskladištiti više vrijednosti odjednom

Deklaracija niza:

```
tip_podataka  [] nazivPromjenljive;
```

Inicijalizacija niza:

```
nazivPromjenljive = new tip_podataka [cijeli_broj];
```

Broj u uglastim zagradama je cijeli, nenegativan broj i predstavlja **kapacitet** niza.

Svaki element ima svoj indeks - cijeli broj koji predstavlja redni broj elementa u nizu. Indeksi elemenata počinju od 0.

Nizovi

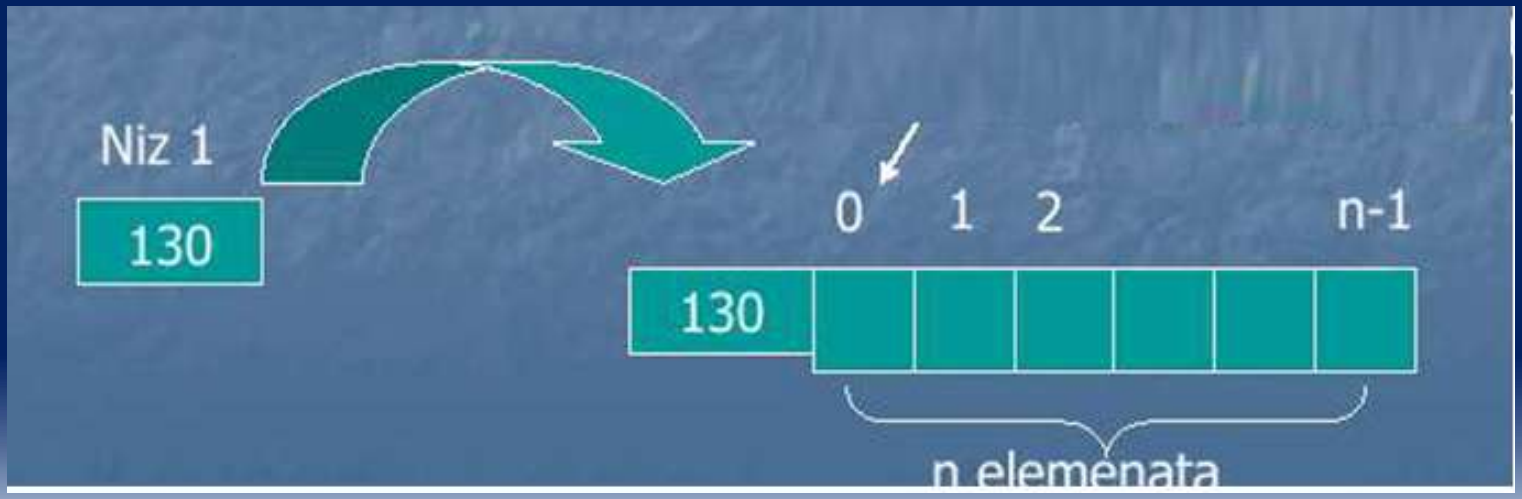
Dobijanje vrijednosti kapaciteta niza vrši se naredbom:

```
nazivPromjenljive.length
```

Pozivanje pojedinačnog elementa niza vrši se naredbom:

```
nazivPromjenljive [indeks]
```

Elementima niza pristupa se preko indeksa (u rasponu od 0 do n-1) – svaki element ima svoj indeks



Primer: niz

Napravi klasu MjesecniProfit koja ima:

- Atribut profiti, koji predstavlja niz od 12 realnih brojeva. Svaki element niza predstavlja profit za odreDeni mesec (januar, februar, ..., decembar)
- Metodu unesiProfit koja kao parametre prima realan broj koji predstavlja profit i ceo broj koji predstavlja redni broj meseca na koji se taj profit odnosi.
- Metodu ispisi koja će na ekranu ispisati profit za svaki mesec

Napraviti klasu TestMjesecniProfiti koja kreira jedan objekat klase MjesecniProfiti, unosi profit za februar koji iznosi 122.33 i ispisuje profit za svaki mesec

```
Class MjesecniProfiti {
   double [] profiti = new doubl [12];
   void unesiProfit (double profit, int mjesec) {
        profiti [mjesec-1] = profit;
   }
   void ispisi () {
        for (i=0; i<profit.lenght; i++)
             System.out.println (profiti [i]);
   }
}
Class TestMjesecniProfiti {
   public static void main (String [] args) {
        MesecniProfiti mp = new MesecniProfiti();
        mp.unesiProfit(122.33, 2);
        mp.ispisi();
   }
}
```

- Napravi klasu **OceneNaIspitnomRoku** koja ima:
  - Atribut ocene koji predstavlja ocene studenata na ispitnom roku. Zna se da ispit može da polaže maksimalno 100 studenata
  - Atribut brojac koji predstavlja trenutni broj elemenata u nizu. Početna vrednost mu je 0.
  - Metodu koja kao parametar dobija ocenu na ispitu i unosi je u niz i to na prvo slobodno mesto.
  - Metodu ispisi koja na ekranu ispisuje sve ocene na ispitnom roku
- Napraviti klasu **TestOceneNaIspitnomRoku** koja kreira jedan objekat klase **OceneNaIspitnomRoku**, unosi u njega ocene 5,10,10,7 i 8 i ispisuje sve ocene na ekranu.

```
Class OceneNaIspitnomRoku {
    int [] ocene = new int [100];
    int brojac = 0;

    void unesiOcenu (int ocene) {
          ocene [brojac] = ocene;
          brojac ++;
    }

    void ispisi () {
          for (int i=0; i<brojac; i++)
                System.out.println (ocene [i]);
    }
}

Class TestOceneNaIspitnomRoku {
    public static void main (String [] args) {
          OceneNaIspitnomRoku oir = new OceneNaIspitnomRoku();
          oir. unesiOcenu (5);
          oir. unesiOcenu (5);
          oir. unesiOcenu (10);
          oir. unesiOcenu (10);
          oir. unesiOcenu (7);
          oir. unesiOcenu (8);
          oir.ispisi();
    }
}
```

Višedimenzionalni nizovi (matrice i tabele sa podacima)

- Dekleracija višedimenzionalnih nizova

```
tip_podatka [] [] nazivPromenljive;
```

- Inicijalizacija niza po svakoj dimenziji:

```
nazivPromenljive = new tip_podatka [ceo_broj_1]
   [ceo_broj_2];
```

- Prvi broj u zagradi predstavlja broj redova matrice, a broj u drugoj zagradi broj kolona.
- Pozivanje pojedinačnih elemenata niza vrši se naredbom:

```
nazivPromenljive [indeks_1] [indeks_2]
```

Nizovi objekata

- U Javi je moguće napraviti niz čiji elementi nisu prostog tipa, već je svaki element po jedan objekat
- Rad sa nizovima objekata je potpuno isti kao i rad sa običnim nizovima – jeino se svaki pojedinačni element tretira malo drugačije jer je u pitanju objekat

Nizovi objekata

- Dekleracija niza objekta

```
tip_podatka [] nazivPromenljive;
```

- S obzirom da su elementi niza objekti neke klase, tip podatka je upravo naziv te klase

```
NazivKlase [] nazivPromenljive;
```

Inicijalizacija

- Inicijalizacija niza je potrebna i u ovom slučaju i vrši se na isti način kao i za običan niz

```
nazivPromenljive = new NazivKlase [ceo_broj];
```

- Kod inicijalizacije niza objekata, inicijalizacija niza samo rezerviše memorijski prostor za pokazivače na objekte, ali ne inicijalizuje same objekte
- Posle inicijalizacije niza objekata, svaki element ima vrednost **null** i sa njim se ne može raditi dok se ne inicijalizuje
- U nekim situacijama je potrebno izvršiti pojedinačnu inicijalizaciju elemenata odmah na početku, a nekad se samo unose već inicijalizovani objekti na odgovarajuća mesta u nizu

Pristup elementima niza i metode

- Pristup elementima niza se vrši preko indeksa, a maksimalni kapacitet niza se dobija komandom **lenght**

```
nazivPromenljive[indeks]
```

```
nazivPromenljive.length
```

- Za razliku od običnih nizova, elementi niza objekata mogu direktno pozivati metode

```
nazivPromenljive[indeks].nazivMetode(...argumenti...)
```

Liste

- Nizovi omogućavaju rad sa više promenljivih bez potrebe za pojedinačnim definisanjem svake promenljive;
- jednom inicijalizovan niz ima fiksan kapacitet i ne može se menjati.
- Liste (dinamički nizovi) nemaju ograničen kapacitet

Liste

- U Javi postoji nekoliko predefinisanih klasa koje su u stvari liste.
- LinkedList – elementi ove liste ne mogu biti prostog tipa, već samo objekti, i to objekti iste klase.

```
import java.util.LinkedList;
```

- Deklaracija liste:

```
LinkedList <NazivKlaseElemenata> nazivPromenljive
```

- Inicijalizacija liste:

```
nazivPromenljive = new LinkedList <NazivKlaseElemenata>
   ();
```

Klasa `String`

- Niz znakova može se definisati korišćenjem primitivnog tipa `char`, ali bi za neke operacije, poput izdvajanja dijelova znakova ili provjere prisustva nekih znakova u nizu, bilo neophodno pisanje metoda
- U javi postoji predefinisana klasa za rad sa nizovima znakova koja se naziva `String`
- Ova klasa ima implementirane brojne funkcionalnosti koje obezbjeđuju realizaciju već pomenutih operacija
- Deklaracija `String` promenljive se realizuje na isti način kao i za bilo koju drugu promenljivu (`String` je klasa pa njen naziv počinje velikim slovom)

```
String nazivPromenljive
```

- Obzirom da je `String` klasa, svaka promenljiva ovog tipa je objekat pa se mora inicijalizovati pre korišćenja

Inicijalizacija

- I način

```
String nazivPromenljive = new String (“neki niz znakova”);
```

- II način

```
String nazivPromenljive = “neki niz znakova”;
```

- III način

```
Char [] niz;


String s = new String (niz);
```

- Napomena: String vrijednosti se uvek pišu pod dvostrukim znacima navoda " ", a vrijednosti promenljivih koje su tipa char pod jednostrukim znacima navoda 'x'

Prazan String i nadovezivanje String vrednosti

- Prazan String – promenljiva String ne sadrži nijedan znak

```
nazivPromenljive = “”;
```

- Nekada je potrebno spojiti više vrednosti u jedan string; u pitanju je nadovezivanje String vrednosti i vrši se korišćenjem operatora za nadovezivanje “+” (konkatenacija)

```
nazivPromenljive = “neki niz znakova” +
“drugi niz znakova”;
```

- Karakteristika ovog operatora je da će sve vrednosti koje nisu String tipa biti pretvorene u odgovarajući niz znakova i nadovezane na postojeću String vrednost. Jedini preduslov je da je makar jedna vrednost u celom izrazu String.

Neki primeri nadovezivanja String vrednosti

- Primer 1

```
String    s1 = "Danas"
String    s2  = "je lijep  dan."
String    s3  =  s1 + s2
```

// string s3 će imati vrednost "Danas je lep dan."

- Primer 2

```
int temperatura = 15;
String s4 = "Napolju je "+temperatura+"
  stepeni";
```

// string s4 će imati vrednost "Napolju je 15 stepeni"

Neki primeri nadovezivanja String vrednosti

- Primer 3

```
String s5 = 12 + 15
// ovo je graška jer nijedan element nije String vrednost
//* ako hoćemo da kao rezultat imamo napisan broj 1215 onda to mo
   na  jedan od sledeća tri načina*//

String s5 = "12" + 15     //1215

String s5 = 12 + "15"     //1215

String s5 = "" + 12 + 15  //1215

String s5 = "" + (12 + 15) //27
```

Poređenje String vrednosti

- Znak "==" se koristi kod poreĎenja prostih tipova podataka (int, char, double) ali za poreĎenje String vrednosti potrebno je koristiti **equals** metodu

- Ova metoda proveravą jednakost dve string vrednosti i vraća **true** ili **false**; prilikom poreĎenja obraća se pažnja i na velika i mala slova

```
promenljiva1.equals(promenljiva2)
```

- Operator za poreĎenje jednakosti može se koristiti samo kada želimo proveriti da li string promenljiva ima "null" vrednost

```
promenljiva == null;
```

Metode charAt i lenght

- Metode charAt omogućiće pristupanje svakom pojedinačnom znaku u nizu znakova – Stringu.
- Svaki znak u Stringu ima svoj indeks koji počinje od 0 i tu je klasa String slična nizu. Ipak za pristup članovima Stringa ne koristi se uglasta zagrada kao kod nizova, već pomenuta metoda charAt

```
promenljiva.charAt(indeks)
```

- Metoda lenght se koristi za vraćanje dužine String vrednosti

```
promenljiva.lenght()
```

Primer: Metode charAt i lenght

- Napraviti klasu PrebrojavanjeZnakova koja ima:
  - Statičku metodu prebroj koja kao parametar prima String i vraća koliko se puta u njemu pojavljuje slovo „a“

```
Class PrebrojavanjeZnakova {
   static int prebroj (String s) {
        int brojac = 0;
        for(i=0; i<s.lenght(); i++);
              if (s.charAt(i)== 'a')
                    brojac++;
              return brojac;
        }
}
```

Zadatak:

- Napraviti klasu AnalizatorSlovaUTekstu koja ima:
  - Atribut tekst koji je tipa String koji ima početnu vrednost "nepoznat"
  - Metodu getTekst koja vraća trenutnu vrednost atributa tekst
  - Metodu koja vraća dužinu teksta (broj znakova u tekstu)
  - Metodu koja proverava i vraća broj pojavljivanja malog slova „i" u tekstu
  - Metodu koja proverava i vraća broj pojavljivanja nekog slova u tekstu; slovo koje se traži je dato u vidu parametra
  - Metodu koja proverava i vraća broj pojavljivanja rečenica u tekstu; svaka rečenica se obavezno završava tačkom, znakom pitanja ili uzvičnikom

```
Class AnalizatorSlovaUTekstu {
    String tekst = "nepoznat";
    String getTekst () {
          return tekst;
    }
    int duzinaTeksta () {
          return tekst.lenght ();
    }
    int brojPojavljivanjaSlovaI () {
          int brojac = 0;
          for(i=0; i<tekst.lenght(); i++);
                    if (tekst.charAt(i) == 'i') brojac++;
          return brojac;
    }
    int brojPojavljivanjaSlova (char c) {
          int brojac = 0;
          for(i=0; i<tekst.lenght(); i++);
                    if (tekst.charAt(i) == c) brojac++;
          return brojac;
    }
    int brojRecenica () {
          int broj_recenica = 0;
          for(i=0; i<tekst.lenght(); i++);
                    If ((tekst.charAt(i) == '.') || (tekst.charAt(i) == '?') ||
(tekst.charAt(i) == '!')) brojac_recenica++;
          return brojac_recenica;
    }
}
```

Klasa GregorianCalendar

- Postavljanje datuma i vremena u računaru je relativno složen problem jer datum I vreme sadrže veliku količinu podataka (dan, mese, godina, sat, ...), pa se ne mogu predstaviti korišćenjem nekog prostog tipa podatka.
- Postoji i više tipova kalendara prema načinu računanja vremena
- Operacije sa ovim podacima (manipulacija podacima, računanje vremena, ispisivanje datuma) su složene operacije pa je dobro koristiti neke gotove metode za ovakve operacije
- U Javi postoji predefinisana klasa GregorianCalendar kojom se predstavljaju datum I vreme i koja sadrži odgovarajuće metode za rad sa njima
- Svaki put kada se kreira objekat klase GregorianCalendar on dobija podatke o trenutnom datumu i vremenu iz računara

```
GregorianCalendar datum = new GregorianCalendar ();
```

- Pre nego što se uopšte može koristiti ova klasa potrebno je dodati odgovarajuću "import" naredbu. Razlog je što se klasa GregorianCalendar nalazi u posebnoj biblioteci java.util, pa je potrebno naglasiti da se koristi ta biblioteka

```
import java.util.GregorianCalendar;
```

- Pored osnovnih podataka o datumu i vremenu, klasa GregorianCalendar sadrži I dodatne podatke o tome koja je nedelja u godini, koji je dan u godini i sl. Zbog toga je važno da se prilikom ispisivanja vodi računa o tome koje podatke treba ispisati
- Najčešće se koristi metoda **getTime** koja objekat klase Date koji predstavlja klasičan način ispistivanja klase GregorianCalendar jer sadrži samo osnovne podatke o datumu i vremenu

Primer: Klasa GregorianCalendar

- Napraviti klasu Datumi koja ima:
  - Statičku metodu koja na ekranu ispisuje trenutni datum i vreme
- Napraviti klasu TestDatumi koja poziva metodu klase Datumi za ispisivanje trenutnog datuma i vremena na ekranu

```
import java.util.GregorianCalendar;

Class Datumi {
   static void ispisiTrenutniDatumIVreme () {
        GregorianCalendar kal = new GregorianCalendar ();
        System.out.println (kal.getTime());
   }
}

Class TestDatumi {
   public static void main (String [] args) {
   Datumi.ispisiTrenutniDatumIVreme ();
    }
}
```

`Set` metoda

- Objekti klase GregorianCalendar mogu da postavljaju i neki drugi datum I vreme – nečiji datum roĎenja, datum prodaje i dr.
- Ručno postavljanje datuma i vremena vrši se pozivanjem `set` metode
- Prva verzija ove metode prima tri parametra (godina, mesec, dan) I podešava samo datum, dok druga varijanta ima šest parametra (godina, mesec, dan, sat, minute, sekunde) i podešava i datum i vreme

```
promenljiva.set(p1, p2, p3);
```

- Potrebno je obratiti pažnju da meseci počinju od 0 a ne od 1 (0-januar, 1- februar, ..., 11-decembar)

```
promenljiva.set(p1, p2, p3, p4, p5, p6);
```

Primer: `Set` metoda

- Dopuni klasu Datumi tako da ima:
- Statičku metodu promeniDatum koja kao parametar prima objekat klase GregorianCalendar i postavlja da njegov datum bude 01.01.2012.
- Statičku metodu promeniDatumIVreme koja kao parametar prima objekat klase GregorianCalendar i postavlja da njegov datum bude 31.12.2011. 23:45:12

```
static void promeniDatum (GregorianCalendar datum ) {
   datum.set(2012, 0, 1);
}

static void promeniDatumIVreme (GregorianCalendardatum ) {
   datum.set(2011, 11, 31, 23, 45, 12);
}
```

Metoda `get`

- U nekim slučajevima je potrebno preuzeti samo neke elemente datuma (dakle ne sve)
- U tu svrhu se koristi `get` metoda; ova metoda kao parametar prima ceo broj i vraća neki od elemenata datuma
- U zavisnosti od toka koji ceo broj se unese, metoda će vratiti godinu, mesec, dan, sat, minut ili nešto drugo
- Da se ove celobrojne vrednosti ne bi pamtile, klasa GregorianCalendar ima i statičke konstante koje predstavljaju odgovarajuće brojeve

```
GregorianCalendar.YEAR
GregorianCalendar.MONTH
GregorianCalendar.DAY_OF_MONTH
GregorianCalendar.HOUR
GregorianCalendar.MINUTE
GregorianCalendar.SECOND
```

Metode `before` i `after`

- Često je potrebno uporediti dva datuma i utvrditi koji se od njih odnosi na raniji trenutak
- Klasa GregorianCalendar ima dve predefinisane metode koje se koriste u ovakvim situacijama `before` i `after`
- Metoda `before` poredi dva datuma i vraća **true** ako se prvi datum odnosi na raniji trenutak, a **fallse** ako to nije slučaj
- Metoda `after` radi potpuno suprotno

Primer

- Dopuni klasu Datumi tako da ima:
- Statičku metodu ispisiPosebno koja kao parametar dobija objekat klase GregorianCalendar i na ekranu ispisuje godinu, mesec, dan, sat, minut i sekundu u formatu "DD.MM.GGGG SS:MI:SE"
- Statičku metodu starijaOsoba koja kao parametre prima datume roĎenja dve osobe I na ekranu ispisuje koja osoba je starija

```
static void ispisiPosebno (GregorianCalendar datum ) {
    int godina = datum.get(GregorianCalendar.YEAR);
    int mesec = datum.get(GregorianCalendar.MONTH);
    int dan = datum.get(GregorianCalendar.DAY_OF_MONTH);
    int sat = datum.get(GregorianCalendar.HOUR);
    int minut = datum.get(GregorianCalendar.MINUTE);
    int sekunda = datum.get(GregorianCalendar.SECOND);
    System.out.println (dan+"."+mesec+"."+godina+".
    "+sat+":"+minut+":"+sekunda);
}

static void starijaOsoba (GregorianCalendar datum_rodjenja1,
    GregorianCalendardatum_rodjenja2) {
    if (datum_rodjenja1.before(datum_rodjenja2))
         System.out.println ("Starija je prva osoba");
    else
         System.out.println ("Starija je druga osoba");
}
```