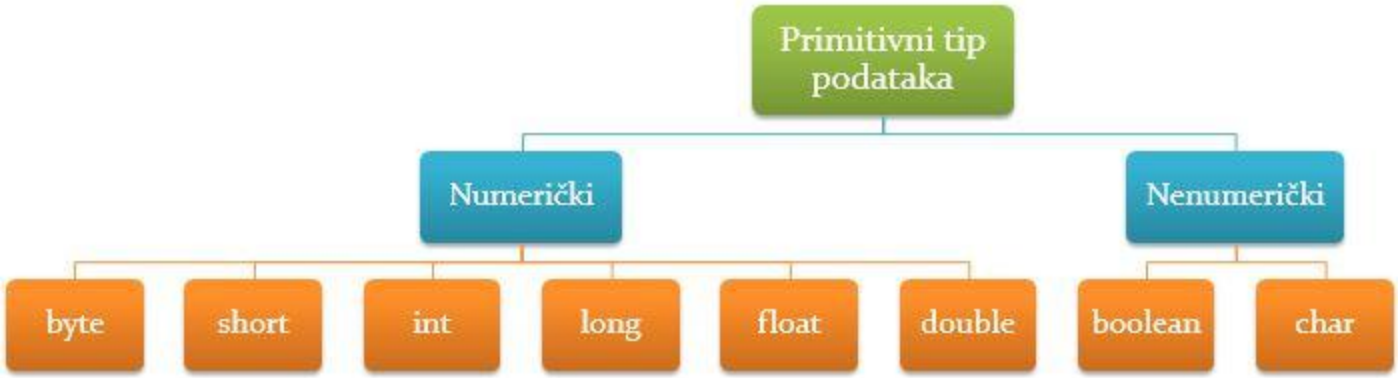
Objektno-orijentisano programiranje

Primitivni tipovi u Javi, nizovi i matrice

Nevena Ćirić
www.matf.bg.ac.rs/~nevena_ciric

Primitivni tipovi podataka u Javi



- Jedine 'stvari' koje nisu objekti u Javi su promenljive koje odgovaraju nekom od primitivnih tipova podataka.
- Primitivne tipove podataka definiše tip vrednosti koji mogu da čuvaju i veličina memorije u bajtovima za zapisivanje odgovarajućih vrednosti, ali ne i metodi za rad sa tim podacima (za razliku od klasnih tipova podataka).

Omotač klase

- Svaki od primitivnih tipova ima definisanu ekvivalentnu, tzv. omotač klasu:
  - byte – Byte
  - short – Short
  - int – Integer
  - long – Long
  - float – Float
  - double – Double
  - boolean – Boolean
  - char – Character
- Omotač klase sadrže razne statičke metode koje možemo koristiti u radu nad podacima primitivnog tipa.

Omotač klasa Character

- Klasa Character je i dostupna u okviru Java standardne biblioteke klasa. Nalazi se u java.lang paketu koji se importuje automatski.
- Neki od statičkih metoda iz klase Character:
  - isLetter()
  - isLetterOrDigit()
  - isWhiteSpace()
  - isLowerCase()
  - isUpperCase()
  - toLowerCase()
  - toUpperCase()
- Svi kao argument uzimaju jedan char

Konverzija primitivnih tipova u String

- Konverzija primitivnih tipova u tip String zapravo se vrši korišćenjem statičkog metoda toString() standardne klase koja odgovara tom primitivnom tipu (npr. Integer.toString(), Double.toString())
- Kadgod se vrednost nekog od primitivnih tipova pojavi kao operand za + (operator konatenacije), a drugi operand je String objekat, kompajler prosleđuje vrednost primitivnog tipa metodu toString() definisanom u odgovarajućoj omotač klasi. Metodi toString() vraćaju String ekvivalentan vrednosti koja im se prosledi kao argument. Sve se to dešava automatski pri konkatenaciji stringova operatorom +.

Klasa Scanner

- Nalazi se u java.util paketu koji se mora importovati (java.util.Scanner je puno kvalifikovano ime klase).
- Klasa Scanner definiše metode za čitanje i parsiranje primitivnih vrednosti (ne svih – karakteri ne) i stringova.
- Ulaz može biti string, fajl ili tok podataka, uključujući i standardni ulazni tok System.in
- Funkcije (konstruktori) kojima se kreiraju objekti klase Scanner:
  - public Scanner(InputStream source)
  - public Scanner(File source)
  - public Scanner(String source)

Klasa Scanner

- Da bismo kreirali objekat klase koristimo (unarni) operator new za kojom sledi poziv konstruktora. Operator new dinamički alocira memoriju za novi objekat, kreira i inicijalizuje objekat pozivanjem odgovarajućeg konstruktora klase i vraća referencu na tu memoriju.
- Prema prethodno rečenom, memorija za sve objekte u Javi se alocira dinamički.
- U Javi se svim objektima se pristupa preko referenci, tzv. ručki (eng. handle) za pristup objektu.
- Kreiranje objekta klase Scanner:

```
Scanner sc = new Scanner(System.in);
```

- Promenljiva sc ne predstavlja objekat klase Scanner, već čuva referencu na objekat tipa Scanner koji je vezan za ulazni tok System.in

Klasa Scanner

- Ulaz se deli na tokene koristeći kao podrazumevani delimiter beline (' ', '\n', '\t', ...).
- Metodi next*() i hasNext*() preskaču karaktere sa ulaza koji odgovaraju delimiterima i pokušavaju da vrate naredni token koji odgovara tipu. Ovi metodi postoje za većinu primitivnih tipova podataka (za karaktere ne) i stringove.
- Metodom close() zatvara se skener, kao i ulazni tok za koji je skener vezan.
- Kada je skener vezan za standardni ulaz ne treba ga zatvarati jer bi se u tom slučaju zatvorio i standardni ulaz tako da program ne može da ga koristi na dalje. Eventualno, možemo ga zatvoriti na samom kraju programa.

Nizovi u Javi

- Nizovi su specijalni tipovi objekata koji mogu da čuvaju linearno uređenu kolekciju elemenata.
- Java podržava nizove elemenata svih primitivnih i klasnih tipova. Za svaki nizovni tip dostupna je odgovarajuća klasa koja ga definiše.
- Kadgod definišemo promenljivu nizovnog tipa Java implicitno kreira klasu odgovarajućeg nizovnog tipa. Ove klase su deo programskog jezika Java i nisu dostupne neposredno na programerskom nivou. Možemo saznati samo naziv klase pozivom metoda getClass().getName() nad promenljivom odgovarajućeg nizovnog tipa.

Nizovi u Javi

Array type	Corresponding class Name
int[]	[I
int[][]	[[I
double[]	[D
double[][]	[[D
short[]	[S
byte[]	[B
boolean[]	[Z

- Iznad je prikazana tabela sa nazivima klasa koje odgovaraju nekim nizovnim tipovima.
- PRIMER: klasa [I
  - jedna [zato što klasa odgovara jednodimenzionalnom nizovnom tipu
  - I je zato što je bazni tip (tip elemenata niza) int

Nizovi u Javi

- Deklaracija nizovne promenljive: int[] a; ili int a[];
- Kreiranje objekta nizovnog tipa: a = new int[10];
- Nizovna promenljiva čuva referencu na objekat nizovnog tipa. Deklaracijom se ne alocira memorija za sam niz – to se vrši prilikom kreiranja objekta odgovarajućeg nizovnog tipa.
  Veličina niza se ne navodi u deklaraciji!
- Moguće je u jednoj naredbi deklarisati nizovnu promenljivu i definisati niz na koji će ona referisati:
  - int[] a = new int[10];
  - int a[] = new int[10];

Nizovi u Javi

- Prilikom kreiranja niza upotrebom operatora new elementi niza se automatski inicijalizuju na podrazumevanu vrednost, koja zavisi od tipa elemenata:
  - numeričke vrednosti – 0
  - boolean – false
  - char – '\u0000'
  - klasni tip – null
- Moguće je u jednoj naredbi izvršiti kreiranje niza i inicijalizaciju elemente niza:

  ```
  int[] a = {1, 2, 3, 4, 5};
  ```

  pri čemu će veličina niza biti određena kao broj vrednosti za inicijalizaciju
- Dužinu niza dobijamo sa a.length (atribut objekta a)

Nizovi nizova (matrice)

- Možemo da pravimo i nizove nizova jednakih ili različitih dužina:
  - int[][] a = new int[3][5];
  - int[][] a = new int[3][];
    a[0] = new int[4];
    a[1] = new int[5];
    a[2] = new int[6]
- U oba slučaja nizovna promenljiva a referiše na niz referenci na jednodimenzionalne nizove odgovarajućeg tipa.
- Dužine svakog od (pod)nizova čuva se u atributu lenght odgovarajućih objekata:
  a.length = 3
  a[2].length = 6

For each petlja

- Može se koristiti i oblik for petlje za kolekcije umesto klasične numeričke forme kada je potrebno pristupiti svim elementima niza.
- Foreach/kolekcijska petlja nas oslobađa indeksiranja i brige o dužini kolekcije (niza) kroz koji se iterira.

```
int suma = 0;
for(int elem : a)
        suma += elem;
```

- Ovakav oblik for petlje omogućuje samo iteraciju kroz sve elemente niza. Ne može se upotrebiti za pristup elementima niza kako bi se promenila njihova vrednost. U tom slučaju treba koristiti numeričku formu for petlje.

Klasa Arrays

- Nalazi se u java.util paketu koji se mora importovati (java.util.Arrays je puno kvalifikovano ime klase).
- Sadrži veliki broj statičkih metoda za rad sa nizovima. Metodi su generičkog tipa, tj. mogu da prihvate kao argument nizove bilo kog tipa.
- Neki od statičkih metoda iz klase Arrays:
  - toString()
  - deepToString()
  - sort()
  - binarySearch()
  - fill()
  - copyOf()
  - copyOfRange()