Siniša Nikolić

UOP - Predavanje 02

Sadržaj

- Klasa String
- Wraper klase za primitivne tipove,
- Klasa ArrayList i objekti Wraper klasa,
- Formati ispisa na ekran,
- Klase StringBuilder, StringBuffer
- Klasa StringTokenizer,

Klasa String

- Niz karaktera je podržan klasom String. String nije samo niz karaktera – on je klasa!
- Od java 1.7 skladište se na heap memoriji u delu *String pool.*
- Objekti klase String se ne mogu menjati (*immutable*)!
- *Immutable Objects* je objekat kome se definiše vrednost u trenutku njegovog kreiranja. Za njega ne postoje metode, ni načini kako da se ta vrednost dodatno promeni.

Klasa String

- Prethodno omogućava optimizaciju memorije. U slučaju da više string promenljivih imaju isti tekstualni sadržaj tada se za njih kreira samo jedna vrednost u *String pool*–u (optimizacija) i sve promenljive dobijaju referencu ka toj vrednosti.

- To bi značilo da će p1, p2 i p3 pokazivati na istu vrednost u *String pool*–u.

```
String p1= "Tekst je ovo";
String p2= "Tekst " + "je ovo";
String temp = " je ";
String p3= "Tekst" + temp+ "ovo";
```

Klasa String

- Za cast-ovanje String-a u neki primitivni tip koristi se Wrapper klasa i njena metoda parseXxx():

```
int i = Integer.parseInt(s);
```

- Za poređenje Stringova se ne koristi operator ==, već funkcija equals ili equalsIgnoreCase

- Reprezentativne metode
  - `str.length()`
  - `str.charAt(i)`
  - `str.indexOf(s)`
  - `str.substring(a,b), str.substring(a)`
  - `str.equals(s), str. equalsIgnoreCase(s) - ne koristiti ==`
  - `str.toLowerCase()`

Klasa String

```
String s1 = "Ovo je";
String s2 = "je string";
System.out.println(s1.substring(2));  //  o je
System.out.println(s2.charAt(3));  // s
System.out.println(s1.equals(s2));  //false
System.out.println(s1.indexOf("je"));  // 4  , ako nema podstringa vratiće  -1
System.out.println(s2.length());  //9
System.out.println(s2.startsWith("je")); //true
```

Klasa String – metoda split()

- Metoda *split* "cepa" osnovni string na niz stringova po zadatom šablonu
  - originalni string se ne menja
  - parametar je regularni izraz

- rezultat je niz stringova na koje je „pocepan“ originalni string
- Poziv: `String[] rez = s.split("regex");`

- Alternativa ovomo je upotreba klase *StringTokenizer*

Klasa String – metoda split()

```
class SplitTest {
  public static void main(String args[]) {
    String text = "Ovo je probni tekst";
    String[] tokens = text.split(" ");
    for (int i = 0; i < tokens.length; i++)
      System.out.println(tokens[i]);
    }
  }
}
```

Konzola

```
-->Ovo
-->je
-->probni
-->tekst
```

Primer01

Wraper klase za primitivne tipove

- Za sve primitivne tipove postoje odgovarajuće klase:
  - int - Integer
  - long - Long
  - boolean - Boolean
- Imaju statičku metodu Xxxx.parseXxxx()
  - int i = Integer.parseInt("10")
  - long l = Long.parseLong("10")
- Vrednosti objekata Wrapper klasa se smeštaju na Heap
- Ove Wrapper klase rade automatski *boxing* i *unboxing*, odnosno automatsku konverziju primitivnih tipova u objekte i obrnuto kada je to potrebno.

Primer02

Klasa ArrayList i objekti Wraper klasa

- Predstavlja kolekciju, odn. dinamički niz
- U listu se dodaju Java objekti
- Elementi se u ArrayList dodaju metodom add(obj)
- Elementi se iz ArrayList uklanjaju metodom remove(i)
- Elementi se iz ArrayList dobijaju (ne uklanjaju se, već se samo čitaju) metodom get()

Klasa ArrayList i objekti Wraper klasa

```
ArrayList<Integer> lista = new ArrayList<Integer>();
lista.add(5);
lista.add(new Integer(5));
lista.add(1, 15);
System.out.println("Velicina je: " + lista.size());
lista.remove(0);
int broj = lista.get(0);
System.out.println(broj);
System.out.println("Velicina je: " + lista.size());
```

- Prolaz kroz listu

```
for (int i = 0; i < lista.size(); i++) {
    System.out.println("Broj je: " + lista.get(i));
}
```

Primer03

Formati ispisa na ekran

- Za ispis na ekran se koriste funkcije *print* i *println* koje očekuju tekst kao parametar
  - *print* – ispiši tekst
  - *println* – ispiši tekst i pređi kursorom u novi red

```
System.out.print("Poruka");
System.out.println("Poruka");
```

- Između otvorene i zatvorene zagrade dozvoljeno je izvršiti
  - konkatenacijua više stringova
  - konkatenaciju striga sa primitivnim tipovima
  - konkatenaciju stringa sa objektima (poziva se njihova *toString* metoda)

```
int ocena = 8;
System.out.println("Dobili ste ocenu: " + ocena);
```

Formati ispisa na ekran

- Funkcija *printf* omogućuje formatizovani ispis

```
int c = 356;
System.out.printf("celobrojni: %d\n", c);
-->celobrojni: 356
System.out.printf("celobrojni: %10d\n", c);
-->celobrojni: _______356
System.out.printf("celobrojni: %+10d\n", c);
-->celobrojni: ______+356
System.out.printf("celobrojni: %10d\n", -c);
-->celobrojni:_____-356
System.out.printf("celobrojni: %-10d\n", c);
-->celobrojni:356_______
```

Formati ispisa na ekran

```
//formatizovani ispis na ekran
System.out.printf("Ispis celog broja %d \n", 10);
-->Ispis celog broja 10
System.out.printf("Ispis karaktera %c \n", 'A');
-->Ispis karaktera A
System.out.printf("Ispis karaktera %c \n", 66);
-->Ispis karaktera B
System.out.printf("Ispis razlomljenog broja %f \n", 3.14);
-->Ispis razlomljenog broja 3.140000
System.out.printf("Ispis razlomljenog broja preciznosti 2
    decimale %5.2f \n", 3.123456789);
-->Ispis razlomljenog broja preciznosti 2 decimale _3.12
```

Primer04

Ipsis slova ćirilice i latinice š,ć,đ,č

- Na **windows** platformi se slova ćirilice ili latinice (š,ć,đ,č) ne mogu inicijalno sačuvati u okviru Java fajla jer je taj fajl sačuvan pod enkodingom Cp1252 koji ne podržava ta slova.
- Rešenja
  - Ukoliko se Java fajl sačuva u UTF-8 enkodingu pisanje navedenih slova je moguće
    - Potencijalni problem predstavlja kopiranje UTF-8 fajlova, gde se pri kopiranju kopija čuva pod Cp1252 enkodingom i napisana slova ćirilice ili latinice se gube
    - Napisati problematična slova korišćenjem njihovih UNICODE brojnih oznaka
      - đ - "\u0111", Đ- "\u0110“, š - "\u0161", Š- "\u0160“
      - ć - "\u0107", Ć- "\u0106“, č- "\u010D", Č- "\u010C“
      - ž - "\u017E", Ž- "\u017D“,
      - Više na http://www.fileformat.info/info/unicode/char/search.htm

Ipsis slova ćirilice i latinice š,ć,đ,č

- Za ispis Unicode karaktera u okviru konzole potrebno je:
  - Postaviti podešavanje za pokretanje java fajla
    - *Desni klik na klasu->Run As->Run Configurations...* ->pa iz liste pokrenutih programa selektujete željeni
    - Nad željenim programom sa desne strane odaberite karticu *Common-> Encoding*
    - Odaberite stavku *Other* i unesite vrednost UTF-8
  - ili podešavanje za pokretanje eclipse alata
    - na sam kraj fajla eclipse.ini ubaciti red -Dfile.encoding=UTF-8

Klase StringBuffer i StringBuilder

- Izmena stringa konkatenacijom ili dodelom novog string literala kreira se novi objekat na *heap* memoriji – alternativa *StringBuffer* ili *StringBuilder* klasa.
- Omogućavaju kreiranje teksta koji se može proširiti

- *StringBuffer* metode su sinhornizovane, dok metode *StringBuilder* nisu (manji overhed = efikasniji)

```
StringBuffer buf = new StringBuffer("Pocetni tekst ");
buf.append("dodatni tekst 1");
buf.append("dodatni tekst 2");
String konacniTekst = buf.toString();
```

Primer05

- Koristi uvek StringBuilder osim ako je potrebno deliti tekst između programskih niti.

Klasa StringTokenizer

- slične namene kao metoda split() klase String
- "cepa" osnovni string na delove po zadatom delimiteru/delimiterima
  - originalni string se ne menja
  - parametar je tekst koji se deli
  - delovi se dobijaju pozivom metode objekta tipa StringTokenizer

Klasa StringTokenizer

- Postoje dva načina (konstruktora) za kreiranja objekta
  - Konstruktor sa jednim parametrom i predefinisanim setom delimitera " \t\n\r\f" (razmak, tab, novi red, carriage-return, form-feed)
  - Konstruktor sa dva parametra, pri čemu je drugi parametar test u kome su navedeni delimiteri

```
StringTokenizer st = new StringTokenizer("this is a test", " ");
while (st.hasMoreTokens()) {
        System.out.println(st.nextToken());
}
```