Sortiranje nizova

Prva oblast

Dušan Simić

Katedra za Računarske Nauke

2025-12-17

1. Sortiranje

1.1 Sortiranje nizova u Javi

- Nizovi kao `[i]`, ne kao `ArrayList`
- Zasniva se na poređenju
  - Poređenje primitivnih tipova – trivijalno
  - Poređenje objekata/referencijalnih tipova – metoda za poređenje
- Imeplementacija metoda za sortiranje
  - `Arrays.sort();`
  - Podrazumevani poredak je rastući

1.1 Sortiranje nizova u Javi

- Poredenje instanci klase koju smo mi napravili nije podrazumevano
  - Java ne zna kako da ih poredi ako joj nismo rekli kako!
  - Kao programeri smo dužni da implementiramo metodu kojom će se porediti objekti naše klase i samim tim će moći i da se sortiraju
  - Klasa mora da implementira interfejs `Comparable`
  - Mogu se porediti samo objekti istog tipa!

2. Poređenje

2.1 `Comparable` interfejs

- Moramo implementirati metodu `compareTo` iz interfejsa `Comparable`
- Ova metoda vraća `int`: nula ako su objekti jednaki, vrednost veću od nule ako je trenutni (`this`) objekat veći od prosledenog a vrednost manju od nule ako je trenutni objekat manji.

2.2 Algoritam za `compareTo` metodu

```
1  Input: T, U
2  Output: i
3  for each property pair (t, u) from T and U do
4      if t > u then
5          i = 1
6          break
7      else if t < u then
8          i = -1
9          break
10     else
```

2.2 Algoritam za compareTo metodu

```
11        i = 0
12    end
13 end
14 return i
```

2.3 Uredno poređenje objekata

- Pozivanjem `compareTo` metode nismo osigurali da objekti postoje!

```
this.nekiString.compareTo(that.nekiString);
```

- Ovo može prijaviti grešku pri izvršavanju!
- Pravilno je proveriti objekte prvo po referenci a tek onda po vrednosti.
- Ukoliko su objekti jednaki po referenci oni su automatski jednaki.
  - Specijalan slučaj: ukoliko su oba objekta `null`, oni se smatraju jednakim.
- Ukoliko je jedan od objekata `null`, može se očekivati `NullPointerException`.

2.4 Primer za compareTo metodu

```
public class Automobil {
  private String model;
  private int godiste;
}
```

ova klasa potom implementira interfejs Comparable

```
public class Automobil implements Comparable<Automobil> {
  ...
}
```

2.4 Primer za compareTo metodu

```
@Override
public int compareTo(Automobil that) {
  // Prvo poredimo naziv modela
  int rezultat = Objects.compare(this.model, that.model,
String.CASE_INSENSITIVE_ORDER);
  // Ako je model isti, poredimo godiste
  if (rezultat == 0) {
    rezultat = this.godiste - that.godiste;
  }
  // Vracamo rezultat poredjenja
  return rezultat;
}
```

2.4 Primer za `compareTo` metodu

Imajući prethodnu klasu i implementiranu metodu u vidu, mi znamo sledeće stvari:

- Instance klase Automobil mogu da se porede zato što klasa ima implementiranu metodu `compareTo`, tj. implementira interfejs `Comparable`.
- Za dva automobila (`a1` i `a2`) znamo da je `a1` veći od `a2` ako je njegov model "veći" od modela `a2` a manji ako je model "manji".
- U slučaju da su modeli automobila jednaki, veći je onaj čije je godište veće.

2.5 Komparatori

- Kako porediti objekte po još nekom kriterijumu osim prirodnog?

2.5 Komparatori

- Kako porediti objekte po još nekom kriterijumu osim prirodnog?
- Komparator je klasa koja implementinra interfejs `Comparator`.
  - Ima metodu `compare(this, that)`.

2.5 Komparatori

```
class ComparatorAutomobilGodinaModel implements
Comparator<Automobil> {
  public int compare(Automobil a1, Automobil a2) {
    // Poredimo po godištu
    int rezultat = a1.getGodiste() - a2.getGodiste();
    if (rezultat == 0) {
      // Poredimo po modelu
      rezultat = Objects.compare(this.getModel(),
that.getModel(), String.CASE_INSENSITIVE_ORDER);
    }
    // Vraćamo rezultat poređenja
    return rezultat;
  }
}
```

2.6 Diskusija

- Zasto mora `Comparable`? Zašto samo ne implementirati `compareTo`?
- Kako pojednostaviti kod kada postoji mnogo polja u klasi i mnogo traženih kriterijuma za poređenje?