Objektovo orientované programovanie

Učiteľ:
Ing. Jozef Wagner PhD.

Učebnica:
https://oop.wagjo.com/

OPG
Teória 17

1. Usporiadanie prvkov
2. Prirodzené usporiadanie
3. Comparator

Prirodzené usporiadanie

Anglicky *Natural ordering*, predstavuje predvolené (defaultné) poradie prvkov.

Má ho väčšina štandardných dátových typov

- `Integer`, `Long`, `Double`, ... - číselne vzostupne
- `String` - lexikograficky (podľa Unicode)
- `LocalDate` - od najstaršieho dátumu
- `Enum` - podľa poradia deklarácie

Usporiadanie poľa čísel

```
// Vytvoríme pole čísel
Integer[] poleCisel = {4, 5, 2, 1, 3};

// 4, 5, 2, 1, 3

// Usporiadanie vzostupne (od najmenšieho po najväčšie)
Arrays.sort(poleCisel);

// 1, 2, 3, 4, 5

// Usporiadanie zostupne (od najväčšieho po najmenšie)
Arrays.sort(poleCisel, Comparator.reverseOrder());

// 5, 4, 3, 2, 1
```

Usporiadanie zoznamu čísel

```
// Zoznam čísel
List<Integer> cisla = new ArrayList<>();
cisla.add(4); cisla.add(5);
cisla.add(2); cisla.add(1);
cisla.add(3);

// Vytvorenie kópie zoznamu
// kopírovací konštruktor
List<Integer> cislaVzostupne = new ArrayList<>(cisla);

// Prirodzené zotriedenie vzostupne
Collections.sort(cislaVzostupne);

// 1, 2, 3, 4, 5
```

Comparator

Ak chceme iné ako prirodzené usporiadanie, alebo ak trieda nemá svoje prirodzené usporiadanie, musíme určiť vlastné pravidlá.

Vlastné usporiadanie sa implementuje pomocou rozhrania `java.util.Comparator`

Comparator je malé rozhranie, ktoré vie porovnať dva objekty a povedať, ktorý má ísť skôr.

```
public interface Comparator<T> {
    int compare(T o1, T o2);
}
```

Comparator

Základom Comparatora je metóda `compare`, pomocou ktorej bude Java vedieť porovnať 2 objekty.

Táto metóda má vrátiť číslo podľa nasledovných pravidiel:

- vráti číslo **menšie ako 0** ak o1 je „menší“ ako o2
- vráti **0** ak je rovnaké poradie
- vráti číslo **väčšie ako 0** ak o1 je „väčší“ ako o2

Comparator príklad

```
class IntegerComparator implements Comparator<Integer> {
    @Override
    public int compare(Integer a, Integer b) {
        if (a < b) return -1;
        if (a > b) return 1;
        return 0;
    }
};

// ...

Collections.sort(cisla, new IntegerComparator());
```

Zotriedenie objektov podľa atribútu

Typické použitie Comparatora je pri usporiadaní objektov našich vlastných tried, ktoré nemajú prirodzené usporiadanie.

```
public class Osoba {
    private final String meno;
    private final String priezvisko;
    private final Pohlavie pohlavie;
    private final LocalDate datumNarodenia;
    public Osoba(String meno, String priezvisko,
                 Pohlavie pohlavie, LocalDate datumNarodenia) {
        this.meno = meno;
        this.priezvisko = priezvisko;
        this.pohlavie = pohlavie;
        this.datumNarodenia = datumNarodenia;
    }
    public String getMeno() {return meno;}
    public String getPriezvisko() {return priezvisko;}
}
```

Zotriedenie objektov podľa atribútu

Keďže dátum narodenia je typu `LocalDate` a táto trieda má prirodzené usporiadanie, vieme toto prirodzené usporiadanie využiť, a to pomocou volania metódy `compareTo`, ktorá porovnáva 2 objekty podobne ako comparator.

```
class NarodenieComparator implements Comparator<Osoba> {
  @Override
  public int compare(Osoba a, Osoba b) {
    return a.getDatumNarodenia().compareTo(b.getDatumNarodenia());
  }
}
```

```
List<Osoba> zoradene = new ArrayList<>(osoby);
Collections.sort(zoradene, new NarodenieComparator());
```

Zotriedenie textu

Prirodzené usporiadanie Stringov je lexikograficky, ktoré vyhovuje pre väčšinu prípadov. Ak však chceme porovnávať text v slovenčine, lexikografické porovnanie nám pri diakritike bude robiť problém.

```
class PriezviskoComparator implements Comparator<Osoba> {
    @Override
    public int compare(Osoba a, Osoba b) {
        return a.getPriezvisko().compareTo(b.getPriezvisko());
    }
}

// ...

List<Osoba> zoradene = new ArrayList<>(osoby);
Collections.sort(zoradene, new PriezviskoComparator());
```

Zotriedenie textu

```
Podla priezviska:
Andrej Bartoš, 1974-02-21: Javorová 5, Martin
Peter Horváth, 1978-11-04: Dlhá 7, Trnava
Martina Kováčová, 1992-07-25: Školská 45, Košice
Veronika Králiková, 2001-08-02: Lipová 10, Trenčín
Zuzana Mihálková, 1982-05-06: Líščia 21, Prešov
Ján Novák, 1985-03-12: Hlavná 12, Bratislava
Lucia Tóthová, 2000-01-18: Mlynská 3, Nitra
Tomáš Urban, 1999-12-14: Borovicová 6, Banská Bystrica
Marek Varga, 1995-09-30: Kvetná 18, Žilina
Katarína Šimková, 1988-10-09: Dubová 8, Piešťany
```

Ako vidno z výstupu, posledná osoba je na zlom mieste. Je to kvôli diakritike, kde písmeno Š je lexikograficky podľa Unicode až za klasickými písmenami

Zotriedenie textu

```
class PriezviskoComparator implements Comparator<Osoba> {
  Collator collator;

  {
    // Collator nakonfigurujeme v inicializačnom bloku triedy
    collator = Collator.getInstance(new Locale("sk", "SK"));
    collator.setStrength(Collator.SECONDARY);
  }

  @Override
  public int compare(Osoba a, Osoba b) {
    return collator.compare(a.getPriezvisko(), b.getPriezvisko());
  }
}
```

Zotriedenie podľa viacerých atribútov

```
class MultiComparator implements Comparator<Osoba> {
  @Override
  public int compare(Osoba a, Osoba b) {
    int r = a.getPohlavie().compareTo(b.getPohlavie());

    // Ak prvé porovnanie objekty usporiadalo,
    // tak už nemusíme porovnávať podľa druhého atribútu
    if (r != 0)
      return r;

    return b.getDatumNarodenia().compareTo(a.getDatumNarodenia());
  }
}
```