Poznámka:

Třídy `BigInteger`, `BigDecimal` a `Date` budou vysvětleny v částech [15./183], [16./185] a [18.1./204]. □

14.4.1. Typický prvek kolekce pro české řazení

Předchozí příklad byl ukázkou, co by měl obsahovat prvek kolekce obecně. V následujícím příkladu, který bude předchozí příklad rozvíjet, se pokusíme typický prvek kolekce ještě dvěma způsoby vylepšit.

První vylepšení bude spočívat v tom, že řazení jmen umožníme podle pravidel českého řazení (zjednodušeně „s ohledem na akcenty" – podrobně viz [UJJ2/287]).

Druhé vylepšení se týká užitečného „přídavku" třídy – komparátorů. Má-li totiž třída více atributů, podle kterých lze objekty této třídy řadit, je velmi vhodné připravit i vnitřní třídy příslušných komparátorů. Tento postup již byl podrobně vysvětlen v [1.2.4./29]. Samozřejmě je vhodné zachovat i možnost přirozeného řazení.

Příklad:

Třída `OsobaProKolekce` bude mít čtyři atributy. Dva z nich budou neměnné řetězce ve významu křestního jména a příjmení. Podle těchto atributů bude také probíhat přirozené řazení. Další dva atributy budou výška a váha. Tyto atributy lze měnit příslušnými metodami `set...()`. Pro řazení podle nich jsou ve třídě připraveny statické vnitřní třídy komparátorů, takže uživatel třídy nemá s případným řazením podle jakéhokoliv atributu žádné další starosti.

Pro řazení podle jména (tj. příjmení a v případě shody i křestního jména) se používá statický atribut `COL`, který je nastaven pro českou lokalitu.

Hodnoty všech atributů jsou kontrolovány a v případě nevhodných hodnot jsou vyhazovány asynchronní (nekontrolované) výjimky `NullPointerException` pro nezadané křestní jméno a příjmení nebo `IllegalArgumentException` pro nulové či záporné hodnoty výšky a váhy.

Za zmínku stojí ještě komparátor `PODLE_VAHY`, ve kterém se poměrně složitě (ale bezpečně) porovnávají dvě čísla typu **`double`**. Je použito porovnávání atributů `longVaha` na rovnost a atributů `vaha` na větší/menší. Pokud by bylo zaručeno (což v případě váhy osoby lze předpokládat), že jednotlivé váhy nebudou rozdílné s nepatrnou odchylkou, pak lze místo složitého porovnání využít zakomentovaný rozdíl obou vah.

Ve výpisu si všimněte, že písmeno „Ch" je řazeno správně za „Č".

V poslední řádce výpisu je vidět, jak zafunguje ochrana před nepovolenou hodnotou atributu. Protože se jedná o použití výjimky odvozené od typu `RuntimeException`, nebylo tuto výjimku nikde třeba ošetřovat pomocí **`try-catch`** nebo deklarovat pomocí **`throws`**.

```
import java.util.*;
import java.io.*;
import java.text.*;

class OsobaProKolekce implements Comparable {
  // základní stavové atributy
  // neměnitelné
  private final String  krestni, prijmeni;
  // měnitelné
  private int vyska;
  private double vaha;
  // odvozený atribut – bitový obraz váhy pro hashCode() a equals()
  protected long longVaha;
  // pomocný atribut pro české řazení
  private static final Collator COL =
            Collator.getInstance(new Locale("cs", "CZ"));

  public OsobaProKolekce(String krestni, String prijmeni,
                         int vyska, double vaha) {
    if (krestni == null  ||  prijmeni == null)
      throw new NullPointerException();
    this.krestni = krestni;
    this.prijmeni = prijmeni;
    setVyska(vyska);
    setVaha(vaha);
  }

  public String getKrestni() {
    return krestni;
  }

  public String getPrijmeni() {
    return prijmeni;
  }

  public void setVyska(int vyska) {
    if (vyska <= 0)
      throw new IllegalArgumentException("vyska=" +
                                                  vyska);
    this.vyska = vyska;
  }
```

```
public void setVaha(double vaha) {
  if (vaha <= 0)
    throw new IllegalArgumentException("vaha=" + vaha);
  this.vaha = vaha;
  this.longVaha = Double.doubleToLongBits(this.vaha);
}

public int getVyska() {
  return vyska;
}

public double getVaha() {
  return vaha;
}

public String toString() {
  return prijmeni + " " + krestni + ", " + vyska +
                                    ", " + vaha + "\n";
}

public boolean equals(Object o) {
  if (o == this)
    return true;
  if ((o instanceof OsobaProKolekce) == false)
    return false;
  OsobaProKolekce opk = (OsobaProKolekce) o;
  boolean stejneKrestni = krestni.equals(opk.krestni);
  boolean stejnePrijmeni =
                          prijmeni.equals(opk.prijmeni);
  boolean stejnaVyska = vyska == opk.vyska;
  boolean stejnaVaha = longVaha == opk.longVaha;

  return stejneKrestni && stejnePrijmeni &&
                              stejnaVyska && stejnaVaha;
}

public int hashCode() {
  int vysledek = 17;
  vysledek = 37 * vysledek + krestni.hashCode();
  vysledek = 37 * vysledek + prijmeni.hashCode();
  vysledek = 37 * vysledek + vyska;
  int pom = (int) (longVaha ^ (longVaha >>> 32));
  vysledek = 37 * vysledek + pom;
  return vysledek;
}
```

```
  // přirozené řazení
  public int compareTo(Object o) {
    OsobaProKolekce opk = (OsobaProKolekce) o;
    int tmpP = COL.compare(this.prijmeni, opk.prijmeni);
    int tmpK = COL.compare(this.krestni, opk.krestni);
    return (tmpP == 0 ? tmpK : tmpP);
  }

  // komparátory pro absolutní řazení
  public static final Comparator PODLE_VYSKY =
                                        new Comparator() {
    public int compare(Object o1, Object o2) {
      return ((OsobaProKolekce) o1).vyska -
                            ((OsobaProKolekce) o2).vyska;
    }
  };

  public static final Comparator PODLE_VAHY =
                                        new Comparator() {
    public int compare(Object o1, Object o2) {
      double v1 = ((OsobaProKolekce) o1).vaha;
      double v2 = ((OsobaProKolekce) o2).vaha;
      long  lv1 = ((OsobaProKolekce) o1).longVaha;
      long  lv2 = ((OsobaProKolekce) o2).longVaha;
      if (lv1 == lv2)
        return 0;
      if (v1 > v2)
        return +1;
      else
        return -1;
//      return (int) (((OsobaProKolekce) o1).vaha -
//                            ((OsobaProKolekce) o2).vaha);
    }
  };

  public static final Comparator PODLE_JMENA =
                                        new Comparator() {
    public int compare(Object o1, Object o2) {
      OsobaProKolekce opk1 = (OsobaProKolekce) o1;
      OsobaProKolekce opk2 = (OsobaProKolekce) o2;
      int tmpP = COL.compare(opk1.prijmeni,
                                          opk2.prijmeni);
```

```
      int tmpK = COL.compare(opk1.krestni, opk2.krestni);
      return (tmpP == 0 ? tmpK : tmpP);
    }
  };
}

public class TestOsobaProKolekce {
  public static void main(String[] argv)
                                        throws IOException {
    OutputStreamWriter o =
              new OutputStreamWriter(System.out, "Cp852");
    PrintWriter p = new PrintWriter(o);

    ArrayList a = new ArrayList();
    a.add(new OsobaProKolekce("Karel","Chytrý",160,60.));
    a.add(new OsobaProKolekce("Karel","Čtvrtý",170,70.));
    a.add(new OsobaProKolekce("Karel","Ctvrtý",180,80.));
    a.add(new OsobaProKolekce("Josef","Čtvrtý",190,65.));
    a.add(new OsobaProKolekce("Kárel","Čtvrtý",200,90.));
    p.println(a);

    Collections.sort(a);
    p.println("Podle jména\n" + a);

    Collections.sort(a, OsobaProKolekce.PODLE_VYSKY);
    p.println("Podle výšky\n" + a);

    Collections.sort(a, OsobaProKolekce.PODLE_VAHY);
    p.println("Podle váhy\n" + a);

    Collections.sort(a, OsobaProKolekce.PODLE_JMENA);
    p.println("Podle jména\n" + a);
    p.flush();

    a.add(new OsobaProKolekce("Jan", "Záporný", 0, -5.));
  }
}
```

Poznámka:

Aby byly jednotlivé sloupečky atributů zarovnány pod sebou, doplnil jsem do výpisu jednu mezeru za otevírací hranatou závorku. Například původní první řádka výpisu vypadá na obrazovce takto: `[Chytrý Karel, 160, 60.0` □

Vypíše:

```
[ Chytrý Karel, 160, 60.0
, Čtvrtý Karel, 170, 70.0
, Ctvrtý Karel, 180, 80.0
, Čtvrtý Josef, 190, 65.0
, Čtvrtý Kárel, 200, 90.0
]
Podle jména
[ Ctvrtý Karel, 180, 80.0
, Čtvrtý Josef, 190, 65.0
, Čtvrtý Karel, 170, 70.0
, Čtvrtý Kárel, 200, 90.0
, Chytrý Karel, 160, 60.0
]
Podle výšky
[ Chytrý Karel, 160, 60.0
, Čtvrtý Karel, 170, 70.0
, Ctvrtý Karel, 180, 80.0
, Čtvrtý Josef, 190, 65.0
, Čtvrtý Kárel, 200, 90.0
]
Podle váhy
[ Chytrý Karel, 160, 60.0
, Čtvrtý Josef, 190, 65.0
, Čtvrtý Karel, 170, 70.0
, Ctvrtý Karel, 180, 80.0
, Čtvrtý Kárel, 200, 90.0
]
Podle jména
[ Ctvrtý Karel, 180, 80.0
, Čtvrtý Josef, 190, 65.0
, Čtvrtý Karel, 170, 70.0
, Čtvrtý Kárel, 200, 90.0
, Chytrý Karel, 160, 60.0
]
Exception in thread "main"
java.lang.IllegalArgumentException: vyska=0
        ...
```