Valentino Vranić

Objektovo-orientované programovanie

Objekty, Java a aspekty

Errata

Pracovná verzia, 20. február 2009

Slovenská technická univerzita
v Bratislave
2008

PUBLIKÁCIU PODPORILO ZDRUŽENIE

GRATEX IT INŠTITÚT

v rámci fondu GraFIIT www.gratex.com

© Ing. Valentino Vranić, PhD.

Lektori: doc. Ing. Pavol Herout, PhD.
doc. Ing. Ján Kollár, PhD.

Vydala Slovenská technická univerzita v Bratislave vo Vydavateľstve STU, Bratislava, Vazovova 5.

Text neprešiel jazykovou úpravou vydavateľstva.

Schválilo vedenie Fakulty informatiky a informačných technológií STU v Bratislave pre študijný program Informatika a študijný program Počítačové systémy a siete.

ISBN 978-80-227-2830-0

Errata

s. 5: Triedy `ZlyObor` a `Rytier` obsahujú triedu `Poloha`.
má byť: Objekty triedy `ZlyObor` a `Rytier` obsahujú objekty triedy `Poloha`.

s. 51 a 53: typu objektu na
má byť: typu referencie na objekt

s. 70: Typ návratovej hodnoty metód `getF1`() a `getF2`() triedy `Elipsa` má byť `Bod`, nie **int**. V triede chýbajú lsti|get| a **set** metódy pre atribúty lsti|a| a `b`. Opravený kód je v Prílohe A.

s. 98: V prvom riadku a v príklade slučky **for** s explicitne použitým iterátorom chýba typ, ktorým je parametrizovaný: namiesto len `Iterator` má byť `Iterator`<`Element`>.

s. 111: V príklade čítania zbajtov z reťazca znakov riadok `System.out.println`() za nekonečnou slučkou **while** je zbytočný.

s. 158–160: Všetky triedy a rozhrania majú byť **public**. V kóde by sa mala uplatňovať generickosť zoskupení.

s. 150: V diagrame prípadov použitia na obr. 14.16 hrany od účastníkov majú siahať až po prípady použitia, ku ktorým smerujú. Opravený obrázok je uvedený v Prílohe B.

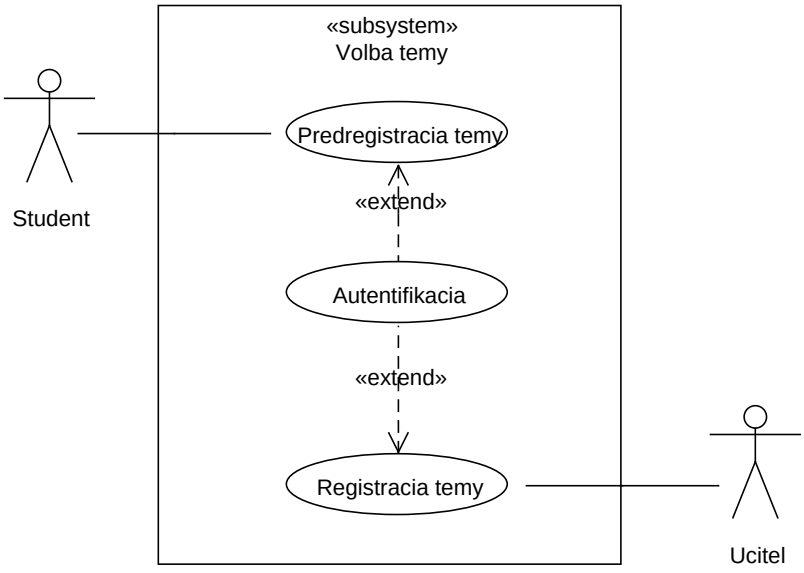
s. Kapitola 15: Všetky triedy a rozhrania uvedené v kapitole majú byť **public**. V triede `HumanTempSensor` chýba implementácia metódy `readTemp`(). V príkaze **switch** v metóde `display`() triedy `RelTemp` hodnoty vymenovaného typu `TempRange` nemajú byť kvalifikované jeho názvom. Aj keď to nie je nevyhnutné, v kóde by sa mala uplatňovať generickosť zoskupení. Opravený kód je v Prílohe C.

Opravený kód triedy Elipsa z Kapitoly 6

```
public class Elipsa extends Utvar {
   private Bod f1;
   private Bod f2;
   private int a;
   private int b;

   public Bod getF1() {
      return f1;
   }
   public Bod getF2() {
      return f2;
   }
   public void setF1(Bod f1) {
      this.f1.setX(f1.getX());
      this.f1.setY(f1.getY());
   }
   public void setF2(Bod f2) {
      this.f2.setX(f2.getX());
      this.f2.setY(f2.getY());
   }
   public int getA() {
      return a;
   }
   public int getA() {
      return b;
   }
   public void setA(int a) {
      this.a = a;
   }
   public void setB(int b) {
      this.b = b;
   }
}
```

Diagram prípadov použitia z Kapitoly 14



Kód vzorov z Kapitoly 15

C.1 Návrhový vzor Visitor

```
public interface DItem { // rozhranie Element
   void display(DDevice d); // prijatie návštevníka
}

public class WordItem implements DItem {
   String word;
   public WordItem(String s) {
      word = s;
   }
   public void display(DDevice d) {
      d.write(this);
   }
}

public class ListItem implements DItem {
   List<String> list;
   public ListItem(List<String> l) {
      list = l;
   }
   public void display(DDevice d) {
      d.write(this);
   }
}

public interface DDevice { // rozhranie Visitor
   void write(WordItem item);
   void write(ListItem item);
}
```

```
public class HorDevice implements DDevice {
   // navštívenie WordItem
   public void write(WordItem item) {
      for(int i = 0; i < item.word.length(); i++)
         System.out.print(item.word.charAt(i));
   }
   // navštívenie ListItem
   public void write(ListItem item) {
      for(int i = 0; i < item.list.size(); i++) {
         System.out.print(item.list.get(i));
         System.out.print(", ");
      }
   }
}

public class VerDevice implements DDevice {
   // navštívenie WordItem
   public void write(WordItem item) {
      for(int i = 0; i < item.word.length(); i++)
         System.out.println(item.word.charAt(i));
   }
   // navštívenie ListItem
   public void write(ListItem item) {
      for(int i = 0; i < item.list.size(); i++) {
         System.out.print(item.list.get(i));
         System.out.println(", ");
      }
   }
}

public class M {
  public static void main(String[] args) {
    List<String> list = new ArrayList<String>();
    list.add("a");
    list.add("b");
    list.add("c");
    DItem[] item = {new WordItem("visit"),
                    new ListItem(list)};
    DDevice[] device = {
       new HorDevice(), new VerDevice()};

    for (int i = 0; i < item.length; i++)
      for (int d = 0; d < device.length; d++) {
         item[i].display(device[d]);
```

```
            System.out.println("");
        }
    }
}
```

C.2 Návrhový vzor Observer

```
public interface TempDisplay { // rozhranie Observer
    void display();
    void measureTemp();
    void refresh(); // aktualizácia pozorovateľa
}

public interface TempSensor { // rozhranie Subject
    // pripoj pozorovateľa:
    void addDisplay(TempDisplay d);
    // odpoj pozorovateľa:
    void removeDisplay(TempDisplay d);
    // pošli notifikáciu pozorovateľovi:
    void notifyDisplays();
    double readTemp();
    public void measureTemp();
}

public class HumanTempSensor implements TempSensor {
    private List<TempDisplay> displays = new ArrayList<TempDisplay>();
    private double temp;
    public double refreshRate;

    public double readTemp() { return temp; }
    public void measureTemp() {
        // zistí teplotu z fyzickej jednotky
        notifyDisplays();
    }
    public void setTempDebug(double t) {
        temp = t;
    }
    public void addDisplay(TempDisplay d) {
        displays.add(d);
    }
    public void removeDisplay(TempDisplay d) {
        // . . .
    }
```

```
    public void notifyDisplays() {
        for (int i = 0; i < displays.size(); i++) {
            (displays.get(i)).refresh();
        }
    }
}

public class DigitalTemp implements TempDisplay {
    private HumanTempSensor sensor;
    private float temp;

    public DigitalTemp(HumanTempSensor s) {
        sensor = s;
    }
    public void refresh() {
        temp = (float)sensor.readTemp();
    }
    public void display() { // len dve desatinné miesta
        System.out.println(
            Math.round(temp * 100.0) / 100.0);
    }
    public void measureTemp() {
        sensor.measureTemp();
    }
}

public enum TempRange { LOW, NORMAL, HIGH }

public class RelTemp implements TempDisplay {
    private HumanTempSensor sensor;
    TempRange range;
    double high = 37.0;
    double low = 35.0;

    public RelTemp(HumanTempSensor s) {
        sensor = s;
    }
    public void refresh() {
        double temp = sensor.readTemp();

        if (temp <= low)
            range = TempRange.LOW;
        else if (temp >= high)
            range = TempRange.HIGH;
```

```
      else
         range = TempRange.NORMAL;
   }
   public void display() {
      switch (range) {
         case LOW:
            System.out.println("LOW");
            break;
         case HIGH:
            System.out.println("HIGH");
            break;
         default:
            System.out.println("NORMAL");
      }
   }
   public void measureTemp() {
      sensor.measureTemp();
   }
}

public class M {
   public static void main(String[] args) {
      HumanTempSensor s = new HumanTempSensor();

      DigitalTemp d1 = new DigitalTemp(s);
      s.addDisplay(d1);
      RelTemp d2 = new RelTemp(s);
      s.addDisplay(d2);

      s.setTempDebug(37.33333333);
      s.notifyDisplays();

      d1.display(); // 37.33
      d2.display(); // HIGH
   }
}
```