Uitwerking 3e deeltentamen Imperatief programmeren

3 november 2006

Opgave 1

Zie je eigen practicumuitwerking ☺

Opgave 2

a. De door de gebruiker bij het opstarten van het programma ingetikte woorden zijn in het programma beschikbaar als een String-array parameter van de methode `main`.
Je kunt een string aan de gebruiker tonen door aanroep van `System.out.println`

b. Het resultaattype van `getSource` is `Object`. Om dit te kunnen opslaan in een `TextField`-variabele, moet de programmeur aan de compiler door deze *cast* notatie aangeven dat er in de praktijk alleen maar objecten zullen uitkomen die inderdaad van de subklasse `TextField` zijn. Zonder cast keurt de compiler het programma af. Als er toch iets anders de bron is van de aktie (bijvoorbeeld een `Button`), dan treedt er runtime een exception op.

c. Een DataInputStream is zinvol als er een binaire file moet worden gelezen die (binair gecodeerde) ints en/of doubles bevat.

```
DataInputStream dis = new DataInputStream(
                          new FileInputStream(naam));
```

d. Je kunt een Iterator gebruiken om de elementen van een collection langs te lopen.
Iterator iter = coll.iterator();
while (iter.hasNext()) doeIetsMet(iter.next());

e. Een `Graphics`-object bevat variabelen voor penkleur, font, clipping-area, oorsprong, tekenmodus, (en read-only achtergrondkleur), *maar niet: windowsize, bitmap, hoekpunten van het te tekenen vierkant/cirkel enz.*
Een `Graphics2D`-object bevat bovendien variabelen voor achtergrondkleur, Stroke (oftewel lijndikte, lijnstijl), Paint (oftewel opvulstijl, -patroon, -gradient), Transform (oftwel rotatie en schaling), kwaliteitshints (zoals anti-aliasing), *maar niet: Shape, Key.*

Opgave 3

a.

```
public boolean raak(int x, int y)
{
     return x>=loc.x-5 && x<=loc.x+5
         && y>=loc.y-5 && y<=loc.y+5;
}
```

b.

```
Collection<Stad> steden = new LinkedList<Stad>();
Test parent;
```

c.

```
     BufferedReader br = new BufferedReader
                              (new FileReader(naam));
     String s;
     while ( (s=br.readLine()) != null )
     {    Scanner scan = new Scanner(s);
          int x = scan.nextInt();
          int y = scan.nextInt();
          steden.add( new Stad(
                           scan.next(), new Point(x,y) ));
     }
```

d.

```
    public Stad welkeStad(int x, int y)
    {     for (Stad s : steden)
          if (s.raak(x,y))
               return s;
          return null;
    }
    public void stadPlus(Stad s)
    {     parent.a.append(s.getNaam() + "\n");
          s.increment();
    }
```

e.

```
    public void paint(Graphics g)
    {   for (Stad s : steden)
           s.teken(g, this.kleurVan(s) );
    }
```

f.

```
     public void verwerk(String str)
     {    for (Stad s:steden)
               s.reset();
          Scanner scan = new Scanner(str);
          while (scan.hasNext())
          {    Stad s = this.welkeNaam(scan.next());
               if (s!=null) s.increment();
          }
          repaint();
     }
```

met een hulp-methode `welkeNaam` *om het niet al te ingewikkeld te maken. Velen hadden die in de code van* `verwerk` *verwerkt, wat natuurlijk ook goed is.*

```
    public Stad welkeNaam(String str)
    {     for (Stad s:steden)
              if (s.getNaam().equals(str))
                  return s;
          return null;
    }
```

g.

In ieder geval moet je een subklasse van Kaart maken, om het herbruikbare deel te kunnen hergebruiken.

```
class Kaart2 extends Kaart
{
```

Voor het zoeken van de grootste maken we een extra methode (kan ook in een van de andere methodes erbij gezet worden).

```
    public int grootste()
    {   int max = 0;
        for (Stad s : steden)
           if (s.value()>max)
                max = s.value();
        return max;
    }
```

Wie moet nu deze methode aanroepen? Ik had in gedachten om de methode paint uit te breiden, om de grootste waarde in een object-variabele te stoppen. Je kunt dan mooi super gebruiken voor het oorspronkelijk paint-werk.

```
    int hoog;
    public void paint(Graphics g)
    {   hoog = this.grootste();
        super.paint(g);
    }
```

En nu kun je kleurVan her-definieren om deze variabele te inspecteren:

```
     public Color kleurVan(Stad s)
     {    if (s.value()>=hoog)
               return Color.RED;
          return super.colorOf(s);
     }
```

Velen hadden echter de aanroep van `grootste` *direct in de methode* `kleurVan` *gezet, of zelfs de bijbehorende code daar direct ingezet. Dat is schreeuwend inefficient (de grootste wordt voor elk cirkeltje opnieuw berekend) maar niet fout. En zo kan het dus zelfs zonder* `super` *nodig te hebben ☹ :*

```
     public Color kleurVan(Stad s)
     {    if (s.value()>=grootste())
               return Color.RED;
          return Color.GREEN;
     }

}
```