J. Dietel, J. Fuchs FH Aachen, Campus Jülich; IT Center, RWTH Aachen

Präsenzaufgaben 8 27./28.11.2024

Die Lösung der Aufgaben wird am Ende der Übung von Ihnen vorgestellt.

Geometrische Figuren

In dieser Aufgabe sollen geometrische Figuren grafisch auf dem Schirm angezeigt werden. Laden Sie sich dazu bitte von der Veranstaltungsseite die Quelldatei `Grafik.java` herunter. Sie liefert einen Rahmen zur Anzeige Ihrer geometrischen Figuren auf dem Bildschirm und besteht aus den Klassen `Grafik` und `Figur`. `Grafik` hat diese Methoden und Konstruktoren:

- ```
  /* Oeffnet ein Grafikfenster mit einer Zeichenflaeche
  der angegebenen Breite und Hoehe */
  public Grafik(int width, int height)
  ```

- ```
  /* Fuegt die genannte geometrische Figur in die Zeichenflaeche ein */
  public void add(Figur f)
  ```

- ```
  /* Entfernt die geometrische Figur aus der Zeichenflaeche */
  public void remove(Figur f)
  ```

- ```
  /* Aktualisiert die Zeichenflaeche. Aenderungen werden erst sichtbar,
  nachdem diese Methode aufgerufen wurde. Wartet vorher ms
  Millisekunden. Damit ist eine einfache Animation moeglich */
  public void aktualisiere(int ms)
  ```

Ihre geometrischen Klassen müssen die Klasse `Figur` ableiten. `Figur` besitzt zwei abstrakte Methoden. Abstrakte Methoden *müssen* in den Unterklassen überschrieben werden. Dies sind die Methoden:

```
public abstract int getPunktCount()
```

```
public abstract Point getPunkt(int i)
```

`getPunktCount` gibt die Anzahl der zur Figur gehörenden Punkte zurück. (Die Klasse `Point` ist eine Java-Klasse aus dem Paket `java.awt.`). `getPunkt` gibt den zu einem Index gehörenden Punkt zurück. In der Grafik werden Verbindungslinien zwischen den jeweils benachbarten Punkten der Figur gezeichnet.

Außerdem besitzt `Figur` die Methode

```
public void verschiebe(int x, int y)
```

die alle Punkte der Figur um den Vektor `(x,y)` verschiebt.

a) **Linie**

Laden Sie die Klassen `Linie` und `LinieTest` von der Veranstaltungsseite und machen Sie sich anhand des Beispiels die Funktionsweise des Programms klar.

b) **Rechteck**

Schreiben Sie eine Klasse für ein achsenparalleles Rechteck. Verwenden Sie als Attribute nur die zwei gegenüberliegenden Punkte, die im Konstruktor übergeben werden. Fügen Sie die folgenden Konstruktoren und Methoden hinzu:

- ```
  /* Konstruktor aus zwei gegenueberliegenden Eckpunkten */
  public Rechteck(Point p1, Point p2)
  ```
- ```
  /* Kopierkonstruktor */
  public Rechteck(Rechteck r)
  ```
- ```
  /* Skaliert das Rechteck mit dem Faktor d. Der Mittelpunkt des
  Rechtecks bleibt erhalten */
  public void skaliere(double d)
  ```
- ```
  /* --> Rechteck: Hoehe = ..., Breite= ... */
  public String toString()
  ```

Testen Sie Ihre Implementation mit folgendem Code:

```
Grafik window = new Grafik(610,610);
Rechteck r0 = new Rechteck(new Point(10,10), new Point(300,300));
Rechteck r1 = new Rechteck(r0);

Rechteck r2 = new Rechteck(new Point(600,300), new Point(300,10));
Rechteck r3 = new Rechteck(r2);

Rechteck r4 = new Rechteck(new Point(10,600), new Point(300,300));
Rechteck r5 = new Rechteck(r4);

Rechteck r6 = new Rechteck(new Point(600,300), new Point(300,600));
Rechteck r7 = new Rechteck(r6);
Rechteck[] testArray = {r0,r1,r2,r3,r4,r5,r6,r7};
System.out.println(r0);
r0.skaliere(0.5);
r2.skaliere(0.5);
r4.skaliere(0.5);
r6.skaliere(0.5);
for (int i = 0; i < testArray.length; i++) {
      window.add(testArray[i]);
}
```

c) **Doppeltes Rechteck**

Schreiben Sie eine anonyme innere Klasse, die von Rechteck erbt und zwei Rechtecke zeichnet. Die Klasse soll einen weiteren Punkt als Attribut erhalten. Dieser Punkt ist die Spiegelung von p2 an p1. Also, wenn p1=(100,100) und p2=(150,150), dann ist der gespiegelte Punkt p=(50,50). Die zwei Rechtecke, die gezeichnet werden sollen, sind definiert über die Punkte p1,p2 und p1,p. Überschreiben Sie die Methoden: `getPunktCount()`, `getPunkt(int i)`, `skaliere(double d)` und testen Sie Ihre Implementation mit den folgenden Zeilen:

```
window.add(zweiRechtecke);
zweiRechtecke.skaliere(2);
```

d) **Dreieck**

Schreiben Sie eine anonyme innere Klasse, um ein gleichseitiges Dreieck zu zeichnen. Erben Sie dafür von der Klasse Figur. Für zwei gegebene Punkte lässt sich ein dritter Punkt folgendermaßen berechnen:

```
newPoint((int)((p1.x+p2.x+Math.sqrt(3)*(p2.y-p1.y))/2),
         (int)((p1.y+p2.y+Math.sqrt(3)*(p2.x-p1.x))/2));
```

Implementieren Sie die Methoden: `getPunktCount()`, `getPunkt(int i)`, `skaliere(double d)` und `toString()`.

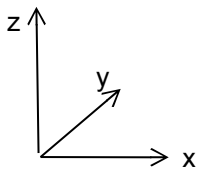
e) **Kreis**

Schreiben Sie eine Klasse für einen Kreis. Konstruieren Sie einen Kreis aus 100 kleinen Sekanten. Benutzen Sie die Sinus- und Cosinus-Funktion zur Berechnung der Endpunkte der Sekanten. Fügen Sie folgende Methode hinzu:

```
/*-> Kreis: Mittelpunkt = ..., Radius = ...*/
public String toString()
```

f) **Zusatzaufgabe: Quader**

Schreiben Sie eine Klasse für einen dreidimensionalen Quader. Die Projektion der y-Achse soll in Richtung (1,1) gehen, so wie in der Zeichnung dargestellt. Die Skalierung der y-Achse ist gegenüber der x- und der z- Achse um den Faktor $\sqrt{2}$ gestaucht.



Schreiben Sie einen Konstruktor:

```
public Quader(Rechteck x, int tiefe)
```

Die Dimensionen des Rechtecks ergeben die x- und die z-Achse des Quaders, die Tiefe ergibt die y-Achse des Quaders. Beachten Sie, dass Sie einen Würfel nicht in einem Zug zeichnen können, sondern Linien mehrfach zeichnen müssen.

Hinweise:

Sie können den Code von `Grafik.java` gerne modifizieren, falls Sie Swing beherrschen und wissen, was Sie tun. Lassen Sie die Vorlage ansonsten lieber unangetastet (dass das möglich ist, ist ein großer Vorteil der objektorientierten Programmierung). Die Methode `paint` aus `Figur` ist `public`, um den Code kurz zu halten. Die Methode ist aber nur zur internen Benutzung gedacht.