

Übungen zu „Objektorientierte Programmierung in Java“
PD Dr. Wolfgang Goerigk
Sommersemester 2009

Musterlösungen Serie 4

Aufgabe 4.1 (Laboraufgabe, Ein wenig Graphik)

Implementieren Sie die Klassen `Circle` und `GraphicsCircle` aus der Vorlesung. Beachten Sie dabei folgendes:

- Die Graphik-Routinen (z.B. `drawOval ...`) skalieren Graphikelemente in Anzahl von Pixeln und erwarten entsprechend Integer-Argumente sowohl für Koordinaten als auch für Längenangaben (Dimensionen).
- Die Methoden `drawOval (int x, int y, int width, int height)` und auch `drawRect (int x, int y, int width, int height)` (zeichne ein Rechteck, s.u.) zeichnen die graphischen Elemente an der Stelle (x, y) nach rechts und nach unten, d.h. ein Kreis wird nicht mit Mittelpunkt (x, y) , sondern mit Mittelpunkt $(x+r, y+r)$ gezeichnet. Analoges gilt für Rechtecke.

Ändern Sie die Definitionen aus der Vorlesung derart, dass sowohl Kreismittelpunkt als auch Radius Attribute vom Typ `int` sind.

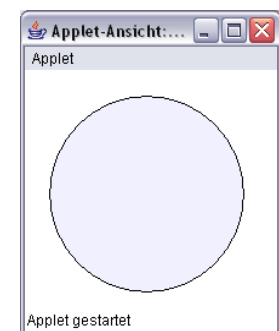
Instanziieren Sie die Klasse `GraphicsCircle` nach Belieben und zeichnen Sie die Kreise durch Aufruf der Methode `void draw (Graphics g)` der Klasse `GraphicsCircle`. Benutzen Sie dazu ein Applet nach dem folgenden Schema und starten Sie es als Java Applet.

```
import java.applet.Applet;
import java.awt.*;

public class Aufgabe_4_1 extends Applet {

    public void init () {
        this.setSize(400, 400);
        this.setVisible(true);
    }
    public void start () {
        Graphics g = this.getGraphics();

        /**
         * Hier folgt Ihr Programm
         */
    }
}
```



Hinweise:

Applets haben kein Hauptprogramm `main (...)`, sondern sie werden typischerweise im Kontext eines Web-Browsers aufgerufen, der sie bei Betreten der entsprechenden Webseite initialisiert (Aufruf von `init()`) und bei Mausklick startet (Aufruf von `start()`). Eclipse enthält einen integrierten Appletviewer, den Sie durch `Run as ... Java Applet` verwenden.

Lösung:

```
import java.applet.Applet;
import java.awt.*;

public class Aufgabe_4_1 extends Applet {

    public void init () {
        this.setSize(400, 400);
        this.setVisible(true);
    }

    public void start () {
        Graphics g = this.getGraphics();
        new GraphicsCircle(100, 100, 80).draw(g);
    }
}
```

```
public class Circle {

    int x,y,r;

    double umfang() {
        return 2 * Math.PI * r;
    }
}
```

```
import java.awt.*;

public class GraphicsCircle extends Circle {

    Color fill, outline;

    public void draw (Graphics g) {

        g.setColor(fill);
        g.fillOval(x-r, y-r, 2*r, 2*r);
        g.setColor(outline);
        g.drawOval(x-r, y-r, 2*r, 2*r);
    }

    public GraphicsCircle (int x, int y, int r) {
        this.x = x; this.y = y; this.r = r;
        fill = new Color (240, 240, 255);
        outline = Color.black;
    }
}
```

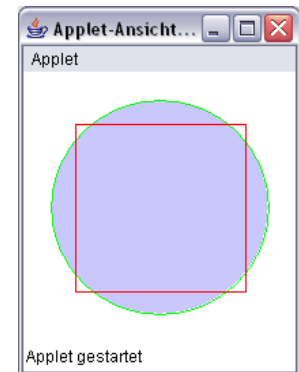
Die Klassen Circle und GraphicsCircle können für die Aufgabe 4.2 wiederverwendet werden. Deshalb enthält die Klasse Circle bereits eine Methode umfang().

Aufgabe 4.2 (Laboraufgabe, Ein wenig Umfang, ein wenig Rechteck)

Ergänzen Sie Ihr Programm aus Aufgabe 4.1 so, dass jeweils mit jedem Kreis auch ein entsprechendes Quadrat gleichen Umfangs ebenfalls mit Mittelpunkt (x, y) gezeichnet wird. Benutzen Sie dazu die Methode `umfang()` von Kreisen und beachten Sie, dass der Umfang eines Quadrates $4 \times$ die Seitenlänge beträgt.

Hinweise:

Ein Rechteck kann mit der Methode `drawRect (int x, int y, int width, int height)` der Klasse `java.awt.Graphics` gezeichnet werden, wobei auch `width` und `height` Integer-Parameter sind. Die Seitenlängen müssen also gerundet werden. Das geht am einfachsten durch ein `Typcast` (explizite Typumwandlung):



- Wird die Seitenlänge durch einen Ausdruck `e` vom Typ `float` oder `double` berechnet, dann ist `(int)(e)` der entsprechend gerundete Wert vom Typ `int`.

Lösung

Die Klassen `Circle` und `GraphicsCircle` sind wie in Aufgabe 4.1.

```
import java.applet.Applet;
import java.awt.*;

public class Aufgabe_4_2 extends Applet {

    public void init () {
        this.setSize(400, 400);
        this.setVisible(true);
    }

    public void start () {
        Graphics g = this.getGraphics();

        GraphicsCircle gc = new GraphicsCircle(100, 100, 80);

        gc.draw(g);

        int side = (int)(gc.umfang()/4);
        g.setColor(Color.red);
        g.drawRect(gc.x-side/2, gc.y-side/2, side, side);
    }
}
```

Aufgabe 4.3 (Hausaufgabe, Schleifen, ein Hin und Her)

1. Gegeben sei eine `while`-Schleife der Form:

```
while (<Bedingung>) {
    <Schleifenrumpf>
}
```

Schreiben Sie dafür eine äquivalente Anweisungsfolge mit einer `for`-Schleife beziehungsweise einer `do_while`-Schleife.

2. Gegeben sei die folgende `for`-Schleife:

```
for (<Initialisierung>; <Bedingung>; <Abschluss>) {  
    <Schleifenrumpf>  
}
```

Schreiben Sie dafür eine äquivalente Anweisungsfolge mit einer `while`-Schleife beziehungsweise einer `do_while`-Schleife.

Hinweise:

In beiden Fällen werden einzelne Anweisungen sicherlich außerhalb der jeweiligen Schleifen stehen müssen.

Lösung:

1.

```
for (;<Bedingung>;) {  
    <Schleifenrumpf>  
}
```

```
if (<Bedingung>)  
    do {  
        <Schleifenrumpf>  
    } while (<Bedingung>)
```

2.

```
<Initialisierung>  
while (<Bedingung>) {  
    <Schleifenrumpf>  
    <Abschluss>  
}
```

```
<Initialisierung>  
if (<Bedingung>)  
    do {  
        <Schleifenrumpf>  
        <Abschluss>  
    } while (<Bedingung>)
```