6. Übungsblatt

21 (+4) Punkte – Abgabe bis 31.01.2010 (08:00 Uhr) über GOYA

Aufgabe 1 ([Geometrische Objekte (*Vererbung*) — 21 Punkte])

Wir wollen Klassen bereitstellen, die es uns erlauben, geometrische Objekte wie Quadrate und Kreise zu verwalten und graphisch darzustellen. Damit wir die entsprechenden Objekte einheitlich verwenden können, sei folgende *abstrakte Klasse* Figure vorgegeben:

```
import java.awt.Color;

abstract class Figure {

  // Mittelpunkt der Figur
  protected double[] referencePoint;
  // Farbe der Figure
  protected Color c;
  // diese Methode soll die Figur malen
  abstract void draw();
  // diese Methode gibt den Flächeninhalt wieder
  double area() {
    return 0;
  }
  // Methode zum Neusetzen der Figurenfarbe
  void setColor(Color c) {
    this.c = c;
  }
}
```

Figure.java

Diese Klasse beschreibt ein geometrisches Objekt nur abstrakt. Sie legt fest, dass ein geometrisches Objekt

- einen *Mittel*punkt referencePoint haben muss,
- eine Farbe c für die graphische Ausgabe besitzen soll,
- eine Methode draw() implementieren muss, um das geometrische Objekt graphisch darzustellen,

- eine Methode area() bereitstellt, die den Flächeninhalt des Objektes zurück liefern soll, und
- eine Methode setColor() zum Ändern der Farbe anbietet.

Im Folgenden sollen Sie Ableitungen der Klasse Figure wie in Abb. 1 dargestellt implementieren.

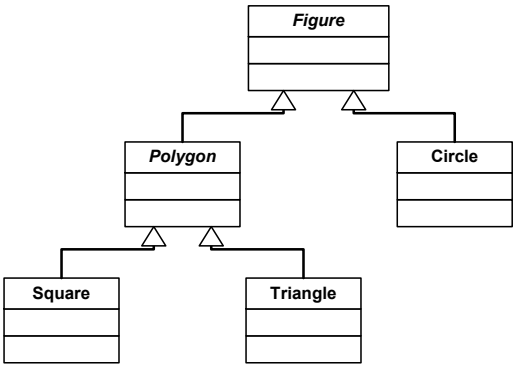

Abbildung 1: Figure und die entsprechenden Ableitungen.

1. *Kreis* (5 Punkte)
   Schreiben Sie eine Klasse Circle, die von der Klasse Figure abgeleitet ist. Diese Klasse soll einen Kreis beschreiben und aus einem Mittelpunkt und einem Radius bestehen.
   Der Konstruktor soll die Signatur Circle(double[] point, double radius) haben, wobei point den Mittelpunkt beschreibt und somit einen x und einen y Wert enthalten soll.
   Für die Ausgabe dürfen Sie die Methode StdDraw.filledCircle() aus dem Package gdp.stdlib benutzen.
   Der Flächeninhalt kann nach der bekannte Formel berechnet werden. Als Wert für π benutzen Sie bitte die Konstante Math.PI aus der entsprechenden Matheklasse, die durch Java bereit gestellt wird.

2. *Polygon* (3 Punkte)
   Schreiben Sie eine *abstrakte* Klasse Polygon, die von der Klasse Figure abgeleitet ist. Diese Klasse soll eine Oberklasse für *gleichseitige* n-Ecke sein und die Länge einer einzelnen Kante bereithalten. Außerdem soll diese Klasse bereits den Konstruktor implementieren, der über folgende Signatur verfügen soll: Polygon(double[] point, double length). Der Parameter point soll dabei den Mittelpunkt des n-Ecks angeben (als x– und y–Wert), length ist die Länge einer einzelnen Kante.

3. *Quadrat* (5 Punkte)
   Schreiben Sie eine Klasse Square, die von der Klasse Polygon abgeleitet ist. Ein

Quadrat soll aus einem Mittelpunkt und einer Kantenlänge bestehen. Überlegen Sie also, welche Variablen Sie in der Klassendefinition noch angeben müssen.
Für die Ausgabe des Quadrates dürfen Sie die Methode StdDraw.filledSquare() aus dem Package gdp.stdlib benutzen. Achten Sie dabei aber auf die Angabe der richtigen Kantenlänge.
Der Flächeninhalt kann nach der bekannten Formel berechnet werden.

4. *Dreieck* (8 Punkte)
   Schreiben Sie eine Klasse Triangle, die von der Klasse Polygon abgeleitet ist. Ein Dreieck[1] soll wie ein Quadrat lediglich aus einem Mittelpunkt und einer Kantenlänge bestehen.
   Für die Ausgabe dürfen Sie die Methode StdDraw.filledPolygon() aus dem Package gdp.stdlib benutzen. Dies bedeutet, dass Sie die eigentlichen Eckpunkte des Dreiecks in Abhängigkeit des Mittelpunktes erst noch berechnen müssen, um diese dann als Parameter für die Zeichenfunktion nutzen zu können.
   Der Flächeninhalt A berechnet sich nach der Formel
   $$A = \frac{a^2\sqrt{3}}{4},$$
   wobei a die Länge einer Seite ist. Sie dürfen die Methode Math.sqrt() benutzen, um die Wurzel einer Zahl zu brechnen.

Achten Sie bei der Implementierung darauf, Variablen und Methoden in der richtigen Klasse zu implementieren. Insbesondere überlegen Sie, welche Sichtbarkeit Ihre Variablen und Methoden haben sollen. Diese soll so restriktiv wie möglich sein.

Zusatzaufgabe (4 Punkte)

Leiten Sie eine Klasse CrazyCircle von der Klasse Circle ab, in der Sie die Methode draw() überschreiben und den Kreis nach Ihren Wünschen farbenfroh gestalten können (Beispiel siehe Abb. 2). Achten Sie jedoch darauf, dass der äußere Rand immer noch einem Kreis mit dem gegeben Radius entspricht, sodass auch der Flächeninhalt stimmt. Die über setColor() gesetzte Farbe soll in der Ausgabe wiederzufinden sein.

Abgabe

Je nachdem, welche der Teilaufgaben Sie gelöst haben, geben Sie bitte folgende Dateien über Goya ab:

- Circle.java
- Polygon.java
- Square.java
- Triangle.java
- CrazyCircle.java

[1] `https://secure.wikimedia.org/wikipedia/de/wiki/Gleichseitiges_Dreieck`

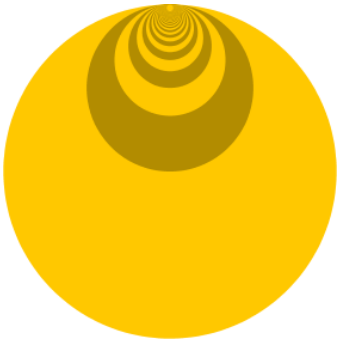

Abbildung 2: Farbenfroher Kreis

Testen

Nachdem Sie die obige Aufgabe abgearbeitet haben, können Sie Ihre Implementation zum Beispiel mit dem folgenden Programm FigureTest.java testen.

```
import gdp.stdlib.*;

public class FigureTest {

  public static void main(String[] args) {

    StdDraw.setXscale(-1, 1);
    StdDraw.setYscale(-1, 1);

    // Ein Quadrat
    double[] point1 = { 0, 0 };
    Figure s = new Square(point1, 0.5);

    // Ein Kreis
    double[] point2 = { 0.5, 0.5 };
    Figure c = new Circle(point2, 0.25);

    // Ein Dreieck
    double[] point3 = { -0.5, -0.5 };
    Figure t = new Triangle(point3, 0.5);

    s.setColor(StdDraw.BLUE);
    s.draw();
    System.out.println(s.area());

    c.setColor(StdDraw.RED);
    c.draw();
    System.out.println(c.area());

    t.setColor(StdDraw.GREEN);
    t.draw();
    System.out.println(t.area());
```

```
    }

}
```

Figure.java

Die Ausgabe, die das Programm erzeugen könnte, ist in Abb. 3 zu sehen.

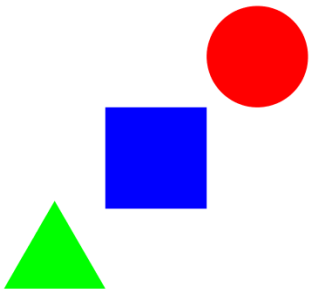

Abbildung 3: Ergebnis der Testausgabe

Auf der Standardausgabe ergeben sich folgende Flächeninhalte für das Quadrat, den Kreis und das Dreieck:

```
0.25
0.19634954084936207
0.10825317547305482
```

Hinweise

- Die Ausrichtung der Figuren ist nicht vorgegeben. Wichtig sind die richtige Position des Mittelpunktes und die richtige Größe der Figuren.