Universität der Bundeswehr München
Elektrotechnik und Technische Informatik
Oswald / Baumann

Wintertrimester 2021
Praktikum 08
Abgabe: 09.03.2021

Grundlagen der Programmierung

Vererbung

Aufgabe 1: Vererbung und Polymorphie

Für diese Aufgabe steht Ihnen wieder die Datei `GraphicsPanel.java` zur Verfügung. Außerdem erhalten Sie die Datei `Position.java`. Die Objekte der Klasse `Position` repräsentieren eine beliebige Position auf der Malfläche.

Erstellen Sie in einem neuen Netbeans-Projekt ein Package `edu.unibw.etti.graphics` und kopieren Sie die beiden Dateien in das Package. Die Klassen `GraphicsPanel`, `Graphics` (aus dem Package `java.awt`) und `Position` verwenden Sie zur Initialisierung der Malfläche am Besten wie folgt:

```
GraphicsPanel panel = new GraphicsPanel("Kreise", Position.X_MAX, Position.Y_MAX);
Graphics graphics = panel.createGraphics();
```

a) Schreiben Sie eine Klasse `Kreis`. Schreiben Sie einen Konstruktor für die Klasse, so dass es möglich ist mit der folgenden Anweisung ein `Kreis`-Objekt zu instanziieren:
`Kreis k = new Kreis(175.5, 210.5, 75.5, Color.red);`
Der erste beiden Parameter sind die x- und y-Koordinate des Kreismittelpunkts. Der dritte Parameter ist der Kreisradius und der vierte Parameter ist die Kreisfarbe. Verwenden Sie im Kreis zur Speicherung der Position die Klasse `Position`.

Setzten Sie alle Attribute Ihrer Klasse `Kreis` auf `private`.

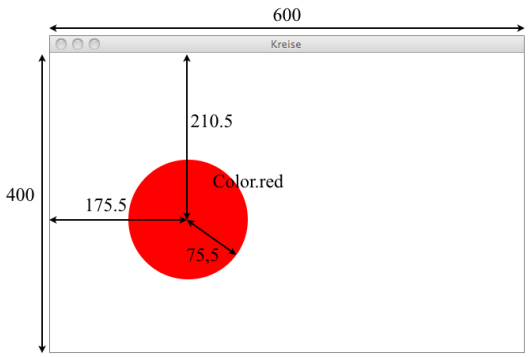

Figure 1: Roter Kreis mit Radius 75.5 und Mittelpunkt bei (175.5, 210.5)

b) Ergänzen Sie in der Klasse `Kreis` eine Methode, die das Anzeigen des Kreises ermöglicht:
`public void anzeigen(Graphics graphics)`
Die Methode soll einen - mit der im Konstruktor angegeben Farbe - gefüllten Kreis malen. Die Ausführung der folgenden `main`-Methode sollte zur einer Darstellung, wie in Abbildung 1, führen:

```
public static void main(String[] args) {
    GraphicsPanel panel = new GraphicsPanel("Kreise",
                                            Position.X_MAX, Position.Y_MAX);
    Graphics graphics = panel.createGraphics();
    Kreis k = new Kreis(175.5, 210.5, 75.5, Color.red);
    k.anzeigen(graphics);
    panel.updateGraphics();
}
```

c) Implementieren Sie jetzt eine Klasse `Snooker`. Die Klasse `Snooker` soll von der Klasse `Kreis` erben. Außerdem soll die Klasse `Snooker` zusätzlich eine Methode `public void bewegen()` erhalten, die ein Bewegen des Kreises ähnlich zu einer Billard- bzw. Snookerkugel ermöglicht (siehe Abbildung 2).

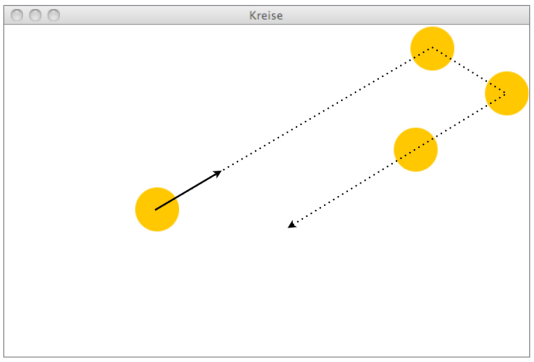

Figure 2: Oranger Kreis mit Radius 25.5, einem Startpunkt bei (175.5, 210.5) und einer initialen Bewegungsrichtung von (0.5, -0.3)

Die Ausführung der folgenden `main`-Methode:

```
public static void main(String[] args) {
    GraphicsPanel panel = new GraphicsPanel("Kreise",
                                            Position.X_MAX, Position.Y_MAX);
    Graphics graphics = panel.createGraphics();
    Snooker k = new Snooker(175.5, 210.5, 25.0, Color.orange, 0.5, -0.3);
    while (true) {
        graphics.setColor(Color.white);
        graphics.fillRect(0, 0, Position.X_MAX, Position.Y_MAX);
        k.anzeigen(graphics);
        panel.updateGraphics();
        k.bewegen();
    }
}
```

sollte dazu führen, dass initial ein oranger Kreis an der Position (175.5, 210.5) mit einem Radius von 25.0 erzeugt wird. Jeder Aufruf der Methode `bewegen` bewegt den Kreis um (0.5, -0.3) - bis er an eine *Bande* stößt. Je nachdem an welche Bande der Ball stößt wird die Bewegungsrichtung für den Ball geändert. Stößt der Ball oben bzw. unten an die Bande, dann muss die y-Richtung von negativ auf positiv bzw. von positiv auf negativ geändert werden. Stößt der Ball links bzw. rechts an die Bande, dann muss die x-Richtung von negativ auf positiv bzw. von positiv auf negativ geändert werden. Die initiale Bewegungsrichtung (0.5, -0.3) wird also im Konstruktor der Klasse `Snooker` zusätzlich zu den Parametern, die schon im Konstruktor der Klasse `Kreis` vorhanden sind, angegeben.

Müssen Sie die Klasse `Kreis` anpassen, um die Klasse `Snooker` zu implementieren?

d) Implementieren Sie jetzt eine Klasse `Ballon`. Die Klasse `Ballon` soll ebenfalls von der Klasse `Kreis` erben. Außerdem soll die Klasse `Ballon` zusätzlich eine Methode `public void bewegen()` erhalten, die dazu führt, dass der Kreis vergrößert und verkleinert wird (siehe Abbildung 3).

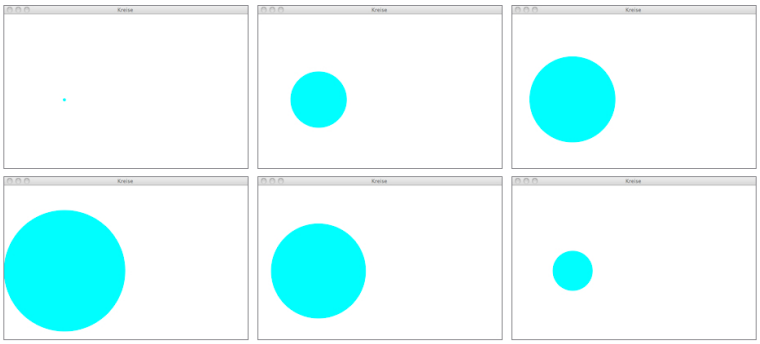

Figure 3: Türkiser Kreis mit einem Startpunkt bei (150.0, 210.5) und einem initialen Vergrößerungsschritt von 0.5

Die Ausführung der folgenden `main`-Methode:

```
public static void main(String[] args) {
    GraphicsPanel panel = new GraphicsPanel("Kreise",
                                            Position.X_MAX, Position.Y_MAX);
    Graphics graphics = panel.createGraphics();
    Ballon k = new Ballon(150.0, 210.5, Color.cyan, 0.5);
    while (true) {
        graphics.setColor(Color.white);
        graphics.fillRect(0, 0, Position.X_MAX, Position.Y_MAX);
        k.anzeigen(graphics);
        panel.updateGraphics();
        k.bewegen();
    }
}
```

sollte dazu führen, dass initial ein türkiser Kreis an der Position (150.0, 210.5) mit einem Radius von 1.0 erzeugt wird. Jeder Aufruf der Methode `bewegen` vergrößert den Kreis um die Schrittweite von 0.5 - bis er an eine *Bande* stößt. Stößt der Ball an eine der Banden, dann soll sich der Kreis in der gleichen Schrittweite, in der die Vergrößerung stattgefunden hat wieder verkleinern. Wird der Radius kleiner oder gleich 1.0, dann soll wiederrum die Vergrößerung stattfinden.

Die Schrittgröße, in der der Ballon vergrößert bzw. verkleinert wird, wird also im Konstruktor der Klasse `Ballon` zusätzlich zu den Parametern, die schon im Konstruktor der Klasse `Kreis` vorhanden sind, angegeben.

Müssen Sie die Klasse `Kreis` anpassen, um die Klasse `Ballon` zu implementieren?

e) Überlegen Sie, wie Sie Ihre Klasse `Kreis` anpassen können, damit die folgende `main`-Methode lauffähig ist:

```
public static void main(String[] args) {
    int varianten = 2;
    int anzahl = 5 * varianten;
    GraphicsPanel panel = new GraphicsPanel("Kreise",
                                            Position.X_MAX, Position.Y_MAX);
    Graphics graphics = panel.createGraphics();
    Kreis[] kreise = new Kreis[anzahl];
    for (int i = 0; i < kreise.length; i++) {
        double x = Math.random() * Position.X_MAX;
        double y = Math.random() * Position.Y_MAX;
        switch (i % varianten) {
            case 0:
                kreise[i] = new Snooker(x, y, 25.0, Color.red, 1.0, 1.0);
                break;
            case 1:
                kreise[i] = new Ballon(x, y, Color.blue, 1.0);
                break;
        }
    }

    while (true) {
        graphics.setColor(Color.white);
        graphics.fillRect(0, 0, Position.X_MAX, Position.Y_MAX);

        for (Kreis k : kreise) {
            k.anzeigen(graphics);
        }

        panel.updateGraphics();

        for (Kreis k : kreise) {
            k.bewegen();
        }

    }
}
```

f) Ergänzen sie die Klasse `Kreis` um zwei öffentliche Klassenmethoden:
`public static boolean kollidieren(Kreis k1, Kreis k2)`
und
`public static void mischeFarben(Kreis k1, Kreis k2)`.

Die Klassenmethode `kollidieren` soll `true` zurückliefern, wenn sich die beiden als Parameter angegebenen Kreise berühren. Wenn sich die Kreise nicht berühren, dann soll die Methode `false` zurückgeben.

Die Klassenmethode `mischeFarben` soll die Farben der beiden Kreise, die als Parameter übergeben werden, mischen und beiden Kreise sollen die neue gemischte Farbe erhalten. Um z.B. die beiden Farben gelb und grün zu mischen könnte man wie folgt vorgehen:

```
Color c1 = Color.green;
Color c2 = Color.yellow;
float[] f1 = c1.getComponents(null);
float[] f2 = c2.getComponents(null);
float[] fm = new float[f1.length];
for (int c = 0; c < fm.length; c++) {
    fm[c] = (f1[c] * 0.5f) + (f2[c] * 0.5f);
}
Color mixedColor = new Color(fm[0], fm[1], fm[2], fm[3]);
```

g) Überlegen Sie, wie Sie die in der Teilaufgabe f) angegeben `main`-Methode erweitern können, so dass in jeder `while`-Schleifenwiederholung geprüft wird, ob zwei Kreise kollidieren und falls Sie kollidieren Ihre Farben gemischt werden.

h) Schreiben Sie eine Klasse `Dominant`, die von der Klasse `Kreis` erbt. Die Kreise der Klasse sollen sich weder bewegen, noch Ihre Farbe ändern. Wie können Sie dies umsetzten? Evtl. müssen Sie den Rumpf Ihrer Klassenmethode `mischeFarben` nochmal anpassen.

Ergänzen Sie Objekte der Klasse `Dominant` in der `main`-Methode aus der letzten Teilaufgabe, in dem Sie die Variable `varianten` anpassen und die `switch-case`-Anweisung erweitern. Erzeugen Sie am Besten Kreise vom Typ `Dominant` mit zufälliger Farbe.