Universität der Bundeswehr München
Elektrotechnik und Technische Informatik
Oswald / Baumann

Wintertrimester 2021
Praktikum 07
Abgabe: 01.03.-02.03.2021

Grundlagen der Programmierung

Klassen und Objekte

Aufgabe 1: Geometrische Figuren

In dieser Aufgabe sollen Sie Klassen schreiben, die geometrische Figuren repräsentieren. Anschließend sollen Sie Instanzen dieser geometrischen Figuren auf dem Bildschirm visualisieren.

a) Schreiben Sie als Erstes eine Klasse `Position`. Die Klasse soll eine beliebige Position auf einer Fläche repräsentieren, d.h. die Klasse muss zwei private Objektattribute `x` und `y` vom Typ `double` beinhalten.

Schreiben Sie für die Klasse wenigstens einen sinnvollen Konstruktor, mit dem die Werte `x` und `y` gesetzt werden können, sowie die Abfragemethoden:
`public double getX()` bzw. `public double getY()`
und die Änderungsmethoden:
`public void setX(double x)` bzw. `public void setY(double y)`.

b) Ergänzen Sie in der Klasse `Position` zwei öffentliche Klassenkonstanten `X_MAX = 600` und `Y_MAX = 400` vom Typ `int`. Diese beiden Konstanten sollen die maximale x- und y-Ausdehnung der Fläche repräsentieren. Sorgen sie in den Methoden inkl. Konstruktor der Klasse `Position` dafür, dass die Werte für `x` und `y` immer im Bereich zwischen 0 und `X_MAX` bzw. `Y_MAX` liegen. Verwenden Sie dazu den Modulo-Operator.

Beispiele: Wurde `X_MAX` auf 100 gesetzt, dann sollte nach einem Aufruf von `setX(107)` der Wert von `x` gleich 7 sein (`107%X_MAX`). Wurde `Y_MAX` auf 200 gesetzt, dann sollte nach einem Aufruf von `setY(-107)` der Wert von `y` gleich 93 sein (`-107%Y_MAX + Y_MAX`).

c) In der Abbildung 1 sehen Sie einfache geometrische Figuren: Kreise, Rechtecke und gleichschenklige Dreiecke. Schreiben Sie für diese Figuren jeweils eine Klasse, also `Kreis`, `Rechteck` und `Dreieck` und verwenden Sie dabei u.a. den Datentyp `Position`.

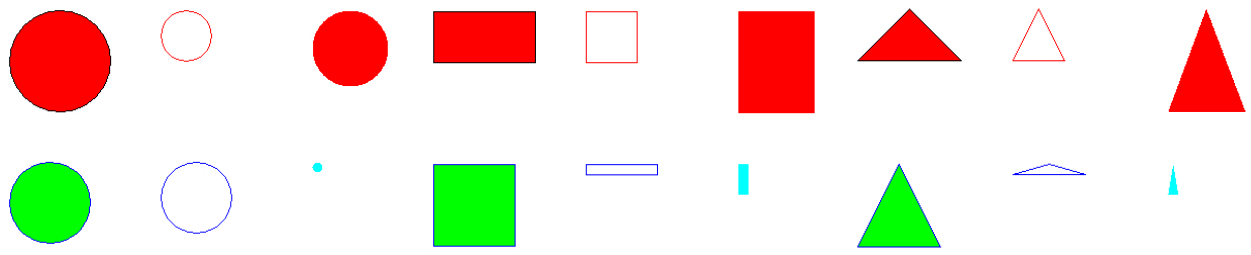

Figure 1: Einfache geometrische Figuren

Überlegen Sie, welche weiteren Objektattribute Sie benötigen, um die verschiedenen Eigenschaften der dargestellten geometrischen Figuren speichern zu können.

In der Abbildung 1 sehen Sie einfache geometrische Figuren: Kreise, Rechtecke und gleichschenklige Dreiecke. Schreiben Sie für diese Figuren jeweils eine Klasse `Kreis`, `Rechteck` und `Dreieck` und verwenden Sie dabei u.a. das Attribut vom Typ `Position`. Überlegen Sie, welche weiteren Objektattribute Sie benötigen, um die verschiedenen Eigenschaften der dargestellten geometrischen Figuren speichern zu können.

Erstellen Sie für jede Klasse wenigstens einen Konstruktor mit dem die in der Abbildung dargestellten Objekte instanziiert werden können. Setzen Sie alle Objektattribute auf 'private' und verwenden Sie zur Speicherung der Farben die Java-Klasse `java.awt.Color`.

d) In dieser Aufgabe sollen Sie die geometrischen Figuren malen. Machen Sie sich dazu mit der Java-Klasse `java.awt.Graphics` vertraut. Die Java-Klasse `Graphics` stellt im Prinzip eine Malfläche dar, auf die graphische Objekte gemalt werden können.

Zum Malen wird Ihnen die Datei `GraphicsPanel.java` zur Verfügung gestellt. Die Datei enthält die Klasse `edu.unibw.etti.graphics.GraphicsPanel` mit der Sie relativ einfach geometrische Figuren malen können. Bevor Sie die Klasse `GraphicsPanel` nutzen können, erstellen Sie am besten in Ihrem Projekt ein package `edu.unibw.etti.graphics` und kopieren Sie die Datei `GraphicsPanel.java` in das package.

Die Methode `createGraphics()` der Klasse `GraphicsPanel` liefert Ihnen eine leere weiße Malfläche. Die Größe des `Graphics`, die Ihnen die Methode `createGraphics()` liefert, wird identisch zur Größe des `GraphicsPanel` erstellt. Die Größe des `GraphicsPanel` kann im Konstruktor angegeben werden.

Um die Klasse `GraphicsPanel` nutzen zu können, erstellen Sie eine neue Klasse `Anwendung`, kopieren Sie nachfolgenden Sourcecode in deren `main`-Methode und führen Sie diese aus:

```
// Erzeugen des GraphicsPanel
GraphicsPanel panel = new GraphicsPanel("Mein Fenster", 600, 400);

// Erzeugen des Graphics, auf das die graphischen Objekte gemalt
// werden sollen und Oeffnen des Fensters. Beim erneuten Aufruf
// erhaelt man ein neues leeres Graphics.
Graphics g = panel.createGraphics();

// Setzen der Stiftfarbe
g.setColor(Color.red);

// Malen eines nicht gefuellten Kreises
g.drawOval(250, 150, 100, 100);

// Malen eines gefuellten Rechtecks
g.fillRect(50, 50, 100, 20);

// Neuzeichnen erzwingen, alles was bisher auf das Graphics gemalt wurde
// wird angezeigt. Der Aufruf fuehrt nicht dazu, dass das Graphics geleert wird.
panel.updateGraphics();
```

Ergänzen Sie nun jede der drei Klassen `Kreis`, `Rechteck` und `Dreieck` um eine Methode `public void anzeigen (Graphics g)`, die die jeweilige geometrische Figure auf `g` malt. Z.B. kann die Methode `drawRect` der Klasse `Graphics` verwendet werden, um ein nicht gefülltes Rechteck zu malen. Malen Sie die geometrischen Figuren ohne Verdrehung an die in `Position` angegebene Position.

Testen Sie nun, ob Sie ähnliche Objekte wie in der Abbildung 1 erzeugen und darstellen können. Erstellen Sie dazu in der `main`-Methode der Klasse `Anwendung` Rechtecke, Kreise und Dreiecke und rufen Sie deren Objektmethode `anzeigen` auf.

e) In Abbildung 2 sind ein Haus, ein Baum und ein Schneemann dargestellt. Erstellen Sie die Klassen `Haus`, `Baum` und `Schneemann`, so dass Häuser, Bäume und Schneemänner in beliebiger Größe und Position auf der Malfläche erzeugt werden können. D.h., Sie müssen zu jeder Klasse mindestens einen Konstruktor und eine `anzeigen`-Methode schreiben.

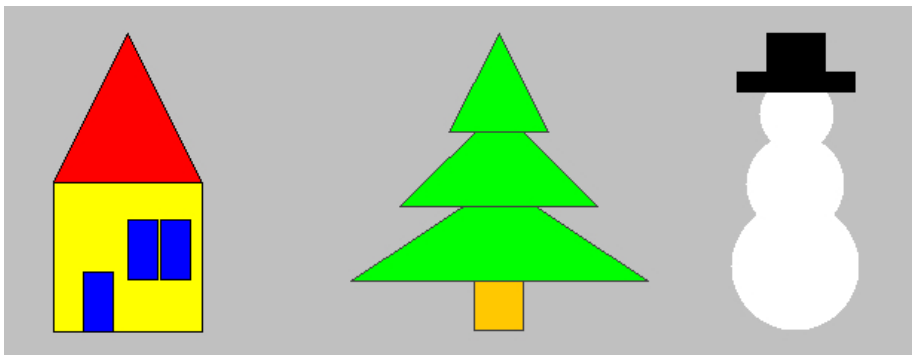

Figure 2: Komplexe graphische Objekte

Wie Ihr Haus, Baum und Schneemann genau aussieht bleibt Ihnen überlassen. Ihre Häuser, Bäume und Schneemänner sollen aber klar erkennbar sein. Ein Haus soll dabei wenigstens aus vier Rechtecken und einem Dreieck, ein Baum aus drei Dreiecken und einem Rechteck und ein Schneemann aus drei Kreisen und zwei Rechtecken bestehen. Die `anzeigen`-Methode muss alle Bestandteile des jeweiligen Objekts malen. Sie müssen dabei auf die Reihenfolge achten, in der die Bestandteile der Objekte gemalt werden, z.B. sollten die Fenster des Hauses erst gemalt werden, nachdem die Grundmauer des Hauses gemalt wurde.

Testen Sie Ihre Klassen in einer `main`-Methode.

f) *Freiwillige Zusatzaufgabe:* Jetzt wollen wir es noch schneien lassen, siehe Abbildung 3. Die Schneeflocken wollen wir mit Hilfe der Klasse `Kreis` darstellen.

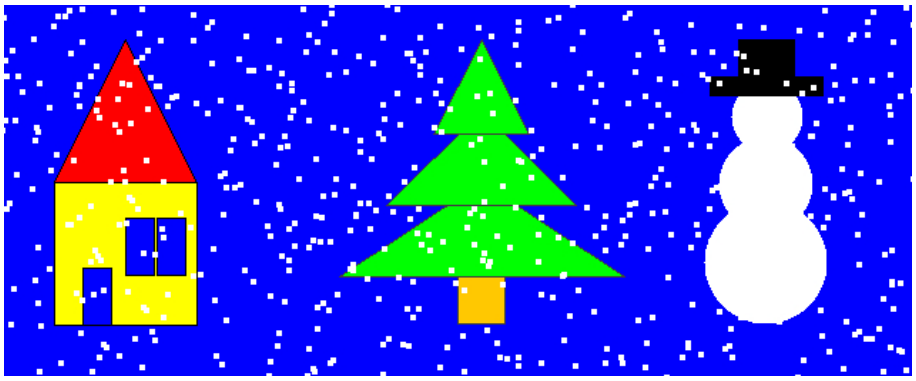

Figure 3: Bild mit Schneefall

Erstellen Sie dazu eine Klasse `Schnee`. Die Klasse soll den folgenden Konstruktor besitzen:
`public Schnee(int anzahl, double xMax, double yMax, double groesse)`.
Die `anzahl` legt fest, wie viele Schneeflocken bzw. Kreise der Schnee besitzt. Der Konstruktor erzeugt `anzahl` weiße Kreise mit einem Durchmesser von `groesse`. Die initiale Position der Kreise wird zufällig festgelegt. Die x-Koordinate muss zwischen 0 und `xMax` und die y-Koordinate zwischen 0 und `yMax` liegen. Überlegen Sie in welcher Datenstruktur Sie die Kreise speichern wollen.

Die `anzeigen`-Methode muss dieses Mal nicht nur alle Kreise malen, sondern auch die Position der Kreise nach dem Anzeigen ändern.
Dazu können Sie z.B. jede Flocke um `+ Math.random() - 0.5` in x-Richtung und um `+ 1.0` in y-Richtung verschieben. Ergänzen Sie, falls nötig, Methoden in der Klasse `Kreis`.

Testen Sie Ihre Klasse in einer `main`-Methode. Erstellen Sie dazu eine while-Schleife, die endlos läuft. In der Schleife wird jedes Mal ein blauer Hintergrund erzeugt, auf den Sie dann das Haus, den Baum und den Schneemann oder auch noch mehr Objekte malen. Außerdem malen Sie natürlich auch den Schnee. Vergessen Sie nicht, nachdem Sie alle Objekte gemalt haben, die Methode `updateGraphics` aufzurufen. Mit einem Aufruf der Methode `sleep` der Klasse `GraphicsPanel` am Ende der while-Schleife können Sie steuern, wie lange pausiert wird, bevor die while-Schleife wieder von vorne beginnt.