Projektblatt P12 (20 P)

Abgabe: Donnerstag 5. Dezember 2018, 12h
Entpacken Sie zunächst die Archiv-Datei `vorgaben-p12.zip`, in der sich die Rahmendateien für die zu lösenden Aufgabe befinden. Ergänzen Sie die Dateien durch Ihre Lösungen gemäß der Aufgabenstellung unten. Der hinzuzufügende Java-Sourcecode sollte **syntaktisch richtig** und **vollständig formatiert** sein. Alle Dateien sollten am Ende fehlerfrei übersetzt werden können.
Verpacken Sie die von Ihnen bearbeiteten `.java` Dateien für Ihre Abgabe in einem ZIP-Archiv mit dem Namen `IhrNachname.IhrVorname.P11.zip`, welches Sie auf Ilias hochladen.
Führen Sie dazu in dem Verzeichnis, in dem Sie die Dateien bearbeitet haben, folgenden Befehl auf der Kommandozeile aus:

```
zip  IhrNachname.IhrVorname.P12.zip  *.java
```

Nutzen Sie beim Bearbeiten der Aufgaben nur die Klassen und Methoden aus der Java API, welche explizit durch den Aufgabentext erlaubt sind.

Aufgabe 1: 6 + 10 + 4 Punkte

In früheren Zeiten wurden Cartoons (Zeichentrick) erstellt, indem ein Bild, einschließlich des Hintergrunds und aller Objekte, auf ein einzelnes Blatt Papier gezeichnet und dann fotografiert wurde. Aus einer Folge solcher Aufnahmen wurde dann ein Film generiert. Wenn die Unterschiede zwischen zwei aufeinanderfolgenden Bildern klein genug sind und die Anzahl der pro Sekunde angezeigten Bilder groß genug, entsteht der Eindruck einer (flüssigen) Bewegung. In einfacher Form ist dieses Prinzip auch bei sogenannten Daumenkinos zu beobachten.
In dieser Aufgabe sollen Sie nun eine Applikation vervollständigen, mit deren Hilfe Sie einfache Animationen erstellen können. Hierbei beschränken wir uns auf starre, geometrische Objekte, die durch affine Transformationen "bewegt" werden.
Folgende Klassen sind dazu (teilweise) vorgegeben:

ColoredShape Diese Klasse repräsentiert eine einzelnes Objekt, welches durch eine Form, eine Füll- und eine Randfarbe sowie durch eine Strichart (zum Zeichnen des Randes) beschrieben wird. Die Klasse besitzt zwei Konstruktoren: Mit dem ersten kann man eine Instanz der Klasse mit einer Form, zwei vorgegebenen Farben und einer Strichstärke initialisieren. Die Werte für die Farben können dabei den Wert `null` haben. Ein Nullwert für die Strichstärke führt dazu, dass die Strichstärke mit dem Wert 1 initialisiert wird. Darüberhinaus gibt es einen Kopierkonstruktor und eine vorgegebene Methode `getCenter`, die das Zentrum der Form bzw. der Bounding Box der Form bestimmt und als Punkt zurückgibt. Dazu sind mehrere, von Ihnen noch zu implementierende Methoden vorgegeben, mit denen die Form verschiedenen affinen Transformationen unterworfen werden kann. Auch die `paint`-Methode muss von Ihnen noch implementiert werden.

ShapeSequenceCreator Diese Klasse dient dazu, Felder von Objekten der Klasse `ColoredShape` zu generieren, die aus einem einzelnen solchen Objekt erzeugt werden, welches jeweils das erste Feldelement bildet. Die folgenden Elemente werden durch eine regelmäßige Anwendung einer affinen Transformation gebildet (z.B. durch eine gleichmäßige Verschiebung oder eine Rotation).

ShapeAnimFrame Diese Klasse stellt die Hauptklasse dar und muss von Ihnen aufgerufen werden um das Programm zu starten. Die Klasse

beinhaltet eine innere Klasse `AnimCanvas`, welche eine Zeichenfläche darstellt. Die Komponente besitzt ein Attribut, welches eine Liste darstellt, in der Felder von `ColoredShape` Instanzen gespeichert sind, die jeweils eine zusammengehörige Folge bilden.

Mit Hilfe eines Threads (welche wir im nächsten Semester behandeln werden), wird diese Zeichenfläche 50 Mal pro Sekunde neu gezeichnet. Bei jedem Zeichenvorgang wird die Komponente zunächst mit einem schwarzen Hintergrund gefüllt. Anschließend wird jeweils das i. Objekt aus jedem in der Liste gespeicherten Objektfeld gezeichnet. Die Felder werden dabei zyklisch durchlaufen.

Die Klasse besitzt eine Methode `addShapeSequence`, die Sie später nutzen sollen, um die von Ihnen generierten Felder mit Objekten der Klasse `ColoredShape` in die o.g. Liste einzufügen.

Windraedchen In dieser Klasse wird zunächst der Code aus der Vorlesung (Folie 37) genutzt um aus einem Dreieck, welches mit 9 verschiedenen Wínkeln um einen Eckpunkt rotiert wird, ein Windrad zu erstellen. Die rotierten Dreiecke werden dann in einem Objekt der Klasse `Area` zusammengefügt. Mit Hilfe der Klassenmethode `getShape` wird dieses Objekt dann zurückgegeben.

(a) Ergänzen Sie zunächst die Klasse `ColoredShape` wie folgt:

- Implementieren Sie die Methode `translate`, in der die Form horizontal um den Wert `dx` und vertikal um den Wert `dy` verschoben werden soll. Ersetzen Sie hierzu den Wert des Attributs `shape` durch das Objekt, welches sich ergibt, wenn Sie eine entsprechende Translations-Transformation auf den aktuellen Wert des Attributs anwenden.
- Implementieren Sie die Methode `rotate` (die, die zwei Parameter `theta` und `p` besitzt). In der Methode soll die Form um den Punkt `p` rotiert werden. Der Rotationswinkel ist dabei durch den Parameter `theta` gegeben.
- Implementieren Sie die Methode `scale`, die die Form um die als Parameter übergebenen Skalierungsfaktoren in x- bzw. y-Richtung skalieren soll. Damit das Zentrum der Form vor und nach der Skalierung am gleich Punkt liegt, bestimmen Sie das Zentrum (mit

Hilfe der vorgegebenen Methode `getCenter`) vor und nach der Skalierung und verschieben Sie die Form am Ende um die Differenzwerte in x- und y-Richtung.

- Vervollständigen Sie die `paint` Methode: In der Methode soll zunächst die Füllung der Form in der Farbe, die durch das Attribut `cInner` gegeben ist, in den Graphics Kontext gezeichnet werden. Danach soll der Rand der Form in der Farbe , die durch das Attribut `cBoundary` gegeben ist, in der Strichart, die durch das Attribut `stroke` gegeben ist. Falls das Attribut `cInner` bzw. `cBoundary` den Wert `null` hat, zeichnen Sie keine Füllung bzw. keinen Rand.

(b) Schreiben Sie nun eine neue Klasse, in der Sie eine eigene Form erstellen. Gehen Sie dabei analog zu dem in der Klasse `Windraedchen` gegebenen Beispiel vor: Erstellen Sie ein "interessantes" geometrisches Objekt in einem statisch ausgeführten Codeblock und geben Sie das Ergebnis durch die Methode `getShape` zurück. Das Objekt sollte sich entweder aus mehreren einfachen Teilen zusammensetzen und eine erkennbare Figur bilden (z.B. einen Schneemann, einen Tannenbaum etc.) oder eine komplexe geometrische Form oder Kurve darstellen (wie z.B. eine Spirale). Wählen Sie die geometrischen Abmessungen für Ihr Objekt so, dass die obere linke Ecke der zugehörigen Bounding Box im Nullpunkt des Koordinatensystems liegt.

(c) In der Klasse `ShapeSequenceCreator` soll zunächst eine Objekt der Klasse `ColoredShape` und daraus dann ein Feld von Objekten erstellt werden, dessen Elemente aus diesem Objekt durch Anwendung affiner Transformationen generiert wurden. Wenn die Objekte dann (im `AnimCanvas`) in kurzen Abständen nacheinander angezeigt werden, soll sich eine gleichförmige Bewegung ergeben (Rotation, Translation, Skalierung oder eine Kombination daraus). Im Konstruktor der Klasse ist der Code vorgegeben, mit dem zunächst ein einzelnes `ColoredShape` Objekt und dann ein Feld mit restlichen (inkrementell transformierten) Objekten erzeugt wird, welches dann dem `AnimCanvas` hinzugefügt wird. Das einzelne `ColoredShape` Objekt und das Feld sollen in zwei Methoden generiert werden.

Als Beispiel ist hier die Erzeugung eines "Windrads" und dessen Rotation in den Methoden `createShape_1()` und `createSequence_1` vorgegeben. Sie können dieses zum Testen Ihrer Implementierung der Auf-

gabe 1a verwenden (nur Rotation und Translation) und später die zugehörigen drei Zeilen im Konstruktor auskommentieren und stattdessen den Kommentar um die folgenden drei Zeilen entfernen, um Ihre Implementierung der Aufgaben 1b und 1c zu testen. Hierzu müssen Sie nun noch die beiden (noch leeren) Methoden `createShape_2` und `createSequence_2` wie folgt implementieren:

- In der Methode `createShape_2` soll zunächst das Objekt geladen werden, welches durch die in der Aufgabe 1b entwickelte Klasse generiert wird. Verschieben Sie dieses Objekt ggf. in eine passende Ausgangsposition für die Bewegung (z.B. an den oberen oder linken Rand) und / oder skalieren Sie es.
- Überlegen Sie sich nun, wie das Objekt bewegt werden soll und wie diese Bewegung inkrementell erstellt werden kann. Definieren Sie ein Feld mit einer Länge, die einen Teiler von 1000 darstellt und speichern Sie das der Methode als Parameter übergebene `ColoredShape` Objekt als erstes in dem Feld. Generieren Sie dann inkrementell die weiteren Feldelemente. Nutzen Sie dabei den Kopierkonstruktor der Klasse `ColoredShape` und die Methoden `rotate`, `translate`, oder `scale`. Da die Inkremente vom i. ten zum $(i+1)$. Element ggf. sehr klein sein können, ist es zumindest bei gleichmäßigen Inkrementen günstiger die affine Transformation auf das erste Element anzuwenden als auf das jeweils vorhergehende (siehe Windrad-Beispiel).

Hinweis: Da die Felder immer wieder zyklisch durchlaufen werden, müssen Sie darauf achten, dass zwischen dem letzten und dem ersten Feldelement kein sichtbarer "Sprung" auftritt.

Um Ihre Implementierung zu testen, müssen Sie nun noch den Blockkommentar um die zugehörigen Zeilen im Konstruktor entfernen. Den Code, mit dem das rotierende Windrad generiert wird, sollten Sie dagegen auskommentieren.