SoSe 2019
Prof. Dr. Margarita Esponda
Objektorientierte Programmierung
7. Übungsblatt

Ziel: Auseinandersetzung mit Vererbung, Abstrakten Klassen und Interfaces in Java.

Aufgabe (30 Punkte)

In dieser Aufgabe sollen verschiedene Klassen definiert werden, die folgende zwei Schnittstellen implementieren:

```
public interface  Shape {
      public void draw(Graphics g);
       public Point getCenter();
       public double getRadius();
       public Color  getColor();
       public void setShapesWorld( ShapesWorld theWorld );
       public void  userClicked ( double  atX, double  atY );
       public void userTyped( char key );
       public void moveTo( double x,  double y );
}

public interface  Animation {
       public static int sleep_time = 30;
       public void  play();
}
```

In der Implementierung der **Shape**-Schnittstelle wird das Aussehen von konkreten Shape-Objekten definiert, indem die draw-abstrakte Methode programmiert wird.

Die Reaktion nach einem Maus-Klick innerhalb von Shape-Objekten wird in der userClicked-Methode programmiert und die userTyped-Methode wird ausgeführt, wenn eine beliebige Taste betätigt wird.

Die Implementierung der **Animation**-Schnittstelle definiert das Verhalten oder die Veränderungen der Eigenschaften der Shape-Objekte. Diese Veränderungen finden während der Ausführung der **play**-Methode statt.

Eine genaue Beschreibung der Semantik aller einzelnen abstrakten Methoden befindet sich direkt in den Kommentaren der Schnittstellen (Interfaces), die für diese Aufgaben vorgegeben sind. Siehe die zusätzlichen Ressourcen im KVV. Lesen Sie zuerst diese Kommentare bevor Sie anfangen zu programmieren.

Eine **ShapesPanel**-Klasse für die Visualisierung der Shape-Objekte befindet sich innerhalb der ShapeFrame-Klasse, und in dieser Klasse wird die **ShapesWorld**-Schnittstelle implementiert.

Das **ShapePanel**-Objekt ruft alle 30 Millisekunden die **draw**- und die **play**-Methoden aller Shape-Objekte auf, und die Animation wird damit simuliert.

Innerhalb der Methoden der Shape-Klassen kann die Referenz des ShapesWorld-Objekts verwendet werden, um mit anderen Shape-Objekten zu interagieren. Die Referenz zum ShapesWorld-Objekt wird in der **setWorld**-Methode des Interfaces als Argument übergeben, und jedes Shape-Objekt soll diese Referenz lokal speichern. Die **setWorld**-Methode wird für jedes neue Shape-Objekt vom **ShapesPanel**-Objekt aufgerufen.

Die **ShapesPanel**-Objekte benutzen nur Konstruktoren ohne Argumente. Sie können für die Interaktion zwischen Objekten zusätzlich Konstruktoren mit Argumenten in den selbstdefinierten Klassen programmieren.

Folgende Methoden können mit einer Referenz des **ShapesWorld**-Objektes benutzt werden:

```
public interface ShapesWorld {
        public double  getMin_X();
        public double  getMin_Y();
        public double  getMax_X();
        public double  getMax_Y();
        public  Shape getClosestShape (Shape currentShape);
        public void  addShape (Shape aNewShape);
        public void  removeShape (Shape shapeToBeRemoved);
}
```

Zwei Beispielsimplementierungen der Shape- und Animation-Schnittstellen (die **Around** und **Ellipse** Klassen) werden zur Verfügung gestellt.

Das Programm muss mit der **ShapeWorld_Main**-Klasse gestartet werden, und die implementierten **Shape**-Klassennamen müssen als Argumente der **main**-Methode angegeben werden. Wenn man mit Eclipse arbeitet, müssen die Argumente in der entsprechenden Ausführungskonfiguration eingegeben werden.

Die vorgegebenen Klassen dürfen nicht verändert oder mit anderen Klassen ersetzt werden. In den Aufgaben a) bis g) müssen Sie nur neue Klassen definieren, die zusammen mit den vorgegebenen Klassen korrekt funktionieren.

a) (4 Punkte) Implementieren Sie zuerst eine einfache Klasse **GoAndBack**. Objekte dieser Klasse sollen sich einfach in einer horizontalen Linie hin und zurück bewegen, ohne das Fenster zu verlassen.

b) (5 Punkte) Programmieren Sie eine Klasse **Captive** mit einer selbst konzipierten originalen Gestalt. Die Captive-Objekte bewegen sich in alle Richtungen, ohne das Fenster zu verlassen.

c) (7 Punkte) Programmieren Sie eine **Stein** Klasse, die steinförmige Objekte produziert. **Stein**-Objekte erscheinen oben und fallen nach dem Gravitationsgesetz und zerbrechen in viele kleine Stein-Objekte, die im Boden bleiben. Eine **MiniStein** Klasse müssen Sie selber als Unterklasse der **Stein**-Klasse definieren.

d) (8 Punkte) Programmieren Sie eine **Scared** Klasse, die ängstliche Objekte produziert. Scared-Objekte fangen an zu zittern, wenn andere Objekte zu nah kommen. Wenn die Scared-Objekte von anderen Objekten berührt werden, springen sie in eine zufällig gewählte neue Position. Wenn sie beim Springen in die neue Position wieder in Berührung mit anderen Objekten kommen, platzen sie und produzieren kleine **Stein**-Objekte.

e) (6 Punkte) Programmieren Sie eine Klasse **Feuerwerk**. Die Objekte der Feuerwerk-Klasse können mit einem Mausklick angezündet werden. Dabei werden viele kleine **Bunte**-Objekte, die am Anfang in alle Richtungen geschleudert werden, produziert.

f) (6 Punkte) Programmieren Sie eine **Roboter**-Klasse, die Objekte produziert, die mit der Tastatur steuerbar sind. Dadurch kann interessantes Verhalten produziert werden. Das Drücken der **j**-Taste soll bewirken, das ein vorher gewählter Robot springt. Sie müssen hier eine eigene originale Gestalt für Ihren Roboter programmieren.

Sie dürfen zwischen **e)** und **f)** eine Aufgabe wählen. Wenn Sie beide programmieren, bekommen Sie entsprechende Bonuspunkte.

g) Sie erhalten 2 Bonuspunkte für jede zusätzliche Klasse von **Shape**-Objekten mit einem selbstdefinierten Verhalten Ihrer Wahl. Alle zusätzlichen Shape-Objekte sollen mit viel Fantasie miteinander interagieren. Bonuspunkte können erzielt werden, wenn die vorgegebenen Klassen programmiert worden sind. (Maximum: 3 zusätzliche Klassen)

Definieren Sie in allen Klassen sinnvolle Instanz-Variablen, die den Zustand Ihrer **Shape**-Objekte zwischenspeichern.

Definieren Sie zusätzliche Instanz-Methoden, um das Verhalten Ihrer Shape-Objekte möglichst gut strukturiert zu programmieren.

Versuchen Sie möglichst viele Methoden der **ShapesWorld**-Schnittstelle zu verwenden, um ein interessantes Verhalten in Ihren Klassen zu produzieren.

Wichtige Hinweise für die Java-Programmierung:

1) Verwenden Sie selbsterklärende Namen von Variablen und Methoden.
2) Für die Namen aller Bezeichner müssen Sie die Java-Konventionen verwenden.
3) Verwenden Sie vorgegebene Klassen- und Methodennamen.
4) Methoden sollten klein gehalten werden, sodass auf den ersten Blick ersichtlich ist, was diese Methode leistet.
5) Methoden sollten möglichst wenige Argumente haben.
6) Methoden sollten entweder den Zustand der Eingabeargumente ändern oder einen Rückgabewert liefern.
7) Verwenden Sie geeignete Hilfsvariablen, und definieren Sie sinnvolle Hilfsmethoden in Ihren Klassendefinitionen.
8) Zahlen sollten durch Konstanten ersetzt werden.
9) Löschen Sie alle Programmzeilen und Variablen, die nicht verwendet werden.