ALP II SS 2012
Prof. Dr. Margarita Esponda

9. Übungsblatt

Ziel: Weitere Auseinandersetzung mit grundlegenden OO-Programmierkonzepten.

1. Aufgabe (15 Punkte)

a) Was ist der genaue Unterschied zwischen Klassenvariablen, Instanzvariablen und lokalen Variablen in Java?

b) Wann ist es sinnvoll, Methoden als Klassenmethoden zu definieren?

c) Warum macht es keinen Sinn lokale Variablen als **private** zu deklarieren?

d) Können Klassenmethoden als **private** deklariert werden? Wenn ja: wann wäre das sinnvoll?

e) Was passiert, wenn in einer Klassendefinition kein Konstruktor definiert wird?

f) Können Konstruktoren vererbt werden?

g) Warum wird die main-Methode immer als **static** definiert?

h) Wo sind Instanzvariablen sichtbar, wenn diese als **protected** deklariert werden?

i) Was sind abstrakte Klassen? Wozu sind sie gut?

j) Können Variablen einer abstrakten Klasse deklariert werden?

k) Können Objekte einer abstrakten Klasse erzeugt werden?

l) Kann man Konstruktoren in abstrakten Klassen definieren?

m) Was ist ein Interface (Schnittstelle)? Wozu sind Interfaces gut?

n) Kann einer Variablen vom Typ A ein Objekt der Klasse B zugewiesen werden, wenn A eine Unterklasse von B ist? Begründen Sie Ihre Antwort.

o) Was ist der Unterschied zwischen Methoden-Überladung und Methodenüberschreibung?

2. Aufgabe (16 Punkte)

In dieser Aufgabe sollen verschiedene Shape-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();
}
```

Eine genaue Beschreibung der Semantik der einzelnen abstrakten Methoden befindet sich in den Dateien der Java-Interfaces und Java-Klassen, die für diese Aufgaben vorgegeben sind (siehe unsere Veranstaltungs-Seite).

Es soll eine bereits vorhandene *ShapesWorld*-Klasse verwendet werden, die als Behälter innerhalb eines Fensters für die Visualisierung der *Shape*-Objekte zur Verfügung gestellt wird. In dem *ShapesWorld*-Objekt können sich die *Shape*-Objekte bewegen.

Innerhalb der Implementierung 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 *setShapesWorld*-Methode des Interfaces als Argument gegeben.

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);
        public Color  getBackgroundColor ();
}
```

Eine einfache *Shape*-Implementierung wird zur Verfügung gestellt.

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

a) (2 Punkte) Zum leichten Starten implementieren Sie eine **Sun** Klasse. Objekte dieser Klasse sollen eine einfache Sonne mit zufälligen rot/gelbe-Farben darstellen, die in einer bestimmten Position erscheint und sonst nichts tut.

b) (4 Punkte) Implementieren Sie eine **Venus** Klasse, deren Objekte sichtbar werden, wenn diese sich über einem Sun-Objekt bewegen. Wenn die *ShapesWorld*-Fläche zu Ende ist, verschwinden die Objekte und sollen aber auf der linken Seite der Fläche wieder erscheinen.

c) (4 Punkte) Programmiere eine **Jumper** Klasse.

d) (6 Punkte) Programmieren Sie eine Klasse **Wrapper**. Die Objekte der **Wrapper**-Klasse sind durchsichtig und verpacken und begleiten für den Rest ihres Lebens die Objekte, die sie als erstes berühren. **Wrapper**-Objekte sollen, wenn diese noch niemand gefangen haben, die **ShapesWorld**-Fläche nicht verlassen.

e) (6 Punkte) Programmieren Sie eine **TimeWrapper** Klasse als Unterklasse der **Wrapper** Klasse, in dem eine zusätzlichen Zeiteigenschaft definiert wird. Bei den **TimeWrapper**-Objekten platzt die Verpackung nach eine Weile, die ursprünglichen Objekte werden befreit, und die **TimeWrapper**-Objekte produzieren mehrere **MiniWrapper**-Objekte. Die **MiniWraper**-Objekte werden mit der Zeit größer und ab einer bestimmten Größe verhalten sie sich wie **Wrapper**-Objekte.

f) (6 Punkte) Programmieren Sie eine **Roboter** Klasse, die Objekte produziert, die mit der Tastatur steuerbar sind. Die Steuerung wird erst dann möglich, wenn die Objekte mit der Maus zuerst ausgewählt worden sind.

Was mit der Tastatursteuerung alles gemacht werden kann, kann frei entschieden werden.

Von e) und f) muss nur eins implementiert werden. Für die Implementierung beider Aufgaben gibt es 6 Bonuspunkte.

g) (4 Punkte) Programmieren Sie eine zusätzliche Shape-Klasse mit einem Verhalten Ihrer Wahl.

Verwenden Sie dabei sinnvolle Instanzvariablen, die den Zustand Ihrer Shape-Objekte zwischenspeichern.

Definieren Sie zusätzliche Instanzmethoden, um das Verhalten Ihrer *Shape*-Objekte möglichst gut strukturiert zu implementieren.

Versuchen Sie möglichst alle Methoden der *ShapesWorld*-Klasse zu verwenden.

Die *Shape*-Objekte sollen mit viel Fantasie miteinander interagieren.

Es gibt Bonus-Punkte für besonders originelle und interessante *Shape*-Klassen.

Im Rahmen dieser Aufgabe findet ein Wettbewerb statt.

a) Für die originellsten und gut programmierten Klassen von *Shape*-Objekten gibt es einen Preis. Dabei darf der graphischen Rahmen nicht erweitert oder verbessert werden.

b) Nur offensichtliche Fehler sollen angekündigt und dann werden diese für alle Studenten korrigiert.