Prof. Dr. Stefan Brass
Dipl.-Inform. Annett Thüring
Dipl.-Inform. Steffen Schiele
Dipl.-Inform. Andreas Bienert

Martin-Luther-Universität Halle-Wittenberg
Institut für Informatik

Wintersemester 2014/15

Objektorientierte Programmierung: Hausaufgabenblatt 9

Abgabe: 15.01.2015, 12:00 | Übung: 19./20.01.2015

Hausaufgabe 9: (4 Theoriepunkte)

Passend zum Jahresende ist Ziel dieser Hausaufgabe die Simulation einer Feuerwerksrakete oder Feuerwerksbombe. Das Rahmenprogramm, das die Anzeige der einfachen Animation übernimmt, ist schon vorgegeben:

`http://www.informatik.uni-halle.de/~brass/oop14/homework/Fireworks.java`

Ihre Aufgabe ist das Herzstück des Programms, nämlich eine (oder mehrere) Klassen für Leuchtkugeln. Die wesentliche Funktion eines solchen Objektes ist, dass man zu jedem Zeitpunkt abfragen kann, ob die Leuchtkugel überhaupt sichtbar ist, und was ggf. die XY-Koordinaten sind, sowie Farbe und Größe. Es ist schon eine abstrakte Oberklasse `Star` vorgegeben, die die benötigten Methoden definiert:

```
abstract class Star {
        // Die Zeitpunkte sind in Hunderstel Sekunden.
        // Wenn 20 Bilder pro Sekunde erzeugt werden,
        // wird nur jeder 5-te Wert angefragt.

        // Ist das Objekt sichtbar zum Zeitpunkt t?
        abstract boolean visible(int t);

        // Wenn das Objekt zum Zeitpunkt t nicht sichtbar
        // ist, spielen die Ergebnisse der folgenden
        // Methoden keine Rolle.

        // x-Position (Bereich 0 .. 500) zum Zeitpunkt t:
        abstract int posX(int t);

        // y-Position (Bereich 0 .. 500) zum Zeitpunkt t:
        abstract int posY(int t);

        // Farbe des Objektes zum Zeitpunkt t:
        abstract Color color(int t);

        // Radius der Leuchtkugel zum Zeitpunkt t (Pixel):
        abstract int size(int t);
}
```


Sie sollen nun mindestens eine Subklasse `SimpleStar` von `Star` schreiben. Diese Subklasse wird auch schon im Testprogramm (Klasse `FireworksShow`) verwendet. Objekte der Klasse `SimpleStar` sollen während eines Zeitintervalls $[t_1, t_2]$ sichtbar sein, und sich in dieser Zeit gradlinig von einem Punkt (x_1, y_1) zum Punkt (x_2, y_2) bewegen. Diese sechs Werte (alle vom Typ `int`) und die Farbe (vom Typ `java.awt.Color`) sollen die Parameter des Konstruktors sein. Der Aufruf `visible(`t`)` soll `true` liefern, gdw. $t_1 \le t \le t_2$. Die X-Koordinate zum Zeitpunkt t kann man berechnen als

$$x(t) = x_1 + (x_2 - x_1) * \frac{(t - t_1)}{(t_2 - t_1)}.$$

Die Farbe des Leuchtsterns, die der Aufruf `color(`t`)` liefert, soll der Wert aus dem Konstruktor-Aufruf sein. Die Größe bzw. der Radius `size(`t`)` soll für `SimpleStar`-Objekte immer 3 sein. Überlagern Sie außerdem die Methode `toString()`, und erzeugen Sie eine Ausgabe der folgenden Art:

```
SimpleStar: [300, 400] (250, 100) -> (321, 29) Color: R=0,G=255,B=0
```

Die dreistellige Ausgabe von Koordinaten/Zeiten erreichen Sie mit `String.format(...)`. Zur Abfrage der RGB-Farbwerte (Red, Green, Blue) studieren Sie die Dokumentation

`http://docs.oracle.com/javase/7/docs/api/java/awt/Color.html`

Da die eigentliche Animation (Klasse `Fireworks`) nur ein Array von `Star`-Objekten erwartet, steht es Ihnen frei, noch weitere Unterklassen von `Star` zu definieren. Sie könnten z.B. Sterne programmieren, die durch die Schwerkraft nach unten gezogen werden statt sich gradlinig zu bewegen. Bei realen Feuerwerksbomben gibt es solche, die einen heftigen Zerleger und nur kurz brennende Sterne haben (z.B. "Päonien"). Dann ist die Grade eine gute Approximation der Flugbahn. Es gibt aber auch Bomben mit lang brennenden Sternen und etwas schwächerer Zerleger-Ladung (z.B. "Kronen"). Dann wird die Schwerkraft wichtig. Sie können auch blinkende Sterne definieren, oder Sterne mit Farbwechsel.

Sie dürfen auch das Testprogramm in der Klasse `FireworksShow` erweitern. Die Klasse `SimpleStar` müssen Sie aber mit der vorgegebenen Schnittstelle definieren. Hier ist die aktuelle Version der Klasse, die einen roten Leuchtstern als Aufstiegseffekt zeigt, der anschliessend in 8 kreisförmig verteilte grüne Leuchtsterne platzt. Die wesentliche Methode ist `stars()`, die das Array von `Star`-Objekten liefert. Außerdem gibt es eine Methode `duration()`, die die Gesamtdauer der Animation liefert. Beachten Sie, das in Java die Koordinaten von der linken oberen Ecke des Fensters aus gerechnet werden. Die Y-Koordinate wird also nach unten größer. Man hätte dies vielleicht in der Animationsklasse umdrehen können, das ist aber aus didaktischen Gründen nicht geschehen.

```
class FireworksShow {

        static Star[] stars() {
                // Parameter der Feuerwerksbombe:
                int startHeight = 0;
                int burstHeight = 350;
                int startX = 250;
                Color cometColor = Color.RED;
```

```java
        Color starColor = Color.GREEN;
        int cometSize = 4;
        int starSize = 2;
        int numStars = 8;
        int spreadSize = 100;
        int startTime = 0;
        int burstTime = 300;
        int spreadTime = 100;

        // Abgeleitete Werte:
        int startY = Fireworks.Y_SIZE - startHeight;
        int burstY = Fireworks.Y_SIZE - burstHeight;
        int burstX = startX;
        int stopTime = burstTime + spreadTime;

        Star[] s = new Star[numStars+1];

        // Roter Leuchtstern als Aufstiegseffekt:
        s[0] = new SimpleStar(startTime, burstTime,
                    startX, startY,
                    burstX, burstY,
                    cometColor);

        // Platzt kreisfoermig in 8 gruene Leuchtsterne:
        for(int i = 0; i < numStars; i++) {
            double angle = Math.toRadians(
                    360*i/(double)numStars);
            int stopX = burstX +
                (int) Math.round(
                    spreadSize *
                    Math.cos(angle));
            int stopY = burstY +
                (int) Math.round(
                    spreadSize *
                    Math.sin(angle));
            s[i+1] = new SimpleStar(
                    burstTime, stopTime,
                    burstX, burstY,
                    stopX, stopY,
                    starColor);
        }
        return s;
    }

    static int duration() {
        return 501;
    }
}
```


Es empfiehlt sich, für das Programm ein eigenes Verzeichnis anzulegen, denn die Java-Datei enthält (mit `SimpleStar`) insgesamt vier Klassen. Für jede erzeugt der Compiler eine eigene `.class`-Datei. Das Hauptprogramm steht in der Klasse `Fireworks`. Sie starten das Programm dann also mit

```
java Fireworks
```

Mit der Leertaste können Sie die Animation anhalten und fortlaufen lassen (“Pause”), mit der Enter-Taste können Sie die Animation neu starten. Viel Spass und alles Gute für das neue Jahr!

Übungsaufgabe 9A: (ohne Abgabe)

Bitte bearbeiten Sie die Übungsaufgaben auf dem Hausaufgabenblatt, aber geben Sie diese Aufgaben nicht ab. Diese Aufgaben werden in der Übung besprochen. Sie müssen Ihre Lösung eventuell in der Übung vorführen.

Was gibt das folgende Programm aus?

```
class C {
        private int a = 1;
        public static int b = 2;

        public C() { System.out.println("A"); }

        private void p() { System.out.println("B"); }
        public  void f() { System.out.println("C"); }
        public  void g() { System.out.println(a * 10 + b); }
        public  void h() { p(); }
}

class D extends C {
        private int a = 3;
        public static int b = 4;

        public D() { System.out.println("D"); }

        private void p() { System.out.println("E"); }
        public  void f() { System.out.println("F"); }
}

class Test {
        public static void main(String[] args) {
                System.out.print("a)␣"); C c = new C();
                System.out.print("b)␣"); D d = new D();

                C x = d;

                System.out.print("c)␣"); c.f();
                System.out.print("d)␣"); d.f();
                System.out.print("e)␣"); x.f();

                System.out.print("f)␣"); c.g();
                System.out.print("g)␣"); d.g();
                System.out.print("h)␣"); x.g();

                System.out.print("i)␣"); c.h();
                System.out.print("j)␣"); d.h();
                System.out.print("k)␣"); x.h();
        }
}
```


Übungsaufgabe 9B: (ohne Abgabe)

Das folgende Programm enthält mindestens 5 Fehler. Finden Sie möglichst viele davon.

```
abstract class C {
        private int n;

        C(int i) {
                this.n = i;
        }

        public abstract int m() {
                return this.n * i;
        }

        public abstract int f();

        abstract public static int s();
}

class D extends C {
        public D(int i) {
                super(i);
                System.out.println(this.n);
        }

        public int m() {
                return 5;
        }
}

class E extends D {
        public E() {
                System.out.println("Objekt␣erzeugt");
        }
}
```

Auch diese Aufgabe bitte nicht einsenden, aber bis zur Übung bearbeiten.