

## 6. Übungsblatt „Programmierung“ im Wintersemester 2016/17

Abgabe bis 28. November 2016, 07:59 Uhr über das Abgabesystem

### Aufgabe 1: Klausurzulassung

15 Punkte

Schreiben Sie eine Klasse `Uebungsblatt`, die drei Integers als Attribute hat und einen Konstruktor, dem drei Integers übergeben werden müssen. Die drei Ganzzahlen sollen die Nummer des Übungsblattes, die maximal zu erreichende Anzahl an Punkten und die erreichte Anzahl an Punkten speichern. Achten Sie dabei darauf, dass alle drei Attribute auch außerhalb der Klasse `Uebungsblatt` zugreifbar sind.

Schreiben Sie nun eine weitere Klasse `Rechner`. Diese Klasse muss eine `main`-Methode beinhalten, in der Übungsblätter erstellt und in einem Array abgelegt werden. Erzeugen Sie 13 Übungsblätter mit aufsteigender Nummer (ab 1), jeweils 60 zu erreichenden Punkte und einer Zufallszahl zwischen 0 und 60 für die erreichte Anzahl an Punkten.

Die `main`-Methode der Klasse `Rechner` soll anschließend die Gesamtanzahl an zu erreichenden Punkte und erreichter Punkte bestimmen und den prozentualen Anteil (abgerundet, ohne Nachkommastellen) der erreichten Punkte ermitteln. Dazu soll über das Array iteriert und auf die Attribute der einzelnen Übungsblätter zugegriffen werden.

Geben Sie zum Schluss aus, ob die Klausurzulassungsgrenze von 50% erreicht wurde oder nicht.

### Aufgabe 2: Polynome

20 Punkte

Ein Polynom dritten Grades hat die Gestalt  $ax^3 + bx^2 + cx + d$ . Implementieren Sie die unten beschriebenen Konstruktor und Methoden in der vorgegebenen Klasse `Polynom`, welche ein maximal kubisches Polynom repräsentiert. In der Vorgabe ist die `main`-Methode implementiert und testet ihre Implementierung. Achten Sie darauf, dass Ihre Methodennamen, -parameter und -rückgabewerte exakt der Beschreibung entsprechen. In der Vorgabe finden Sie außerdem die Methode `toString`, welche eine Instanz der Klasse `Polynom` in einen String umwandelt.

Die ganzzahligen Koeffizienten  $a$ ,  $b$ ,  $c$  und  $d$  sollen im Konstruktor übergeben werden. Implementieren Sie auch weitere Konstruktor zum Erzeugen von Polynomen mit dem Grad  $< 3$ . Darüber hinaus, soll ihre Klasse die folgenden Methoden implementieren:

- `public void add(Polynom p)`  
Addiert ein weiteres Polynom  $p$  auf die Instanz (`this`).  
Beispiel:  $(x^3 + 3x^2 + x) + (5x^2 - 2x + 3) = (x^3 + 8x^2 - x + 3)$ .
- `public void subtract(Polynom p)`  
Subtrahiert das Polynom  $p$ .
- `public int map(int x)`  
Berechnet den Wert des Polynoms an der Stelle  $x$ . Verwenden Sie dafür die Methode `Math.pow(double x, double n)`, die die  $n$ -te Potenz von  $x$  berechnet.  
Beispiel: Polynom:  $(x^3 + 8x^2 + x + 3)$ , `map(4)` =  $4^3 + 8 * 4^2 + 4 + 3 = 199$

- `public static Polynom derivation(Polynom p)`  
Berechnet die Ableitung eines Polynoms und gibt diese zurück (Rückgabewert soll eine Instanz vom Typ Polynom sein!).

### Aufgabe 3: Objektreferenzen

5 Punkte

- a) Was gibt das folgende Codefragment aus? Erklären Sie, warum die entsprechende Ausgabe erfolgt.

```
MyColor c1 = new MyColor(255, 255, 255);
MyColor c2 = new MyColor(255, 255, 255);
if (c1 == c2) {
    System.out.println("true");
} else {
    System.out.println("false");
}
```

*Hinweis: Die Klasse MyColor ist nicht in der Vorgabe enthalten.*

- b) Was gibt das folgende Codefragment aus? Erklären Sie, warum die entsprechende Ausgabe erfolgt.

```
MyPoint p1 = new MyPoint(0, 0);
MyPoint p2 = new MyPoint(0, 0);
p1 = p2;
p1.x = 250;
p2.y = 25;
p2 = p1;
System.out.println("X: " + p2.x + ", Y: " + p2.y);
```

*Hinweis: Die Klasse MyPoint ist nicht in der Vorgabe enthalten. Sie hat zwei Attribute x und y, die mit den übergebenen Werten im Konstruktor initialisiert werden.*

### Aufgabe 4: Minesweeper - Teil 2

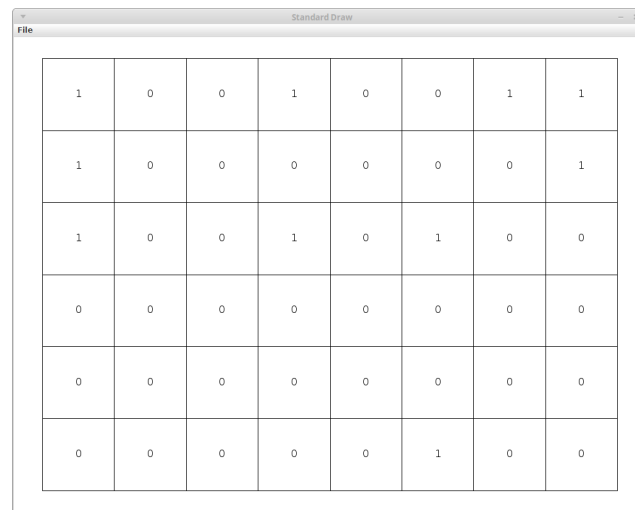
20 Punkte

- a) Erweitern Sie Ihre Lösung von der Aufgabe “Minesweeper, Teil 1” (oder die Musterlösung) so, dass das Minesweeper-Board mit Hilfe von `StdDraw` graphisch ausgegeben wird. Schreiben Sie dazu eine Methode `drawBoard`, die als Parameter das zweidimensionale Spielfeld erhält und dieses zeichnet. Ein Beispiel für eine solche Ausgabe finden Sie unten. Beachten Sie dabei, dass das Spielfeld nicht quadratisch sein muss und stellen Sie nicht-quadratisch Felder sinnvoll dar. Schreiben Sie eine eigene main-Methode, in der die Parameter Anzahl an Spalten, Anzahl an Zeilen, Anzahl an Minen und die Spalte und Zeile eines Feldes ausgelesen (per Parameterübergabe) und anschließend die Methoden `makeRandomBoard`, `drawBoard` und `isMine` ausgeführt werden. Die Methode `isMine` soll dazu mit dem als Parameter übergebenen Feld (Spalte und Zeile) ausgeführt werden. Überprüfen Sie alle übergebenen Parameter, um Abstürze Ihres Programmes vorzubeugen.
- b) Beim vorherigen Aufgabenteil (“Minesweeper, Teil 1”) haben Sie zur Positionierung von n Minen die Methode `selectRandomPosition` n-mal aufgerufen. Wenn die Methode mehrmals die gleichen Koordinaten zurückgibt, werden somit weniger Minen platziert als angegeben. Modifizieren Sie die Methode `makeRandomBoard` so, dass immer n Minen platziert werden. Überprüfen Sie auch hier die übergebene Anzahl an Minen. Diese sollte nicht größer als die Anzahl an Feldern auf dem Spielfeld sein.
- c) In der Vorlesung haben Sie gelernt, Audio-Dateien mithilfe von `StdAudio` auszugeben. Wenden Sie dieses Wissen an, um die vorgegeben Audio-Dateien abzuspielen. Dabei soll die Datei `explosion.wav` ausgegeben werden, wenn die Methode `isMine` `true` zurückgibt und die Datei `click.wav` bei `false`.

*Hinweise:*

- Achten Sie darauf, dass sich die Koordinate (0,0) der graphischen Ausgabe unten links befindet. Der Umgang mit der Spielfeld-Matrix wird erleichtert, wenn Sie die Zeilen in umgekehrter Reihenfolge speichern, sodass das zweidimensionale Array ebenfalls nach oben “wächst”.

- Die Methoden `setCanvasSize`, `setXscale` und `setYscale` können bei der Darstellung des Spielfeldes hilfreich sein.



1	0	0	1	0	0	1	1
1	0	0	0	0	0	0	1
1	0	0	1	0	1	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	1	0	0