Programmierzertifikat Objekt-Orientierung mit Java, Blatt Nr. 5
http://proglang.informatik.uni-freiburg.de/teaching/java/2009/

5.1 Aufgabe, Typen

```
interface SalesItem {}

class DeepDiscount implements SalesItem {
  int originalPrice;
  ...
}

class RegularDiscount implements SalesItem {
  int originalPrice;
  int discountPercentage;
  ...
}
```

Stellen Sie in diesem Zusammenhang die Typen der Variablen `s`, `t` und `v` fest, falls sie wie in den drei folgenden Zeilen definiert werden.

```
SalesItem s = new DeepDiscount(9900);
SalesItem t = new RegularDiscount(9900,10);
RegularDiscount u = new RegularDiscount(9900,10);
```

Falls jemand die folgenden Zeilen aufgeschrieben hat, welche von Ihnen enthalten Typfehler und wo sind diese Typefehler?

```
SalesItem v = new SalesItem();
RegualrDiscount v1 = new RegularDiscount(9900,10);
DeepDiscount v2 = new RegularDiscount(9900,10);
DeepDiscount v3 = new DeepDiscount(9900,10);
DeepDiscount v4 = new RegularDiscount(9900);
DeepDiscount v5 = new DeepDiscount(9900);
RegularDiscount v6 = new RegularDiscount(9900);
RegularDiscount v7 = new DeepDiscount(9900);
RegularDiscount v8 = new RegularDiscount(9900,10);
RegularDiscount v9 = new DeepDiscount(9900,10);
```

5.2 Aufgabe, Kuehlschrank [optional]

Sie wollen einen futuristischen Kühlschrank entwerfen. Er soll später in der Lage sein, Überblick über seinen Inhalt zu haben, alle nicht vorhandenen Lebensmittel automatisch übers Internet nachzubestellen und Ihnen auf einem Display in der Kühlschranktür Rezepte zum Kochen anzeigen. Überlegen Sie, welche Informationen der Kühlschrank über die Lebensmittel benötigt, die er aufbewahrt, und erstellen Sie ein Klassendiagramm und Code für den Kühlschrank, die Lebensmittel und die Rezepte.

5.3 Aufgabe, Laufen [optional]

Entwerfen Sie, ohne die Folien der Vorlesung zu verwenden, das Klassendiagramm für die Laufdatenbank.

5.4 Aufgabe, Raumplan [optional]

Sie sollen ein Raumplanungsprogramm entwerfen, mit dem es möglich ist, Veranstaltungen an der Uni Räumen zuzuweisen. Überlegen Sie, welche Eigenschaften ein solches System benötigt, um gut bedienbar zu ein. Einige der Eigenschaften sollten sein:

1. Die Uni besteht aus einer Reihe Fakultäten. Da Sie Ihr Programm nicht für jede Uni neu programmieren wollen, gehen Sie davon aus, dass die Anzahl der Fakultäten beliebig ist.

2. Jede Fakultät besitzt eine Reihe von Gebäuden, in denen die Veranstaltungen stattfinden können.
3. Jedes Gebäude besitzt eine Anzahl von Räumen, die zur Verfügung stehen.
4. Jeder Raum hat eine Ausstattung, wie z.B. Platzanzahl, Beamer, Micro, usw.
5. Da Sie nicht jedes Semester der Universität einen teuren Fachman schicken wollen, der ein neues Semester anlegt, muss die Anzahl der Semester ebenfalls beliebig sein.
6. Gleiches gilt für die vorhandenen Lehrveranstaltungen.
7. Die Belegungsslots bestehen aus einem Wochentag, einer Startzeit, einer Endzeit und einem Semester.

6.1 Aufgabe, Bild - Version 3

Erweitern Sie die image-Klasse aus Aufgabe 3.2 um die folgenden Methoden:

- isPortrait stellt fest, ob die Höhe des Bildes größer ist als die Breite
- size berechnet die Anzahl der Pixel des Bildes.
- isLarger gibt an, ob das Bild größer ist als das andere Bild.

Zeichnen Sie als erstes das entsprechende Klassendiagram. Füllen Sie dann den Code mit den Methodensignaturen aus. Schreiben Sie dann zwei Testfälle je Methode und füllen Sie erst anschließend den Rumpf der Methode mit dem passenden **return**-Statement.

6.2 Aufgabe, Bibliothek - Version 3

Erweitern Sie das Diagramm aus Aufgabe 2.2 und den geschriebenen Code um die folgenden Methoden:

1. thisYear. Diese Methode soll anzeigen, ob das Buch in dem übergebenen Jahr publiziert wurde.
2. thisAuthor. Diese Methode soll anzeigen, ob das Buch von dem übergebene Autor geschreiben wurde.
3. sameAuthor Diese Methode soll angeben, ob das Buch von demselben Autor geschrieben wurde wie ein als Parameter übergebenes Buch.

Zeichen Sie zuerst das Klassendiagramm, schreiben Sie dann die Methodensignaturen auf, erstellen Sie dann zwei Testfälle und füllen Sie den Rumpf als letztes mit den passenden **return**-Statements.

6.3 Aufgabe, Datum

Schreiben Sie für die date-Klasse (siehe vorherigen Blättern) zwei Methoden, die feststellen, welches von zwei Datumswerten kleiner ist.

1. earlierWithoutIf. Verwenden Sie keine **if**-Bedingungen, sondern errechnen Sie das Ergebnis.
2. earlierWithIf. Schreiben Sie dieselbe Methode diesmal mithilfe von **if**-Bedingungen, und dafür ohne Rechnungen.

Schreiben Sie zwei Testfälle für die beiden Methoden.

7.1 Aufgabe, Niederschlag

```
//the daily percipitation of three consecutive days
class Precipitation {
  int day1;
  int day2;
  int day3;

  Precipitation(int day1, int day2, int day3) {
    this.day1 = day1;
    this.day2 = day2;
    this.day3 = day3;
  }

  // how much did it rain during these three days?
  int cumulative() {
    return this.day1 + this.day2 + this.day3;
  }
}

// collect examples of precipitations
class PrecipitationExamples {
  p1 = new Precipitation(1,1,1);
  p2 = new Precipitation(2,1,2);
  p3 = new Precipitation(3,2,4);

  boolean testOne = check this.p1.cumulative() expect 3;
  boolean testTwo = check this.p2.cumulative() expect 5;
  boolean testThree = check this.p3.cumulative() expect 9;
}
```

Fügen Sie der Klasse eine Methode average hinzu. Folgen Sie hier den Entwurfkonzepten der Vorlesung und verwenden Sie vorhandene Methoden, falls dies sinnvoll ist. Fügen Sie ebenfalls zwei Tests zu der Beispielklasse hinzu.

7.2 Aufgabe, Wetter - Version 3

Erweitern Sie die Klasse aus Aufgabe 4.2 um eine Eigenschaft precipitation. Dieses Attribut soll die Regenmenge aufnehmen und Kommazahlen speichern.

Erweitern Sie anschließend die Klasse WeatherRecord um die folgenden Methoden:

1. tempDiff. Diese Methode soll die Temperaturschwankung des Tages berechnen.
2. withinRange. Diese Methode stellt fest, ob die heutigen Messwerte innerhalb der normalen Schwankungen liegen.
3. rainyDay. Diese Methode stellt fest, ob die Regenmenge größer als ein übergebener Wert ist.
4. recordDay. Diese Methode stellt fest, ob die Aufzeichnung den Temperaturrekord (noch oben oder unten) verursacht hat.

Schreiben Sie für die WeatherRecord Klasse drei Testfälle.

7.3 Aufgabe, Sterne

Kennen Sie das Märchen *Die Sterntaler* der Gebrüder Grimm? Aufbauend auf diesem Märchen sollen Sie ein Spiel entwickeln, indem Sie die Bewegung vom Himmel fallender Sterne simulieren.

```
//represent a falling star on a 100x100 canvas
class Star {
  int X = 20;
  int y;
  int DELTA = 5;

  Star(int y) {
    this.y = y;
  }
  ... // bitte ergänzen
}
```

Schreiben Sie passend zu der Klasse eine Methode drop, die Ihren Stern jeweils um DELTA Pixel nach unten bewegt, bis er den Boden erreicht hat. Die Methode drop soll hierbei jeweils einen neuen Stern zurückgeben, und den alten Stern unverändert lassen.

Schreiben Sie mindestens 4 Testfälle, um zu prüfen, ob Ihre Sterne sich korrekt verhalten.