Albert-Ludwigs-Universität Freiburg
Informatik 1
Wintersemester 2011/12

Prof. Dr. Wolfram Burgard
Dipl. Ing. Jörg Röwekämper
Dipl. Inf. Markus Kuderer

Übungsblatt 5

Abgabe bis Donnerstag, 1.12.11, 12:00 Uhr

Hinweis: Kompilieren und Testen Sie Ihre Programme mit Hilfe des `ant`-Buildsystemes. Schicken Sie Ihrem Tutor eine Kopie des kompletten Projektordners, der alle benötigten Dateien enthält.

Aufgabe 5.1

1. Laden Sie das Beispielprojekt `MyProject`[1] von der Vorlesungsseite herunter. Kompilieren Sie die in `MyProject` enthaltenen Beispielprogramme mit Hilfe des `ant`-Buildsystemes (Siehe Foliensatz Tools).
2. Erstellen Sie Ihr eigenes Projekt `Info1Exercises`. Übernehmen Sie dafür die Struktur des Beispielprojektes.
3. Fügen Sie die Klasse `Triangle` des letzten Übungsblattes in das `src`-Verzeichnis Ihres Projektes ein und kompilieren Sie Ihr Projekt. Betrachten Sie die Ausgabe des `ant`-Buildsystemes und überprüfen Sie, ob alle Codekonventionen eingehalten wurden.
4. Schreiben Sie eine Klasse `TriangleTest`, die für die Methoden
   - `computeCircumference()`
   - `computeArea()`
   - `isIsosceles()`
   - `isEquilateral()`

   `JUnit`-Tests enthält.

Aufgabe 5.2

1. Auf der Vorlesungshomepage finden Sie eine Datei `data.dat`, die eine Liste von Zahlen enthält. Lesen Sie alle Zahlen aus der Datei ein und speichern Sie diese in einem `Vector<Integer>`-Objekt.
2. Geben Sie alle Zahlen aus, die **genau** einmal in der Liste vorkommen.
   Hinweis: Verwenden Sie zwei geschachtelte Schleifen, um diese Zahlen zu bestimmen.

[1] http://ais/teaching/ws11/info/material/MyProject.zip

Aufgabe 5.3

Betrachten Sie den Auszug der Klasse `GenericTuple`, die ein Tupel aus drei Objekten darstellt.

```
public class GenericTuple<A,B,C> {
  public void setA( ... ) { ... }
  public void setB( ... ) { ... }
  public void setC( ... ) { ... }

  public ... getA(){ ... }
  public ... getB(){ ... }
  public ... getC(){ ... }

  public String toString() {
    ...
  }

  A a;
  B b;
  C c;

}
```

1. Vervollständigen Sie die `set` - und `get`-Methoden der Klasse `GenericTuple`.
2. Vervollständigen Sie die `toString` Methoden der Klasse `GenericTuple`, so dass sie einen String der Form "$(\cdots,\cdots,\cdots)$" ausgibt.
   Hinweis: Verwenden Sie die `toString`-Methoden der drei Member-Variablen.
3. Implementieren Sie eine Klasse `GenericTupleVector<A,B,C>`, die einen Vektor aus `GenericTuple<A,B,C>`-Elementen enthält.
4. Implementieren Sie für die Klasse `GenericTupleVector<A,B,C>` eine Methode `addTuple(GenericTuple<A,B,C> tuple)`, die ein Tupel in den Vektor einfügt.
5. Implementieren Sie für die Klasse `GenericTupleVector<A,B,C>` eine Methode `GenericTupleVector<A,B,C> compareB(B b)`, die einen Vektor zurückgibt, der alle Tupel `t` enthält, für die gilt `b.equals(t.getB())`
6. Implementieren Sie für die Klasse `GenericTupleVector<A,B,C>` eine Methode `int getSize()`, die die Anzahl der Elemente im Vektor zurückgibt.
7. Auf der Vorlesungshomepage finden Sie eine `JUnit`−Klasse `GenericTupleVectorTest.java`. Nutzen Sie diese, um ihre Klassen zu testen.