Programmieren in Java

http://proglang.informatik.uni-freiburg.de/teaching/java/2015/

Java-Übung Blatt 1 (Erste Schritte)

2015-04-23

Hinweise

- Ändern Sie nicht die Schreibweise von Bezeichnen, die auf dem Übungsblatt vorgegeben sind. Dies betrifft sämtliche global sichtbaren Namen von Eclipse-Projekten, Paketen, Klassen, etc.
- Bezeichner und Kommentare bitte auf *Englisch*!
- Schreiben Sie *sinnvolle* Kommentare.
- Laden Sie Ihre Lösungen mit Subversion (SVN) ins Übungssystem hoch. Den entsprechenden Pfad finden Sie online.
- Sollte das Übungssystem Ihre Einreichung nicht übersetzen können, dann wird sie nicht korrigiert und Sie erhalten keine Punkte.
- Die Korrektur Ihrer Abgabe wird unter dem Namen `Feedback-<login>-ex<XX>.txt` eingecheckt. (`<login>` ist dabei Ihr *myAccount*-Name und `<XX>` die Kennziffern des Übungsblatts)
- Sie können Ihre Gesamtpunktzahl im Übungsportal einsehen.
- Dokumentieren Sie Ihren Zeitaufwand, den gefühlten Schwierigkeitsgrad, sowie Probleme beim Lösen der Aufgabe oder auch Anregungen und Vorschläge zur Verbesserung in der Datei `<Projektverzeichnis>/erfahrungen.txt`

Exercise 1 (Primzahlen, 4 Punkte)

Projektverzeichnis: `ex01_1`. Package: `prime` .

Erstellen Sie eine Klasse `PrimePrinter` im Package `prime` nach dem Gerüst in Figure 1

```
1  package prime;
2
3  public class PrimePrinter {
4
5      public static void main(String[] args) {
6          //Constants
7          final int maxNumber = 100;
8          final int maxPrimesPerLine = 10;
9          //TODO: put your implementation here
10     }
11 }
```

Abbildung 1: Gerüst für den PrimePrinter

Implementieren Sie dort eine Main-Methode, die alle Primzahlen, die kleiner als der Wert der Variablen `maxNumber` sind, auf die Konsole schreibt.

Beachten Sie weiter:

- Ihre Implementierung muss natürlich für jeden anderen (positiven) Wert von `maxNumber` funktionieren.
- Aus Gründen der Lesbarkeit müssen die ermittelten Primzahlen in Gruppen von `maxPrimesPerLine` Primzahlen pro Zeile ausgegeben werden. Ausnahme ist natürlich die letzte Zeile, die nur so viele Primzahlen enthält, wie noch übrig sind. Auch hier gilt, dass Ihre Implementierung natürlich auch für jeden anderen (positiven) Wert von `maxPrimesPerLine` funktionieren muss.

Tipp: Sie brauchen lokale `int`-Variablen, `for`-Schleifen, `if`-Statements.

Exercise 2 (Primitive Klassen: Formen, 6 Punkte)

Projektverzeichnis: `ex01_2`. Package: `geometry`.

In dieser Aufgabe modellieren wir eine Ellipse aufgrund ihrer mathematischen Definition. Der Mittelpunkt der Figur kann durch eine x- und y-Koordinate lokalisiert werden und enthält außerdem zwei Radien. Gestalten Sie die Klasse so, dass ihre Instanzen nach Erstellung nicht mehr verändert werden können (also „immutable“ sind). Verwenden Sie als Instanzvariablen Fließkommazahlen (double). Ellipsen sollen folgende Operationen unterstützen:

- Eine Methode *offset(x, y)*, welche die Koordinaten zur Lokalisierung der Figur verändert. Der Rückgabewert ist ein neues Objekt der Klasse mit den veränderten Koordinaten.
- Eine Methode *contains(x, y)*, die überprüft, ob sich eine bestimmte Koordinate innerhalb der Ellipse befindet. Die Methode soll als Rückgabewert einen boolean-Wert zurückgeben. Alle Koordinaten, die auf oder innerhalb der Kurve liegen, sind „innerhalb“.

1. Erstellen Sie zunächst ein Klassendiagramm entsprechend den obigen Angaben. Verwenden Sie die UML-Konventionen aus der Vorlesung. Sie können ein Tool Ihrer Wahl[1] benutzen oder eine ordentliche, handgeschriebene Zeichnung einscannen. Speichern Sie Ihre Lösung bitte als pdf, txt, jpg oder png ab (in das Projektverzeichnis neben dem src-Ordner) und übertragen Sie später die Datei zusammen mit dem Code.
2. Schreiben Sie anhand Ihres Diagramms ein Klassenskelett für die geometrischen Figur. Achten Sie, wie auch bei allen anderen Aufgaben, auf eine vollständige und nachvollziehbare Kommentierung.
3. Erstellen Sie, wie aus der Vorlesung bzw. von Blatt 0 bekannt, einen neuen *JUnit Test Case* für Ihre Klasse.

   Tipps:

   - Testen Sie `contains`, indem Sie bei einigen beispielhaften Instanzen prüfen, ob verschiedene Koordinaten darin liegen (oder eben nicht).
   - Testen Sie `offset`, indem Sie sie auf einige Instanzen anwenden und mittels `contains` verifizieren, dass alles wie erwartet verlaufen ist.
4. Implementieren Sie nun die Funktionalität für Ihre Klassen. Für die Methode `contains` ist es hilfreich, die Ellipsengleichung (kartesische Koordinaten) zu betrachten (siehe: `http://de.wikipedia.org/wiki/Ellipse#Ellipsengleichung_.28kartesische_Koordinaten.29`).

[1] Für dieses kleine UML-Diagramm können Sie auch ASCII-Art (siehe `http://c2.com/cgi/wiki?UmlAsciiArt`) verwenden.