Albert-Ludwigs-Universität Freiburg
Institut für Informatik
Einführung in die Informatik
Sommersemester 2011

Prof. Dr. Wolfram Burgard
PD Dr. Cyrill Stachniss
Dipl. Inf. Axel Rottmann
Dipl. Inf. Markus Kuderer

Übungsblatt 11

Abgabe bis Donnerstag, 28.07.2011, 12:00 Uhr

Aufgabe 11.1

Das Newton-Verfahren ist ein Standardverfahren der Mathematik zur numerischen Lösung von Gleichungen. Beispielsweise kann man für stetig differenzierbare Funktionen $f : \mathbb{R} \to \mathbb{R}$ Näherungen einer Nullstelle finden. Das Newton-Verfahren ist durch folgende rekursive Gleichung gegeben

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)} \, .$$

1. Erweitern Sie die Klasse `Polynom` von Übungsblatt 10 um dieses Verfahren zur Nullstellenbestimmung. Sie finden die Datei `Polynom.java` auf der Vorlesungsseite.

   Hinweis: Das Newton-Verfahren ist ein so genanntes lokal konvergentes Verfahren, d. h. die erzeugte Folge konvergiert nicht für beliebige Startwerte. In der Regel sollte ein Startwert nahe der gesuchten Nullstelle gewählt werden. Was muss daher bei der Implementierung beachtet werden?

2. Kompilieren Sie die Klasse mittels `ant` und implementieren Sie noch mindestens zwei weitere Unittests mittels `JUint`, um Ihre `Polynom`-Klasse zu testen. Ein Unittest, der den Konstruktor testet, könnte wie folgt aussehen.

```
import org.junit.Test;
import org.junit.Assert;

/**
 * Test class for the Polynom class.
 *
 * @author Axel Rottmann
 *
 */
public class PolynomTest {

    /**
     * Test the constructor.
     */
    @Test public void testPolynom() {
        Polynom p = new Polynom(2);
        Assert.assertEquals("0.0 + 0.0x^1 + 0.0x^2", p.toString());
    }
   ...
}
```

Aufgabe 11.2

Betrachten Sie den folgenden Auszug der Klasse `Student`, die eine Studentin bzw. einen Studenten repräsentiert. Sie finden die Datei `Student.java` auf der Homepage.

```
abstract class Student {
        public Student(String fn, String sn, String sjt) {
                this.first_name = fn;
                this.surname = sn;
                this.subject = sjt;
        }

        public String toString() {
                String s = first_name + " " + surname + " studiert " + subject;
                if (hasPassed()) {
                        s += "\nAlle Prüfungen bestanden.\n";
                } else {
                        s += "\nNoch nicht alle Prüfungen bestanden.\n";
                }
                return s;
        }

        abstract boolean hasPassed();

        private String first_name;
        private String surname;
        private String subject;
}
```

Da es verschiedene Voraussetzungen zum Bestehen unterschiedlicher Studiengänge gibt, wurde die Methode `hasPassed` als `abstract` spezifiziert. Zum Beispiel sei die Voraussetzung zum Bestehen des Studienganges `Computer Science` die erfolgreiche Teilnahme an den Prüfungen `Computer Science 1` sowie `Algorithms`. Implementieren Sie dafür eine Klasse `ComputerScientist`, die von der Klasse `Student` abgeleitet ist. Verwenden Sie `ant`, um die Klasse zu kompilieren.

1. Der Konstruktor der Klasse `ComputerScientist` soll die Member-Variablen `first_name, surname` und `subject` entsprechend setzen. Weshalb können diese trotzdem als `private` deklariert werden?

2. Implementieren Sie `boolean` Member-Variablen, die den Prüfungen entsprechen, sowie Methoden, mit denen diese Variablen einzeln gesetzt werden kpnnen. Implementieren Sie zusätzlich die Methode `hasPassed()`, die genau dann `true` zurückgeben soll, wenn die Voraussetzungen zum Bestehen des Studiengangs erfüllt wurden. Was passiert, wernn Sie die Methode `hasPassed()` nicht implementieren?

3. Überschreiben Sie die Methode `toString()`. Der zurückgegebene `String` soll zusätzlich zu den bereits implementierten Ausgaben eine Liste aller bestandenen Prüfungen des Studenten enthalten.

4. Implementieren Sie Unittests mittels `JUint`, um Ihre `ComputerScientist`-Klasse zu testen.

Aufgabe 11.3

Betrachten Sie ein Schachbrett und einen einzelnen Springer. Schreiben Sie eine Klasse, die ein Schachbrett beliebiger Größe repräsentiert und alle möglichen Spielzüge eines Springers berechnen kann. Ein Springer bewegt sich immer um zwei Felder horizontal und eines vertikal bzw. ein Feld horizontal und zwei vertikal. Mit Ihrer Klasse soll bestimmt werden können, welche Felder ein Springer ausgehend von Startposition (x, y) in maximal n Zügen erreichen kann.

1. Führen Sie eine Aufwandsabschätzung Ihres Programms durch und geben Sie den Aufwand in der O-Notation an. Was ist eine geeignete Problemgröße n?

2. Welche Felder können auf einem normalen Schachbrett (8x8 Felder) ausgehend von Feld $(0, 0)$ in maximal 4 Zügen erreicht werden?

3. Wieviele Züge sind mindestens notwendig, um von Startposition $(0, 0)$ ein beliebiges Feld auf einem Brett der Größe 18x18 zu besuchen?

Hinweis: Verwenden Sie zur Darstellung des Schachbretts ein zweidimensionales Array.