Albert-Ludwigs-Universität Freiburg
Institut für Informatik
Einführung in die Informatik
Sommersemester 2012

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

Zusatzblatt

Abgabe in Absprache mit Tutor

Hinweis:

Aufgaben immer per E-Mail (eine E-Mail pro Blatt und Gruppe) an den zuständigen Tutor schicken (Bei Programmieraufgaben Java Quellcode und eventuell benötigte Datendateien).

Aufgabe 1

Betrachten Sie folgende Java-Methode, die ein Integer-Array als Parameter erwartet. Gehen Sie davon aus, dass die Methode `otherMethod(int[] a)` keine Ausgaben verursacht und das übergebene Array `a` der Länge n mit Laufzeit $O(n)$ bearbeitet.

```
public void method(int[] a){
  int n = a.length;
  int count = 0;
  for (int i = 0; i < n; i++) {
    for (int j = i+1; j <  n; j++) {
      if (a[i] == a[j]) {
        otherMethod(a);
        count++;
      }
      System.out.println(count);
    }
  }
}
```

1. Was gibt die Methode `method` für die Eingabe `[0,1,1,0,2]` aus?
2. Führen Sie eine Aufwandsabschätzung in best- und worst case für die Methode `method` in Abhängigkeit von der Länge `n` des Arrays `a` durch.

Aufgabe 2

1. Schreiben Sie eine rekursive Java-Methode `f(int x, int y)`, die folgende Funktion implementiert ($\mathbb{N}$ sind die natürlichen Zahlen, $\mathbb{N}_0$ sind die natürlichen Zahlen inklusive 0): $f : \mathbb{N}_0 \times \mathbb{N} \to \mathbb{N}$:

$$f(x,y) = \begin{cases} 1 & \text{falls } x = 0 \\ y & \text{falls } x = 1 \\ f(x-1, y \cdot y) & \text{falls } x > 1 \end{cases}$$

2. Berechnen Sie $f(3,2)$. Geben Sie dabei alle Zwischenschritte an.

3. Zeichnen Sie für Ihre Methode die Activation Records für den Aufruf von `f(3,2)` zu dem Zeitpunkt, an dem die maximale Rekursionstiefe des Aufrufs erreicht ist.

Aufgabe 3

Die Zahl π kann durch die so genannte Gregory-Leibniz-Reihe

$$\pi = \sum_{n=0}^{\infty} (-1)^n \frac{4}{1+2n}$$

angenähert werden. Eine obere Schranke für den Fehler der Approximation nach n Summanden ist gegeben durch:

$$|R(n)| \leq \frac{4}{2n+1}$$

1. Implementieren Sie eine Methode, die die Zahl π approximiert und die Approximation zurückliefert. Die gewünschte Genauigkeit (obere Schranke) soll als Parameter an die Funktion übergeben werden.

2. Schreiben Sie für Ihre Implementierung eine Testmethode mittels `JUnit`, die testet, ob die Zahl π bis auf die gewünschte Genauigkeit approximiert wird. Überprüfen Sie dafür die Genauigkeit der Approximation für $|R(n)| \leq 0.0001$ und $|R(n)| \leq 0.0000001$. Hinweis: Verwenden Sie `Math.PI` als Referenz für π.

Aufgabe 4

Gegeben seien die folgenden Ausschnitte der Klassen `Rectangle` and `Circle` und der abstrakten Klasse `MovableGeometricObject`, die die Gemeinsamkeiten der Klassen `Rectangle` and `Circle` beschreibt.

```
public class MovableGeometricObject {

  MovableGeometricObject(int centerX, int centerY) { ... }

  public void move(int deltaX, int deltaY) { ... }
  public int centerX() { ... }
  public int centerY() { ... }

  abstract double area();

  private int centerX;
  private int centerY;
}

public class Circle ... {

  Circle(...) { ... }
  ...
  private int radius;
}

public class Rectangle ... {
  Rectangle(...) { ... }
  ...
  private int sideX;
  private int sideY;
}
```

Dabei repräsentieren die Instanzvariablen `centerX` und `centerY` die x- und y-Koordinaten der Mittelpunkte des MovableGeometricObject. Die Methode `void move(int deltaX, int deltaY)` soll den Mittelpunkt des Objektes verschieben und `double area()` gibt den Flächeninhalt zurück.

1. Modifizieren Sie die oben angegebenen Klassen `Circle` und `Rectangle` so, dass sie von der abstrakten Klasse `MovableGeometricObject` erben.

2. Vervollständigen Sie die Konstruktoren und Methoden der Klassen.

3. Schreiben Sie `JUnit`-Tests für die Klassen `Circle` und `Rectangle`. Testen Sie dafür die Methode `move(int deltaX, int deltaY)` sowie die Methode `area()` der Klassen.

Aufgabe 5

Betrachten Sie die folgende Klassenhierarchie. Geben Sie an, was in der `main`-Methode ausgegeben wird.

```
class A {
  public A () { }
  void print() {
    System.out.println("A");
  }
}

class B extends A {
  public B () { }
  void print() {
    System.out.println("B");
  }
}

class C extends A {
  public C () { }
}

class D extends B {
  public D () { }
}

class Hierarchy {
  public static void main (String args[]) {
    A a = new C();
    A b = new B();
    B c = new B();
    A d = new D();
    D e = new D();

    a.print();
    b.print();
    c.print();
    d.print();
    e.print();
  }
}
```