Hochschule Ingolstadt WS 2008/09
Studiengang Elektro- und Informationstechnik
Studiengang Informatik

Grundlagen der Programmierung 2
Praktische Informatik 2 / Softwareentwicklung 2

Aufgabensteller: Prof. Glavina, Prof Grauschopf, Prof. Hahndel, Prof. Schmidt
Prüfungsdauer: 90 Minuten
Hilfsmittel: keine

Aufgabe 1 (Verständnis, Fragen zu Java, ca. 24%):

Gegeben Sei der folgende (unvollständige) Sourcecode in Java:

```
class ArrayN {
     private int einFeld[];

     // a) Hier fehlt noch Ihr Konstruktor ....

     public static void main(String[] args) {
          ArrayN bisZehn, bisZwanzig, bereich;

          // Zwei Objekte  erzeugen:
          bisZehn    = new ArrayN(10);           // (***)
          bisZwanzig = new ArrayN(20);           // (***)

          // b) Ausgabe aller Einzelwerte beider Arrays:

          // c) Der folgende Aufruf erzeugt eine
          // NegativeArraySizeException, die abzufangen ist:
          bereich = new ArrayN(-10);              //(+++)

          System.out.println("Programmende");          //(last)
          }
}
```

a) Schreiben Sie einen Konstruktor, der das Attribut `einFeld` mit einer Reihung (Array) der Länge (als Argument übergebenen Länge) n initialisiert. Diese Reihung sollte die Zahlen von 1 bis n in geordneter Reihenfolge enthalten, d.h. die beiden Zeilen (***) sollten funktionieren!

b) Geben Sie gültigen Javacode an, um die Einzelwerte der Reihungen `bisZehn` und `bisZwanzig` an der oben gekennzeichneten Stelle auszugeben.

c) Die mit (+++) markierte Zeile erzeugt eine `NegativeArraySizeException`, wodurch die Programmausführung an dieser Stelle abbricht. Ergänzen Sie den gezeigten Javacode um eine geeignete Ausnahmebehandlung, die eine Fehlermeldung auf der Konsole ausgibt und mit der Programmausführung regulär fortfährt, so dass auch noch die letzte mit (last) markierte Zeile ausgeführt wird.

d) Gegeben seien die folgenden beiden Java-Klassen:

```
class Quadrat {
     double x;

     Quadrat (double y){ x = y; }
     public String toString(){
          return("Quadrat mit Seitenlänge " + x);
     }
}

class Anwendung {
     private int x = 1;

     Anwendung(int q) { x = q; }

     int getX() { return x; }
     void setX(int x) { this.x = x; }

     Quadrat eineMethode(Quadrat q, int k){
          q.x = k;
          k = 10;
          System.out.println(k);
          return new Quadrat(q.x);
     }

     public static void main(String[] args) {
          int x = 2;
          Anwendung u = new Anwendung(5);
          Quadrat r = new Quadrat(2);
          Quadrat s = r, t;

          u.setX(3);
          t = u.eineMethode(r, 5);
          System.out.println(x);
          System.out.println(r);
          System.out.println(s);
          System.out.println(t);
          r.x = 10;
          s.x = 11;
          t.x = 12;
          System.out.println(r);
          System.out.println(s);
          System.out.println(t);
          System.out.println(u.x);
          }
}
```

d) Geben Sie die vollständige Ausgabe des Programmes an.

e) Wie viele Objekte vom Typ Quadrat gibt es während der Laufzeit (mit Begründung)?

f) Kann das Attribut x in der Quadrat-Klasse problemlos auf private gesetzt werden ? (mit Begründung)

Aufgabe 2 (Modellierung mit Java-Klassen, 21%):

Vorgegebene Klassengerüste nicht abschreiben!

a) Eine Ampel schaltet zyklisch

Rot → Rot-Gelb → Grün → Gelb → Rot → ...

Gegeben ist folgendes Klassengerüst:

```
class Ampel {
     public final static int ROT = 0;
     public final static int ROTGELB = 1;
     public final static int GRÜN = 2;
     public final static int GELB = 3;

// ... Ihre Attribute und Methoden ...

}
```

- Fügen Sie ein Attribut hinzu, das den aktuellen Schaltzustand (Rot, Rot-Gelb, Grün oder Gelb) eines Ampel-Objekts speichert.
- Fügen Sie einen Konstruktor hinzu, der den Schaltzustand auf einen als Parameter übergebenen Wert setzt.
- Schreiben Sie eine Methode `weiterschalten`, die den Schaltzustand auf den in den im Zyklus folgenden weitersetzt.

b) Eine Kreuzung hat entweder vier Ampeln, in jeder Himmelsrichtung eine, oder gar keine.

Sie brauchen in dieser Aufgabe das (vorgegebene) Random-Objekt, dessen Methode `nextDouble()` eine Zufallszahl zwischen 0.0 und 1.0 zurückgibt.

Gegeben ist das folgende Klassengerüst:

```
import java.util.Random;

class Kreuzung {
     private   final static Random rand = new Random();

     // ... Ihre Attribute und Methoden ...
}
```

- Fügen Sie Attribute für die vier Ampeln hinzu.
- Fügen Sie einen Konstruktor hinzu, der mit 20% Wahrscheinlichkeit die vier Ampel-Referenzen uninitialisiert belässt. In 40% der Hälfte der Fälle sollen die Ampeln so initialisiert werden, dass die Nord-Süd-Richtung auf grün geschaltet, und die Ost-West-Richtung auf rot geschaltet ist. In den verbleibenden Fällen genau andersherum.
- Fügen Sie eine Methode `weiterschalten` hinzu, die die vier Ampeln weiterschaltet, sofern vorhanden.

c) Gegeben ist das folgende Klassengerüst:

```
class Manhattan {
     public final static int M = 126;
     public final static int N = 12;

     // ... Ihre Attribute und Methoden ...

}
```

- Fügen Sie als Attribut ein zweidimensionales M x N-Array von Kreuzungen hinzu.
- Fügen Sie einen Konstruktor hinzu, der die Einträge des Arrays mit gültigen Kreuzung-Objekten belegt.
- Fügen Sie eine Methode `weiterschalten` hinzu, die for-Schleifen neuen Typs nutzt, um alle Kreuzungen `weiterzuschalten`. Konventionelle for-Schleifen geben einen Punktabzug.

Aufgabe 3 (Klassen, Polymorphie, 30%):

Eine Gemeinde erfasst ihre Einwohner in Haushalten und unterscheidet dabei zwischen Erwachsenen und Kindern:

```
class Person {
    String name;
    int alter;
}

abstract class Haushalt {
    private Person[] erwachsene, kinder;

    Haushalt(Person[] erwachsene) {
        this.erwachsene = erwachsene;
    }

    Haushalt(Person[] erwachsene, Person[] kinder) {
        this(erwachsene);
        this.kinder = kinder;
    }

    Person[] getErwachsene() {return erwachsene;}
    Person[] getKinder() {return kinder;}
}
```

Ferner wird unterschieden zwischen Single-Haushalten, kinderlosen Paaren, Alleinerziehenden (Teilfamilien) und Familien:

```
class Single extends Haushalt {
    Single(Person p) {super(new Person[] {p});}
}

class Paar extends Haushalt {
    Paar(Person p1, Person p2) {super(new Person[] {p1, p2});}
}

class Familie extends Haushalt {
    Familie(Person p, Person[] kinder) {
        super(new Person[] {p}, kinder); // alleinerziehend
    }

    Familie(Person p1, Person p2, Person[] kinder) {
        super(new Person[] {p1, p2}, kinder);
    }
}
```

Die Gemeindeverwaltung fasst alle Haushalte in einer Liste zusammen:

```
import java.util.*;
class Gemeinde extends LinkedList<Haushalt> {}
```

a) Fügen Sie der Klasse `Gemeinde` eine Methode `kinderProHaushalt` hinzu, welche die durchschnittliche Anzahl der Kinder pro Haushalt ermittelt und als `double` zurückgibt.

b) Fügen Sie der Klasse `Gemeinde` eine Methode `alleinerziehende` hinzu, welche alle Familien mit nur einem Erwachsenen und Kindern ermittelt und als `LinkedList<Haushalt>` zurückgibt.

c) Fügen Sie der Klasse `Gemeinde` eine Methode `steuersatzProHaushalt` hinzu, welche den durchschnittlichen Steuersatz pro Haushalt ermittelt und als `int` zwischen 0 und 100 korrekt gerundet zurückgibt (entsprechend einem Steuersatz zwischen 0 und 100%). Ergänzen Sie dazu die bisherigen Klassen durch Attribute und Methoden zur Implementierung folgender Steuersätze:

Der Basissteuersatz beträgt 40%. Für die verschiedenen Haushalte gelten folgende Ermäßigungen:

- Single: keine
- Paar: 5%

Für Familien gilt der Basissteuersatz, der sich für jedes Kind um 10% ermäßigt, jedoch insgesamt nicht unter 0%.

Aufgabe 4 (GUI, 25%):

Es soll -- mit AWT oder Swing -- eine Oberfläche programmiert werden, die einen (sehr) einfachen Vokabelübersetzer realisiert, der zu einem gegebenen englischen Wort eine deutsche Entsprechung liefert (siehe Abbildung).

a) Skizzieren Sie die Hierarchie der GUI-Elemente des gezeigten Fensters; verwenden Sie dazu eindeutige Bezeichnungen.

b) Schreiben Sie eine Methode

```
private String translate(String english),
```

die den englischen String s auf deutsch übersetzen soll.

Da wir nur die Bedienoberfläche testen wollen, geben wir statt einer deutschen Übersetzung einfach das englische Wort rückwärts aus.

c) Schreiben Sie nun ein Java-Programm, so dass das vom Programm erzeugte Fenster der obigen Abbildung entspricht und das Programm bei Drücken von "Übersetzen" die deutsche Entsprechung aktualisiert. Bei Drücken von "Beenden" wird das Programm beendet.