**Ludwig-Maximilians-Universität München**
**Institut für Informatik**
Dr. Peer Kröger, Dr. Andreas Züfle
Daniel Kaltenthaler, Klaus Schmid

WS 2014/15
20.01.2015

# Klausur zur Vorlesung Einführung in die Programmierung

| | |
|---|---|
| Vorname: | |
| Name: | |
| Geb.-Datum: | |
| Matr.-Nr.: | |

**Die Klausur besteht aus 7 Aufgaben. Die Punktzahl ist bei jeder Aufgabe angegeben. Bitte überprüfen Sie, ob Sie ein vollständiges Exemplar erhalten haben.**

**Tragen Sie die Lösungen in den dafür vorgesehenen Raum im Anschluss an jede Aufgabe ein. Falls der Platz für Ihre Lösung nicht ausreicht, benutzen Sie bitte nur die ausgeteilten Zusatzblätter!**
**Tragen Sie bitte oben auf jeder ungeraden Seite Ihren Namen und Ihre Matrikelnummer ein.**

**Verwenden Sie keinen Rot-, Grün- oder Bleistift!**

| Aufgabe | mögliche Punkte | erreichte Punkte |
|---|---|---|
| 1. Allgemeine Fragen | 17 | |
| 2. Raute | 8 | |
| 3. MinMax | 4 | |
| 4. Datenstrukturen | 9 | |
| 5. Suche im Array | 9 | |
| 6. Polymorphismus | 6 | |
| 7. Typsicherheit | 7 | |
| Summe: | 60 | |
| Note: | | |

**Aufgabe 1** Allgemeine Fragen (4+2+3+1+3+2+2 Punkte)

**Allgemeine Fragen**

(a) Nennen Sie die wichtigsten Konzepte der OO-Programmierung und erläutern Sie diese kurz.

(b) Gegeben sei der folgende Java-Code:

```
public static void swap(int a,int b){
  int tmp = a;
  a = b;
  b = tmp;
}
```

der Code wird folgendermaßen aufgerufen:

```
public static void main(String[] args){
  int a = 1;
  int b = 2;
  swap(a,b);
  //*)
}
```

Was sind die Werte der Variablen `a` und `b` nach Ausführung des Codes $swap(a, b)$ an Position `//*)` der `main`-Methode? Warum?

(c) Erklären Sie kurz folgende Eigenschaften von Algorithmen:

- terminierend
- deterministisch
- partiell korrekt

(d) Welche der oben genannten Eigenschaften lässt / lassen sich durch das Hoare-Kalkül validieren?

(e) Beweisen Sie folgende Behauptung durch vollständige Induktion:

$$\sum_{k=1}^{n}(2k-1) \stackrel{?}{=} n^2$$

(f) Gegeben sei die folgende abstrakte Klasse:

```
public abstract class Sum{
  public int plus(int a,int b){
    return a + b;
  }
  public abstract boolean equals(Sum s);
}
```

Die Klasse wird folgendermaßen verwendet:

```
public static void main(String[] args){
  Sum s = new Sum();
  s.plus(2,3);
}
```

Die Verwendung der Klasse ist so nicht möglich. Wo liegt der Fehler? Wie kann man ihn beheben?

(g) Führen Sie (direkt auf Papier) folgende Umwandlungen durch:

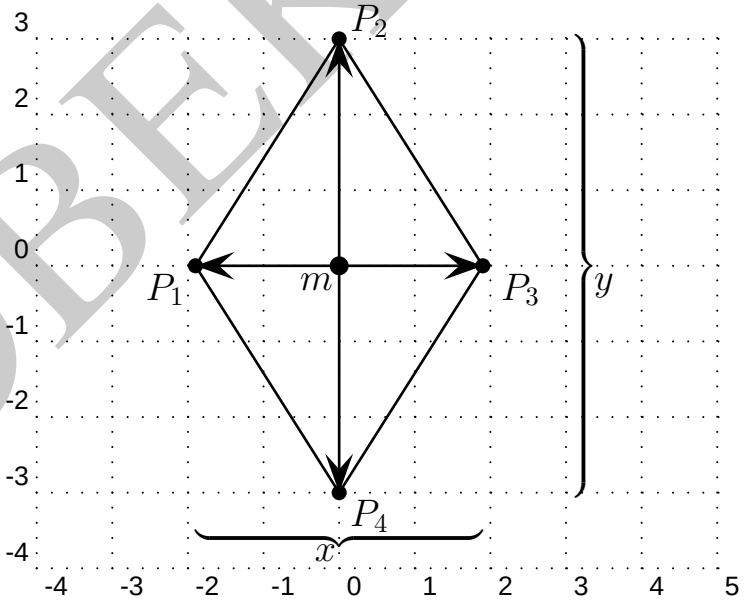
- Die Binärzahl `10010` in das dezimale System.

- Die Dezimalzahl `0,5` in das binäre System.

**Aufgabe 2** Modellierung (5+3 Punkte)

**Raute**

```
public class Punkt {
  private double xKoordinate;
  private double yKoordinate;

  public Punkt(double x, double y) {
    this.xKoordinate = x;
    this.yKoordinate = y;
  }

  public void verschiebeX(double betrag) {
    this.xKoordinate += betrag;
  }

  public void verschiebeY(double betrag) {
    this.yKoordinate += betrag;
  }

  public double getX() {
    return this.xKoordinate;
  }

  public double getY() {
    return this.yKoordinate;
  }
}
```




In dieser Aufgabe sollen Sie geeignete Klassenstrukturen zur Verwaltung von Rauten implementieren. Bei Rauten handelt es sich um Objekte, deren vier Seiten alle gleich lang sind und paarweise parallel in einem zweidimensionalen Koordinatensystem liegen.

Rauten werden spezifiziert durch den Mittelpunkt `m`, die Länge der horizontalen Diagonalen `x` und die Länge der vertikalen Diagonalen `y` (siehe Bild).

(a) Implementieren Sie eine Klasse `Raute`, die die Klasse `Punkt`, die Sie auf der vorherigen Seite finden, sinnvoll verwendet. Die Klasse soll einen geeigneten Konstruktor und geeignete Attribute bereitstellen. Achten Sie bei Ihrer Implementierung auf eine sinnvolle Datenkapselung.

(b) Definieren Sie für die Klasse Raute zusätzlich folgende Methoden:

- **`void`** `verschiebeX(`**`double`** `betrag)`:
  verschiebt die Raute um einen bestimmten Wert entlang der x-Achse.
- **`void`** `verschiebeY(`**`double`** `betrag)`:
  verschiebt die Raute um einen bestimmten Wert entlang der y-Achse.
- `Punkt getP1()`:
  liefert den linken Eckpunkt der Raute.
- `Punkt getP2()`:
  liefert den oberen Eckpunkt der Raute.
- `Punkt getP3()`:
  liefert den rechten Eckpunkt der Raute.
- `Punkt getP4()`:
  liefert den unteren Eckpunkt der Raute.

Implementieren Sie diese Methoden.

**Aufgabe 3** Arrays (4 Punkte)

**MinMax**

Definieren Sie eine statische Methode `minMax`, die ein Array vom Typ **`int`**`[]` als Parameter übergeben bekommt und das Maximum, also den größten Wert, sowie das Minimum, also den kleinsten Wert des Arrays berechnet. Beide Werte sollen gemeinsam in einem zweistelligen Integer-Array zurückgegeben werden. Falls das übergebene Array leer ist, soll **`null`** zurückgegeben werden. Durchlaufen Sie das Array dabei nur einmal.

Zur Definition der Methode dürfen **keine vordefinierten** Hilfsmethoden (z.B. aus der Java-API) verwendet werden, Sie dürfen aber bei Bedarf eigene Hilfsklassen schreiben.

**Aufgabe 4** (9 Punkte)

**Datenstrukturen**

Gegeben ist eine Klasse `Entry<T>`, die die Elemente einer typisierten Liste implementiert.

```
public class Entry<T> {
    /**
     * Eigentliches Element
     */
    private T element;
    /**
     * Verweis auf das nächste Element
     */
    private Entry<T> next;
    /**
     * Erzeugt und initialisiert ein Listen-Element
     * @param o     Wert für das eigentliche Element
     * @param next  Wert für den Verweis auf das nächste Element
     */
    public Entry(T o, Entry<T> next) {
        this.element = o;
        this.next = next;
    }
    /**
     * Liefert den Wert für das eigentliche Element zurück
     * @return  Wert für das eigentliche Element
     */
    public T getElement() {
        return this.element;
    }
    /**
     * Weist den Wert für das eigentliche Element zu
     * @param element   Wert für das eigentliche Element
     */
    public void setElement(T element) {
        this.element = element;
    }
    /**
     * Liefert den Wert für den Verweis auf das nächste Element zurück
     * @return  Wert für den Verweis auf das nächste Element
     */
    public Entry<T> getNext() {
        return this.next;
    }
    /**
     * Weist den Wert für den Verweis auf das nächste Element zu
     * @param next  Wert für den Verweis auf das nächste Element
     */
    public void setNext(Entry<T> next) {
        this.next = next;
    }
}
```

Definieren Sie eine Klasse `Stack`, die eine LIFO (Last-in-first-out)-Datenstruktur *Keller* mit den Mitteln einer **einfach-verketteten** Liste realisiert.

- Der Stack ist über die Anzahl der Elemente (`size`) und einen Verweis auf das erste Element definiert. Falls der Stack leer ist, ist das erste Element `null`.
- Die Klasse `Stack` enthält einen Konstruktor

  `public Stack()`,

  der einen leeren Stack erzeugt und entsprechend initialisiert.
- Die Klasse `Stack` enthält eine Methode

  `public int getSize()`,

  die den Wert für die Anzahl der Elemente auf dem Stack zurückliefert.
- Die Klasse `Stack` enthält eine Methode

  `public void push(T element)`,

  die ein neues Element auf dem Stack ablegt.
- Die Klasse `Stack` enthält eine Methode

  `public T pop()`,

  die das oberste Element (vom Typ `T`) entfernt und dieses zurückgibt. Falls sich kein Element auf dem Stack befindet, wird eine Fehlerbehandlung durchgeführt.

**Aufgabe 5** Arrays (3+6 Punkte)

**Suche im Array**

Zur Definition der Methoden dieser Aufgabe dürfen **keine vordefinierten** Hilfsmethoden (z.B. aus der Java-API) verwendet werden, Sie dürfen aber bei Bedarf eigene Hilfsklassen schreiben.

(a) **Suche im unsortierten Array** Definieren Sie eine statische Methode `sucheLinear`, die einen zu suchenden **`int`**-Wert sowie ein Array vom Typ **`int`**`[]` als Parameter übergeben bekommt und mittels eines **`boolean`**-Wertes angibt, ob das Array den ersten Parameter enthält (`true` für den Fall, dass der Wert im Array enthalten ist). Durchsuchen Sie dazu das Array auf iterative Weise.

(b) **Suche im sortierten Array** Definieren Sie nun eine statische Methode `sucheBinaer`, die mit den gleichen Ein- und Ausgabeparametern das Array rekursiv nach dem Prinzip der **binären Suche** durchsucht. Dazu darf eine Hilfsmethode für den internen Rekursionsaufruf verwendet werden.

Nehmen Sie dazu an, dass das angegebene **`int`**-Array die Werte der Größe nach aufsteigend sortiert hält. Die binäre Suche beginnt dazu mit der mittleren Zelle des Arrays und vergleicht, ob der Eintrag mit dem Suchwert übereinstimmt. Falls ja, wird `true` zurückgegeben. Falls nein, wird die Suche auf der vorderen Hälfte (falls der Suchwert kleiner ist) oder hinteren Hälfte des Arrays wiederholt. Die Suche endet spätestens, wenn das betrachtete Array die Länge `1` hat.

**Aufgabe 6** Polymorphismus (4+2 Punkte)

**Polymorphismus**

Gegeben sind die untenstehenden Klassen `A`, `B` und `C`.

```
public class A {

  private int a;

  public A(int a){
    this.a = a;
  }

  public int getA(){
    return this.a;
  }

  public int m(){
    return 1;
  }

  public int n() {
    return this.m();
  }
}
```

```
public class B extends A {

  private int a;

  public B(int a){
    super(a);
  }

  public int getA(){
    return this.a;
  }

  public int m(){
    return 2;
  }

  public int n() {
    return this.m();
  }
}
```

```
1  public class C {
2
3    public static void main(String[] args) {
4
5      A o1 = new A(2);
6      B o2 = new B(3);
7      A o3 = new B(4);
8
9      System.out.println(o1.m());
10     System.out.println(o2.m());
11     System.out.println(o3.m());
12
13     System.out.println(o3.n());
14   }
15 }
```

(a) Welche Ausgaben liefert das Programm bei Ausführung der Klasse `C`? Begründen Sie jede einzelne Ausgabe.

(b) Welche Ausgabe liefert die Zeile 13 bei Ausführung der Klasse `C`, wenn Sie zuvor die Methode `int n()` der Klasse `A` auskommentieren? Begründen Sie Ihre Antwort.

**Aufgabe 7** Generics (2+5 Punkte)

**Typsicherheit**

Gegeben sind folgende Klassen zur Modellierung eines Zoos mit Bewohnern:

```
1  public class Zoo {
2          public static void main(String[] args) {
3                  Haus elefantenhaus = new Haus();
4                  Elefant jumbo = new Elefant();
5                  jumbo.trompeten();
6                  jumbo.fressen();
7                  elefantenhaus.betreten(jumbo);
8
9                  Waerter waerter = new Waerter();
10
11                 // als er gerade nicht hinsieht, passiert das:
12                 elefantenhaus.verlassen();
13                 elefantenhaus.betreten(new Muecke());
14
15                 waerter.elefantenBuersten((Elefant) elefantenhaus.verlassen());
16         }
17 }
```

```
1  public class Haus {
2          private Tier bewohner;
3
4          public boolean belegt () {
5                  return this.bewohner != null;
6          }
7
8          public boolean betreten (Tier t) {
9                  if (this.bewohner != null) return false;
10                 this.bewohner = t;
11                 return true;
12         }
13
14         public Tier verlassen () {
15                 Tier t = this.bewohner;
16                 this.bewohner = null;
17                 return t;
18         }
19 }
```

```
1  public class Tier {
2          public void fressen () {
3                  System.out.println("Omnomnom");
4          }
5  }
```

```
1  public class Elefant extends Tier {
2          public void trompeten () {
3                  System.out.println("Toot!");
4          }
5  }
```

```java
public class Muecke extends Tier {
        public void summen () {
                System.out.println("Bzzzzzzz");
        }
}
```

```java
public class Waerter {
        public Waerter (String n) {
        }

        public void elefantenBuersten (Elefant e) {
                e.fressen();
                e.trompeten();
        }
}
```

(a) Bei Ablauf des Programmes `Zoo` wird ein Fehler auftreten. Um was für einen Fehler handelt es sich, warum und an welcher Stelle (Klassenname, Zeilennummer) tritt er auf?

(b) Verbessern Sie die Modellierung durch Typvariablen so, dass bei ihrer Verwendung Typsicherheit zur Übersetzungszeit erreicht wird. Demnach sollen im Elefantenhaus nur Elefanten unterkommen, auch wenn die Klasse `Haus` weiterhin mit allen Tieren funktioniert. Ansonsten soll sich aber im Ablauf und Ergebnis des Programmes nichts ändern.

(Es genügt, wenn Sie hier nur die geänderten Zeilen unter Angabe von Klassenname und Zeilennummer eintragen.)

**Ergänzung zu Aufgabe: _____**

**Ergänzung zu Aufgabe: _____**

**Ergänzung zu Aufgabe: _____**

**Ergänzung zu Aufgabe: _____**

**Ergänzung zu Aufgabe: _____**