Ludwig-Maximilians-Universität München
Institut für Informatik
PD Dr. Peer Kröger
Andreas Züfle, Johannes Niedermayer

WS 2012/13
xx.xx.xxxx

Beispielklausur zur Vorlesung Einführung in die Programmierung

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

Die Klausur besteht aus 5 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	20	
2. Quader	10	
3. UML und Java	10	
4. Queue	13	
5. Hoare-Kalkül	10	
Summe:	63	
Note:		

Entwertung der Klausur
Meine Klausur soll nicht korrigiert und nicht gewertet werden.

München, den xx.xx.xxxx **Unterschrift:** _____

Aufgabe 1 Allgemeine Fragen (4+2+2+2+5+2+3 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 voic 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) Am Anfang der Vorlesung haben wir das funktionale Programmierparadigma und mathematische Folgen kennengelernt. Die induktive Definition von Folgen, die unterliegenden Mengen M^+ und M^*, die leere Folge ϵ sowie der Konkatenationsoperator $\circ$ sind aus der Vorlesung bekannt. Sei nun $M = \mathbb{N}$, die Maximumsfunktion $max : \mathbb{N} \times \mathbb{N} \to \mathbb{N}, max(a, b) = \begin{cases} a, & \text{falls } a > b \\ b & \text{sonst} \end{cases}$ kann als gegeben vorausgesetzt werden. Implementieren Sie nur unter Verwendung des Konkatenationsoperators und der Maximumsfunktion eine Operation $max : \mathbb{N}^+ \to \mathbb{N}$, die das Maximum einer gegebenen Folge natürlicher Zahlen bestimmt.

(d) Implementieren Sie eine statische Methode `max(int[] werte)` in Java, die ein Integer-Array als Argument erhält und den größten in diesem Array enthaltenen Wert zurückgibt. Sie können davon ausgehen, dass das Array eine Länge von mindestens 1 hat.

(e) Implementieren Sie eine statische Methode `findeSubarray(int[] werte, int[] subarray)` in Java, die zwei Integer-Arrays `werte` und `subarray` als Argument erhält. Die Funktion soll testen, ob das Array `subarray` im Array `werte` zusammenhängend vorkommt. Die Methode soll `true` zurückgeben, falls das Teilarray gefunden wurde, ansonsten `false`. Sie können davon ausgehen, dass beide Arrays eine Länge von mindestens 1 haben.

(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) Gegeben sei die folgende Klasse $Mensch$ und die Klasse $Student$

```
public class Mensch{
  int alter;
  public Mensch(int alter){this.alter =alter;}
  public String toString(){return "Mensch(" + alter + ")";}
}

  public class Student extends Mensch{
  int matrnr;
  public Student(int alter, int matrikelnummer){
    super(alter);
    this.matrnr =matrikelnummer;
  }
  public String toString(){return "Student(" + matrikelnummer + ")";}
}
```

Die Klassen werden folgendermaßen verwendet:

```
Student s = new Student(20, 12345);
Object o = (Object) s;
Mensch m = (Mensch) s;
System.out.println(o.toString()); // 1)
System.out.println(m.toString()); // 2)
System.out.println(s.toString()); // 3)
```

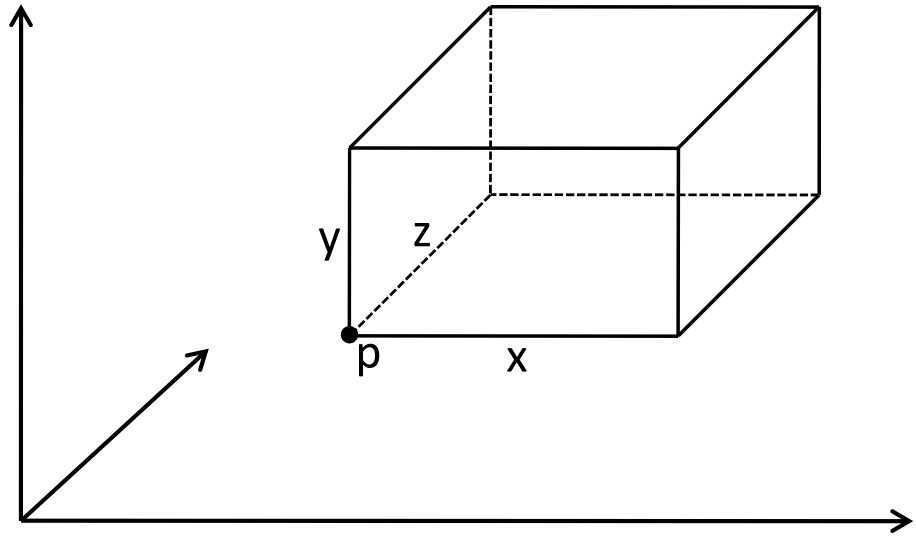
Was wird an den Markierungen `1)`, `2)` und `3)` ausgegeben?

Aufgabe 2 Modellierung (5+5 Punkte)

Quader

Bei Quadern handelt es sich um Objekte, deren Kanten achsenparallel in einem dreidimensionalen Koordinatensystem liegen.

Quader werden spezifiziert (siehe Bild) durch einen der Eckpunkte `p` und die Länge der an `p` angrenzenden Kanten `x` (parallel zur x-Achse), `y` (parallel zur y-Achse) und `z` (parallel zur z-Achse).



Hinweis: Achten Sie im Folgenden in der gesamten Aufgabe auf sinnvolle Datenkapselung.

(a) Implementieren Sie eine Klasse `Punkt3D` zur Verwaltung dreidimensionaler Punkte in einem reellen Koordinatensystem als Erweiterung der folgenden Klasse `Punkt2D`:

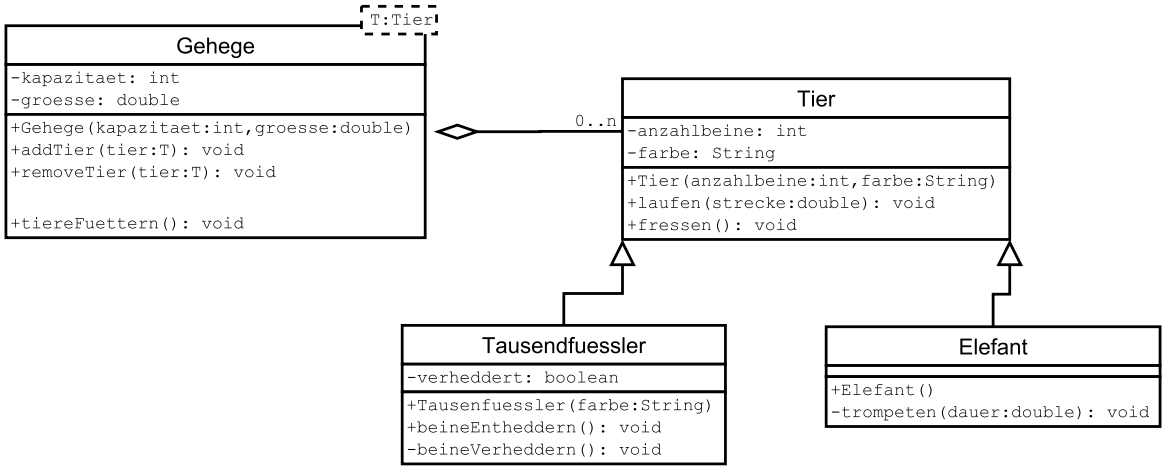
```
public class Punkt2D {
  private double xKoordinate;
  private double yKoordinate;

  public Punkt2D(double x, double y) {
    this.xKoordinate = x;
    this.yKoordinate = y;
  }

  public void verschiebe(double x, double y) {
    this.xKoordinate += x;
    this.yKoordinate += y;
  }

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

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

Definieren Sie neben einem geeigneten Konstruktor und geeigneten Attributen die folgende Methode:
`void verschiebe(double x, double y, double z)`,
die den Punkt um `x` entlang der x-Achse, um `y` entlang der y-Achse und um `z` entlang der z-Achse verschiebt.

(b) Implementieren Sie eine Klasse `Quader`, die die Klasse `Punkt3D` aus der vorherigen Teilaufgabe sinnvoll verwendet. Die Klasse soll einen geeigneten Konstruktor und geeignete Attribute bereitstellen. Implementieren Sie für die Klasse Quader zusätzlich folgende Methode:
`void verschiebe(double x, double y, double z)`,
die den Quader um `x` entlang der x-Achse, um `y` entlang der y-Achse und um `z` entlang der z-Achse verschiebt.
Hinweis: Getter und Setter brauchen Sie in dieser Aufgabe nicht explizit angeben.

Aufgabe 3 Objektorientierter Entwurf (10 Punkte)
UML und Java

Gegeben ist das folgende UML-Klassendiagramm.



Implementieren Sie die Klassen Tier.java, Gehege.java und Elefant.java anhand der aus dem Klassendiagramm ersichtlichen Vorgaben. Die Logik innerhalb der Funktionen ist hierbei nicht ausschlaggebend, achten Sie aber darauf, dass Ihre Klassen fehlerfrei kompilieren würden.

Aufgabe 4 Datenstruktur **Queue** (4+5+4 Punkte)

Gegeben folgende Klasse zur Modellierung einer Warteschlange (Queue) mit den Mitteln einer doppelt verankerten Liste:

```
public class Queue<E> {
  private int size;
  private Entry<E> firstEntry;
  private Entry<E> lastEntry;

  public Queue() {
    this.firstEntry = null;
    this.lastEntry = null;
  }

  public int size() {
    return this.size;
  }
}

public class Entry<E> {
  private E element;
  private Entry<E> next;

  public Entry(E o, Entry<E> next) {
    this.element = o;
    this.next = next;
  }

  public E getElement() {
    return this.element;
  }

  public void setElement(E element) {
    this.element = element;
  }

  public Entry<E> getNext() {
    return this.next;
  }

  public void setNext(Entry<E> next) {
    this.next = next;
  }
}
```

Ergänzen Sie diese Klasse um folgende Methoden, wobei Sie die Eigenschaft der Queue als Datenstruktur nach dem FIFO (First-in-first-out)-Prinzip beachten müssen:

(a) Die Methode **`public void`** `put(E element)` hängt das gegebene Element `element` an das Ende der Liste an.

(b) Die Methode **`public`** `E get()` gibt das erste Element (d.h. das Element, das momentan von allen Elementen am längsten in der Liste gespeichert wird) der Liste aus und löscht dieses Element aus der Liste.

(c) Die Methode **`public boolean`** `contains(E element)` gibt **`true`** zurück, wenn das gegebene Element `element` in der Liste vorhanden ist, ansonsten **`false`**. Die Elemente sollen dabei auf Gleichheit getestet werden.

Aufgabe 5 Korrektheitsbeweis (10 Punkte)
Hoare-Kalkül

Beweisen Sie mit den Mitteln des Hoare-Kalküls die partielle Korrektheit des folgenden Programmstücks. Dabei seien `x`, `y`, `i`, `result` Variablen vom Typ **`int`**.
Verwenden Sie die aus der Vorlesung bekannte Notation!

```
// Vorbedingung: x >= 1 && y >= 1
result = x;
i = 1;
while(i < y)
{
  i = i+1;
  result = result+x;
}
// Nachbedingung: result == x*y
```