Ludwig-Maximilians-Universität München
Institut für Informatik
PD Dr. Peer Kröger
Janina Bleicher, Florian Richter

WS 2016/17
09.02.2017

Wiederholungsblatt
Einführung in die Programmierung

Dies ist eine Wiederholung der Vorlesung EIP. Das Layout wurde an das Klausurschema angenähert, um Sie damit vertraut zu machen. Der Inhalt dieser Zusatzübung ist kein Garant für die inhaltliche Zusammenstellung der Klausur. Die Punktzahl dient nur zur zeitlichen Orientierung und wird nicht auf die Übungspunkte angerechnet. Die Lösung für diese Übung wird ausschließlich in der Vorlesung am 09.02.2017 präsentiert und nicht online gestellt.

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	21	
2. Arrays	5	
3. Klassenhierarchie	15	
4. Programmanalyse	10	
5. Rekursive Datenstrukturen	12	
6. Typsicherheit	10	
Summe:	73	
Note:		

Aufgabe 1 Allgemeine Fragen (21 Punkte)

(a) 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?

(b) 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.

(c) Erklären Sie knapp die catch-or-throw-Regel.

(d) 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?

(e) Ergänzen Sie die Tabelle so, dass in jeder Zeile die verschiedenen p-adischen Zahlendarstellungen für die selbe Zahl stehen.

$p = 2$	$p = 5$	$p = 8$	$p = 10$	$p = 16$
00010100				
	41			
		113		
			99	
				39

(f) Gegeben seien drei (sechsseitige) Würfel. Aufgabe ist es, diese Würfel so auf einem Tisch anzuordnen, dass alle Würfel die selbe Augenzahl zeigen. Dafür werde folgender Algorithmus vorgeschlagen.

```
(1) Wirf alle drei Würfel auf den Tisch
(2) Falls nicht alle Würfel die selbe Augenzahl zeigen,
nimm die Würfel auf und beginne bei (1)
```

Entscheiden Sie, welche der folgenden Eigenschaften dieser Algorithmus erfüllt. Begründen Sie Ihre Entscheidung kurz.

- Terminierend
- Deterministisch
- Partiell korrekt
- Total korrekt

Aufgabe 2 Arrays (5 Punkte)

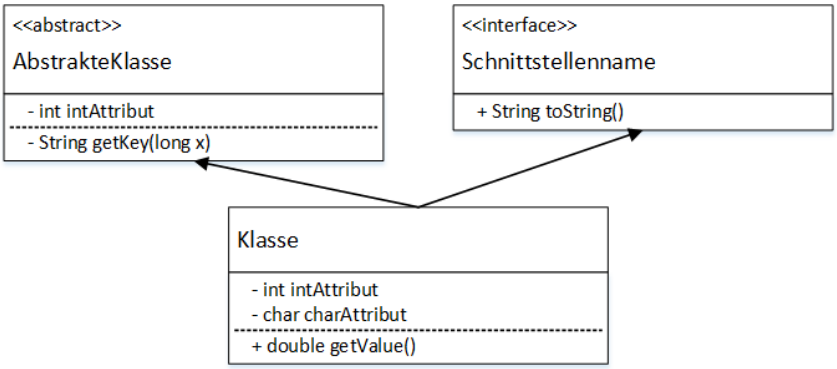
(a) Implementieren Sie eine statische Methode `schnittmenge(int[] m1, int[] m2)` in Java, die zwei Arrays aus natürlichen Zahlen ohne 0 `m1` und `m2` als Argument erhält. Die Funktion soll das Array berechnen, dass alle Werte enthält, die in der Schnittmenge von `m1` und `m2` liegen. Falls die Schnittmenge leer ist, soll der Rückgabewert nur ein Array bestehend aus 0 sein. Sie dürfen hierbei die Reihenfolge und Duplikate ignorieren.

Aufgabe 3 Klassenhierarchie (15 Punkte)

In dieser Aufgabe geht es um die Modellierung einer Klassenhierarchie zur Verwaltung von Serien und Filmen.

- Ein Videomedium wird durch einen Titel identifiziert.
- Jedes Videomedium ist entweder ein Film oder eine Serie.
- Jeder Film hat eine bestimmte Länge in Minuten. Jede Serie hat eine Anzahl von Episoden.
- Ein Film bietet eine Methode, um einen Filmpreis zu bekommen.
- Es gibt Serien sowie Filme, die Nachfolger eines anderen Videomediums sind. Ein Nachfolger muss seinen Vorgänger kennen und daher eine entsprechende Methode besitzen.
- Der Nachfolger einer Serie heißt Ableger. Ein Ableger wird einfach dadurch charakterisiert, ob er gut ist oder nicht.
- Filme haben hingegen Fortsetzungen. Jeder Film hat eine Bewertung zwischen 0 (schlecht) und 10 (gut).

(a) Entwerfen Sie eine geeignete Klassenhierarchie für die oben aufgelisteten Sachverhalte. Beachten Sie dabei die Prinzipien der Datenkapselung. Sie müssen keine Selektoren oder Konstruktoren definieren. Gemeinsame Methoden und Attribute müssen in geeigneten Oberklassen bzw. Interfaces zusammengefasst werden. Notieren Sie Klassen als abstrakt, wenn dies sinnvoll ist. Verwenden Sie die folgende Notation:



(b) Implementieren Sie eine Methode `videomarathon`, die die Zeit zurückgibt, die für die übergebenen Videomedien zum Anschauen benötigt wird. Pausen müssen nicht berücksichtigt werden. Ein Film benötigt dabei seine volle Länge. Eine Serie wird immer komplett geschaut. Jede Serienepisode soll exakt 40 Minuten Länge haben. Bei einem Film müssen auch alle Vorgänger geschaut werden bis zum Ursprungswerk. Benutzen Sie die Methodensignatur `int videomarathon(Videomedium[] medium)`.

Aufgabe 4 Programmanalyse (10 Punkte)

Geben Sie die Ausgabe des Programms für den Aufruf `java M` an. Tragen Sie hierzu jeweils die ausgegebenen Zeichen in die markierten Stellen hinter „`OUT:`“ ein.

```
public class A {
  public static long x = 6;
  public static int y = 1;

  public A(int i) {
    x += 1;
  }

  public A(double d) {
    x -= (int) d;
  }

  public void f(double x) {
    x += 1;
    y = A.y + 1;
  }

  public void f(int x) {
    A.y = x;
  }
}
```

```
public class B extends A {
  public int x = 1;
  public static int y = 17;

  public B(int x) {
    super(x);
    this.x += x;
    this.y = (new A(new Long(x))).y;
  }

  public void f(double x){
    this.x = (int)x;
    A.x = super.x + 3;
  }
}
```

```
public class M {
  public static void main(String[] args) {
    A a = new A(3L);
    System.out.println(a.x + " " + A.y);         // OUT1: [   ] [   ]

    a.f(4);
    System.out.println(A.y);                     // OUT2: [   ]

    B b = new B(3);
    System.out.println(((A) b).x + " " + b.x);   // OUT3: [   ] [   ]

    System.out.println(B.y);                     // OUT4: [   ]

    b.f(2.);
    System.out.println(b.x + " " + A.x);         // OUT5: [   ] [   ]

    A ab = b;
    ab.f(1.);
    System.out.println(ab.x + " " + ((B)ab).x); // OUT6: [   ] [   ]

    ab.f(5);
    System.out.println(A.x + " " + A.y);         // OUT7: [   ] [   ]
  }
}
```

Aufgabe 5 Rekursive Datenstrukturen (12 Punkte)

Folgende Klassen seien gegeben:

```
public class List {
  public Entry head;
}
```

```
public class BinTree {
  public Node root;
}
```

```
public class Entry {
  public Entry next;
  public int value;
}
```

```
public class Node {
  public Node left;
  public Node right;
  public int value;
}
```

(a) Es gibt nur die Klasse `Node` für die Knoten eines Baumes. Wie können Blätter des Baumes identifiziert werden?

(b) Ein Binärbaum der Höhe h hat maximal 2^{h-1} Blätter. Zeigen Sie dies durch vollständige Induktion.

(c) Ein Binärbaum der Höhe h hat maximal $2^h - 1$ Knoten. Zeigen Sie dies durch vollständige Induktion.

(d) Implementieren Sie die Methode `int sum()`, die die Summe aller Werte eines Baumes berechnet. Verwenden Sie dabei nur Rekursion und keine Schleifen.

(e) Implementieren Sie die Klasse `TreeList`, die eine Liste von Bäumen modelliert.

Aufgabe 6 Typsicherheit (10 Punkte)

Die folgenden Klassen modellieren verschiedene Typen von Getränken sowie Flaschen-Verschlüssen.

```
public class Getraenk {}

public class Cola extends Getraenk {}

public class Bier extends Getraenk {}
```

```
public class Verschluss {}

public class Korken extends Verschluss {}

public class Schraubdeckel extends Verschluss {}
```

Definieren Sie in Java eine Klasse `Flasche`, die per Typparameter nur die Füllung mit einem Typ Getränk sowie nur eine Form von Verschluss zulässt. Achten Sie auf eine geeignete Kapselung der Attribute.

`Flasche` sollte ausschließlich im Konstruktor mit Inhalt und Verschluss versehen werden können. Eine Methode `oeffnen` soll den Verschluss zurück geben, eine Methode `leeren` ihren Inhalt.

Eine Flasche soll nur einmal geöffnet und geleert werden können, und eine Leerung nur dann möglich sein, wenn die Flasche bereits geöffnet wurde.

Die Klasse `Flasche` soll sich mit folgendem Beispielaufruf verwenden lassen:

```
Flasche<Korken,Cola> colaFlasche =
                new Flasche<Korken,Cola>(new Korken(), new Cola());
Verschluss v = colaFlasche.oeffnen();
Getraenk g = colaFlasche.leeren();
```

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