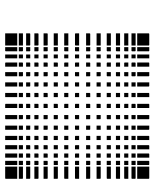

hochschule mannheim

fachbereich|informatik

Techniken der Programmentwicklung (TPE) **Prof. Dr. Wolfgang Schramm**
IB / IBM
Sommersemester 2009 Klausur

Name: .....

Vorname: .....

Matrikel-Nr.: .....

Unterschrift: .....

Hinweise:

1.) Schreiben Sie Ihren Namen und Ihre Matrikelnummer zu Beginn auf das Deckblatt der Klausur. Überprüfen Sie, ob die Klausur vollständig ist.

2.) Bearbeiten Sie die Aufgaben möglichst auf den jeweiligen Blättern. Der Platz auf dem Aufgabenblatt ist so bemessen, dass er für die Lösung der Aufgabe ausreicht. Andernfalls verwenden Sie die Rückseite oder ein mit Ihrem Namen, Ihrer Matrikelnummer und der Aufgabennummer gekennzeichnetes separates Blatt.

3.) Als Hilfsmittel ist, außer einem Schreibgerät, nichts zugelassen.

4.) Schreiben Sie mit dokumentenechten Stiften. Mit Bleistiften, Tintenkiller o.ä. erstellte Lösungen sind ungültig! Rotschreibende Stifte sind ebenfalls verboten!

Schreiben Sie bitte leserlich!

Aufgabe	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**Σ**
Erreichbare Punkte	12	10	15	10	24	24	25	120
Erreichte Punkte								

V i e l E r f o l g ! ! !

Note:

Name: **Matrikelnr.:**

Aufgabe 1 Klassen und Objekte 12 (6 + 6) Punkte

Ein Java−Programm enthalte folgende Klassenhierarchie:

```
class A {
      protected int i;
      public A (int i) {this.i = i;}

      public void  m1 () {System.out.println (i); }
      public void  m2 () {System.out.println (i*2); }
}

abstract class Unter_A extends A {
      protected int i;

      public Unter_A (int i) {super(i*2); this.i = i;}

      public void  m1 () {super.m1(); System.out.println (i); }
      abstract public void  m2 ();
}

class Unter_Unter_A extends Unter_A {
      protected int i;

      public Unter_Unter_A (int i) {super(i*2); this.i = i;}

      public void  m1 () {super.m1(); System.out.println (i); }
      public void  m2 () {System.out.println (i); }}
```

a) Welche Ausgabe erzeugen die folgenden Programmfragmente bei den Methodenaufrufen?

```
A a = new A(1);
a.m1();
Unter_Unter_A uua = new Unter_Unter_A (1);
uua.m1();
uua.m2();
```

b) Welche Programmfragmente sind korrekt, welche führen zu einem Fehler? Nur bei Fehlern: begründen Sie diese **kurz**.

```
A a = new Unter_A (1);
A a = new Unter_Unter_A(1);
Unter_A ua = new Unter_Unter_A (1);
Unter_Unter_A uub = a;
Unter_Unter_A uua = new Unter_Unter_A (1);
Unter_Unter_A uub = (Unter_Unter_A) uua;
```

Aufgabe 2 Objektinitialisierung / Konstruktoren / Polymorphismus 10 Punkte

In den Klassen aus Aufgabe 1 sind die Methoden `SetI` ergänzt und die Konstruktoren verändert. Betrachten Sie folgendes Programmfragment:

```
class A {
      protected int i;
      protected void setI (int i) {this.i = i;}
      public A (int i) {this.setI(i);}

      public void  m1 () {System.out.println (i); }
      public void  m2 () {System.out.println (i*2); }
}

abstract class Unter_A extends A {
      protected int i;

      public Unter_A (int i) {super(i*2); this.i = i;}

      public void  m1 () {super.m1();System.out.println (i); }
      abstract public void  m2 ();
}

class Unter_Unter_A extends Unter_A {
      protected int i;
      protected void setI (int i) {this.i = i;}

      public Unter_Unter_A (int i) {super(i*2); this.setI(i);}

      public void  m1 () {super.m1();System.out.println (i); }
      public void  m2 () { System.out.println (i); }
}

. . .

A a = new Unter_Unter_A (1);

a.m1();
((A)a).m2();
```

Was wird ausgegeben? Schauen Sie sich das Programmstück genau an und schreiben Sie Ihre Antwort erst dann hin, wenn Sie alles bedacht haben!

Aufgabe 3 Exceptions 15 Punkte

Gegeben ist die folgende Klasse `Exceptions` mit Methoden und Exception Handling.

```
public class Exceptions {

	public void a1 (int start, int end) {
		int failures = 0;
		for (int i = start; i < end; i++) {
			try {
				if (i%4 == 0 || 4/i == -1)
					throw new RuntimeException ("A1 No foo");
				if (i%2 == 0)
					throw new RuntimeException ("A1 No bar");
			} catch (RuntimeException e) {
				System.out.println("A1: try failed: "
						+ e.getMessage());
				failures++;
				if (failures >= 3)
					throw new RuntimeException ("A1: 3 times");
			} finally {
				System.out.println("A1 done with " + i);
			}
		}
		throw new RuntimeException ("A1: Catch me");
	}

	public void a2 () {
		a1(3, 15);
	}

	public void a3 () throws Exception {
		try {
			a1 (0, 2);
		} catch (Exception e) {
			System.out.println("A3: A1: "
					+ e.getMessage());
			throw e;
		}
	}

	public static void main(String[] args) throws Exception {
		Exceptions eo = new Exceptions();
		try {
			try {
				eo.a2();
			} catch (Exception e) {
				System.out.println("Main: A2 threw Exception: "
						+ e.getMessage());
			}
			eo.a3();
		} catch (Exception e) {
			System.out.println("Main: An exception occured ");
			e.printStackTrace(System.out);
			System.out.println("Main: I am going to rethrow it now.");
			throw e;
		} finally { System.out.println("Bye"); }
	}
}
```

Aufgabe 3 Exceptions (Forts.) 15 Punkte

Was gibt das Programm aus?

Lösung:

Aufgabe 4 Interfaces 10 Punkte

Schreiben sie eine statische Methode `sortArray`, welche als Parameter ein Sortierverfahren und ein zu sortierendes Array mit `int`-Werten bekommt.

Geben Sie nur das benötigte Interface, die Klassenköpfe und die die benötigten Methoden-Köpfe an. Geben Sie außerdem den Aufruf von `sortArray` für die beiden Sortierverfahren ShakerSort und SelectionSort, welche Sie aus ADS kennen, an.

Sie müssen die Implementierung der Sortierverfahren natürlich **nicht** angeben. Aber die Klassen für die Sortierverfahren müssen so realisiert werden, dass Objekte dieser Klassen als Parameter an die Methode `sortArray` übergeben werden können.

Name: **Matrikelnr.:**

Aufgabe 5 IO-Streams 24 (14 +10) Punkte

a) Definieren Sie eine **Filterklasse** zur Ausgabe von Text. Dabei werden die Umlaute des Textes (das sind die Buchstaben ä, ö, und ü) als Buchstabenfolgen (ae, oe und ue) geschrieben.

Geben sie die **Klassendefinition** für diesen Ausgabefilter an. Die Filterklasse ist eine Unterklasse der Klasse FilterWriter mit den 3 Methoden:

```
public void write(int c)
public void write(char[] b, int off, int len)
public void write(String s, int off, int len
```

Lösung:

Aufgabe 5 IO-Streams (Fortsetzung) 24 (14 +10) Punkte

b) Zeigen Sie die Verwendung Ihrer Filterklasse, indem Sie in einem Hauptprogramm eine Textdatei einlesen und die einzelnen Zeichen mit Ihrem Filter aus Teil a) auf eine Ausgabedatei schreiben. Ein- und Ausgabe soll zudem **gepuffert** erfolgen.

Eine nützliche Methode der Klasse Reader:

`public boolean ready``()`
Liefert `true`, wenn Daten zum Lesen aus dem Stream bereitstehen, so dass ein nachfolgender Aufruf von `read()` nicht blockiert, sonst `false`.

In main:

Aufgabe 6 Threads 24 (10 + 8 + 6) Punkte

Wir betrachten die Klassen `Producer` und `Consumer`. Beide sind eine Spezialisierung der Klasse `Thread`.
`Producer` und `Consumer` schreiben bzw. lesen ihre Daten aus einer `Queue`.

Die `Queue` hat folgendes Interface:

```
public interface Queue {
    public boolean enter (Object e);
    public Object leave ();
    public boolean isEmpty();
    public boolean isFull();
}
```

a) Implementieren Sie die Methode `enter` der Klasse `QueueArray`, so dass der Zugriff auf die Datenstruktur **sicher** und **verklemmungsfrei** ist. Überlegen Sie, was zu tun ist, wenn die Queue voll ist.

Die Klasse `QueueArray` bietet folgendes Gerüst (Sie können weitere Datenelemente hinzufügen. Die o.g. Methoden von `Queue` dürfen Sie für die Implementierung von `enter` verwenden.):

```
public class QueueArray implements Queue {

    private Object queue [];
    private int rp = 0; // read position
    private int wp = 0; // write position

. . .
```

Lösung:

Aufgabe 6 Threads (Fortsetzung) 24 (10 + 8 + 6) Punkte

b) Implementieren Sie die Klasse `Producer`, deren Konstruktor ein Objekt vom Typ `Queue` als Parameter hat. Ein `Producer` soll endlos lange alle 2 Sekunden einen neuen `int`-Wert in die Queue schreiben. Beachten Sie, dass der `Producer` von außen terminiert werden kann.

c) Geben Sie den Teil eines Hauptprogramms an, in welchem zwei `Producer` erzeugt, gestartet und nach 1 Minute sicher beendet werden.

Aufgabe 7 Iteratoren und lokale Klassen 25 Punkte

Wenn man alle natürlichen Zahlen kleiner 20 addiert, die Vielfache von 3 oder 5 sind, erhält man 3 + 5 + 6 + 9 + 10 + 12 + 15 + 18 = 78. Hinweis: Die 15 wird nur einmal addiert.

Schreiben Sie ein Programm, welches eine Antwort auf die Frage findet, wie die Summe aller Zahlen ist, welche kleiner als 1000 und Vielfache von 3 oder 5 sind.

Entwickeln Sie hierzu die Klasse

`public class` `ZahlenQuelle1000`, welche die Zahlen 1 bis 1000 repräsentiert.

Innerhalb der Klasse `ZahlenQuelle` definieren Sie eine **lokale <u>private</u>** Klasse `Iterator35`, welche einen Iterator für die Klasse Zahlenquelle implementiert, der immer die nächsthöhere Zahl, welche ein Vielfaches von 3 oder 5 ist, zurückliefert.

Der Iterator liefert also die Werte 3, 5, 6, 9, . . .

Überlegen Sie, wie die lokale `Iterator35`-Klasse aussehen muss und wie der Iterator, welcher von `Iterator35` implementiert wird, von der Klasse `ZahlenQuelle` zur Verfügung gestellt wird.

Die Berechnung der Summe aller Vielfachen von 3 und 5 unterhalb von 1000 findet in der main-Methode (die Sie auch angeben müssen) statt.

Lösung:

Aufgabe 7 Iteratoren und lokale Klassen (Fortsetzung) 25 Punkte