Universität Mannheim
Fakultät für Mathematik und Informatik
Dr. Heinz Kredel

Teilprüfung
Software- und Internet-Technologie
April 2003: Programmierkurs 1

Name: .... Vorname: ....

Matrikel-Nr.: .... Semester: .... Fach: ....

Hinweise:

1. Bitte füllen Sie sofort den Kopf des Deckblattes aus.
2. Unterschreiben Sie die Klausur auf der letzten Seite.
3. Überprüfen Sie bitte Ihr Klausurexemplar auf Vollständigkeit (**12** Seiten).
4. Tragen Sie die Lösungen – soweit möglich – direkt in die Klausur ein.
5. Eigenes Papier und sonstige Hilfsmittel (z.B. Taschenrechner, Palm, Handy) sind nicht zugelassen.
6. Bearbeitungszeit: 66 Minuten.

Aufgabe	max. Punktzahl	Punkte
1	9	
2	13	
3	11	
4	11	
5	10	
6	12	
Summe	66	

Aufgabe 1: Multiple-Choice-Aufgaben *[9 Punkte]*

Beantworten Sie die folgenden Fragen (es ist jeweils genau eine Lösung richtig):

a) [2 Punkte] Betrachten Sie das folgende Codefragment in Java:

```
public class Dummy {
  private static int a = 0;
  ...
}
```

Welche der folgenden Aussagen ist korrekt?

- ☐ Variable `a` darf nur von Methoden der Klasse `Dummy` verändert werden.
- ☐ Variable `a` darf nur von Methoden benutzt werden, die zur Klasse `Dummy`, zu davon abgeleiteten Klassen oder zu Klassen des gleichen Paketes gehören.
- ☐ Variable `a` darf von Methoden aller Klassen verändert werden.
- ☐ Variable `a` ist konstant und darf daher gar nicht verändert werden.
- ☐ Das obige Codefragment führt zu einem Compilationsfehler.

b) [2 Punkte] Welche der folgenden Feld-Definitionen ist fehlerhaft?

- ☐ `int[] Feld1 = new int[4];`
- ☐ `int Feld2[] = new int[4];`
- ☐ `int[] Feld3 = {1,4,9,16}`
- ☐ `int[] Feld4 = new int[]{1,4,9,16};`
- ☐ Keine, es sind alle korrekt.

c) [2 Punkte] Welchen Wert gibt die folgende Methode zurück?

```
int verwirrung() {
  int a = 0; int i = -1;
  switch(a) {
    case 0:  i = 0;
    case 1:  i = 1;
    default: i = 2; }
  return i;
}
```

- ☐ -1
- ☐ 0
- ☐ 1
- ☐ 2
- ☐ Gar keinen, da ein Laufzeitfehler auftritt.

d) [1 Punkt] Welche Aussage trifft für die folgende XHTML-Zeile zu:

```
<img src="bild01.jpg" alt="Garfield">Garfield</img>
```

- ☐ Sie ist wohlgeformt und korrekt.
- ☐ Sie ist wohlgeformt, aber nicht korrekt.
- ☐ Sie ist nicht wohlgeformt.

e) [1 Punkt] Welche Aussage trifft für die folgende XHTML-Zeile zu:

```
<em><p>Wichtig: </em> Besprechung heute schon um 15 Uhr!</p>
```

- ☐ Sie ist wohlgeformt und korrekt.
- ☐ Sie ist wohlgeformt, aber nicht korrekt.
- ☐ Sie ist nicht wohlgeformt.

f) [1 Punkt] Welche Aussage trifft für die folgenden XHTML-Zeilen zu:

```
<h1 align=center>Termine</h1>
<hr/>
<p>Zur Zeit keine bekannt.</p>
```

- ☐ Sie sind wohlgeformt und korrekt.
- ☐ Sie sind wohlgeformt, aber nicht korrekt.
- ☐ Sie sind nicht wohlgeformt.

Aufgabe 2: HTML-Programmierung *[13 Punkte]*

a) [5 Punkte] Schreiben Sie eine HTML-Tabelle (keine vollständige Seite), die wie folgt aussieht:

<table border="1">
<tr><th colspan="2">1</th><th>2</th></tr>
<tr><td>3</td><td>4</td><td>5</td></tr>
</table>

Beachten Sie, dass die erste Zeile fett gedruckt ist, dass alle Einträge zentriert sind und dass es sich um eine gerahmte Tabelle handelt!

b) [4 Punkte] Schreiben Sie die `body`-Sektion eines (wohlgeformten und korrekten) XHTML-Dokumentes, das die folgende Gestalt besitzt:

Überschrift 1

Überschrift 2

Dies ist der **erste** Absatz.

Dies ist der **zweite** Absatz.

c) [4 Punkte] Geben Sie CSS-Regeln für folgende Elemente und Darstellungen an:

- Alle Überschriften vom Typ 1 bis 3 sollen in fetter, weißer Schrift vor blauem Hintergrund dargestellt werden.
- Ein `strong`-Element innerhalb von `em` soll in kursiver, roter Schrift dargestellt werden.

(Platz zur Lösung von Aufgabe 2)

Aufgabe 3: Programmieren in Java *[11 Punkte]*

Schreiben Sie ein vollständiges Java-Programm, das eine Tabelle des kleinen Einmaleins am Bildschirm ausgibt. In Zeile i und Spalte j soll also der Wert $i \cdot j$ stehen (zur Erinnerung: *kleines* Einmaleins bedeutet $1 \leq i, j \leq 10$). Die Ausgabe sollte wie folgt aussehen:

```
  1   2   3   4   5   6   7   8   9  10
  2   4   6   8  10  12  14  16  18  20
  3   6   9  12  ...
  .
  .
  .
 10  20  30  40  50  60  70  80  90 100
```

Achten Sie dabei auch darauf, dass die Zahlen wie oben gezeigt untereinander stehen (dazu sind keine zusätzlichen Kenntnisse über Klassenbibliotheken erforderlich)!

Aufgabe 4: Objektorientierte Programmierung *[11 Punkte]*

a) [4 Punkte] Erläutern Sie den Unterschied zwischen dem Überladen und dem Überschreiben einer Methode!

b) [5 Punkte] Gegeben seien die folgenden Interfaces:

```
interface Arithmetik {
  int summe(int x, int y);          // gibt x + y zurueck
  int differenz(int x, int y);      // gibt x - y zurueck
}

interface Produkt {
  int produkt(int x, int y);        // gibt x * y zurueck
}
```

Schreiben Sie eine Klasse `AdamRiese`, die beide Interfaces implementiert!

c) [2 Punkte] Wie muss die Zeile `(?)` des folgenden Hauptprogrammes aussehen, damit das Programm für seine Berechnungen die Klasse `AdamRiese` verwendet?

```
public static void main() {
  Arithmetik berechne = null;
  int a = 2, b = 5, c = 0;
  (?)
  c = berechne.summe(a,b);
  System.out.println("Summe(a,b) = "+c);
  c = berechne.differenz(a,b);
  System.out.println("Differenz(a,b) = "+c);
}
```

(Platz zur Lösung von Aufgabe 4)

Aufgabe 5: Ein-/Ausgabeströme
[10 Punkte]

Die Klasse `BufferedReader` ist aus der Vorlesung bekannt:

```
public class BufferedReader extends Reader {
  public BufferedReader(Reader in) {...}
  ...
}
```

Es soll nun eine Klasse `Eingabe` von `BufferedReader` abgeleitet werden, die gepufferte Zeichenströme sowohl vom Keyboard als auch von einer Datei lesen kann:

```
public class Eingabe extends BufferedReader {...}
```

a) [6 Punkte] Wenn der Konstruktor eines `Eingabe`-Objektes einen String `name` als Parameter erhält, dann soll das Objekt den Eingabestrom aus der Datei `name` lesen. Als `Reader` eignet sich dazu die Klasse `FileReader` mit dem folgenden Konstruktor:

   ```
   public FileReader(String fileName)
          throws FileNotFoundException {...}
   ```

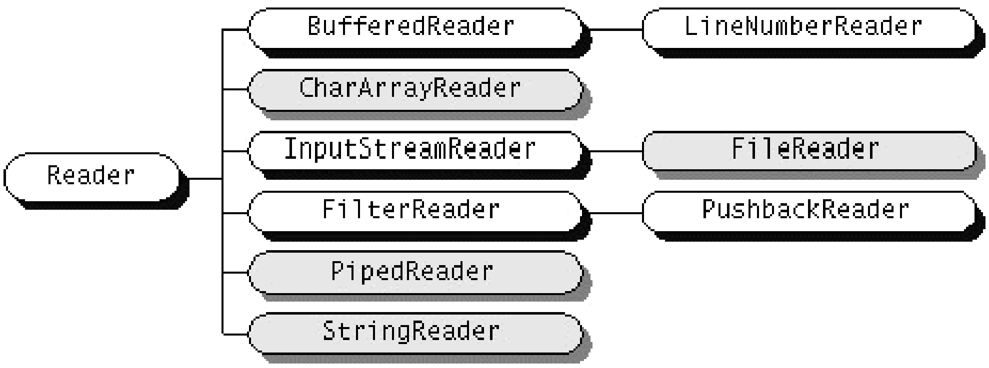
   Implementiere einen geeigneten Konstruktor für die Klasse `Eingabe`!

b) [4 Punkte] Wenn der Konstruktor eines `Eingabe`-Objektes keinen Parameter erhält, dann soll das Objekt den Eingabestrom vom Keyboard lesen. Als `Reader` eignet sich dazu die Klasse `InputStreamReader` mit dem folgenden Konstruktor:

   ```
   public InputStreamReader(InputStream in) {...}
   ```

   Implementiere einen geeigneten Konstruktor für die Klasse `Eingabe`!

Nachfolgende Klassenhierarchie stellt noch einmal das Verhältnis der Klassen `BufferedReader`, `FileReader` und `InputStreamReader` dar.



(Platz zur Lösung von Aufgabe 5)

Aufgabe 6: Studentendatenbank in XML *[12 Punkte]*

Eine Studentendatenbank soll den folgenden Anforderungen genügen:

- Die Datenbank enthält die Datenstammsätze von Studenten, die durch ihren Namen, ihre Matrikelnummer und ihren Studiengang beschrieben werden.
- Jeder Studiengang wird durch den angestrebten Titel und durch eine Liste von Prüfungsfächern charakterisiert, die zu seinem Erreichen bestanden werden müssen.
- Jedes Prüfungsfach hat einen Namen sowie als Attribute eine eindeutige Nummer und eine Credit-Zahl $\in \{3, 6, 9\}$.

a) [7 Punkte] Schreiben Sie eine DTD für diese Studentendatenbank.

b) [5 Punkte] Nehmen Sie an, dass Ihnen die DTD aus Teilaufgabe a) in einer Datei `studi-db.dtd` zur Verfügung steht. Schreiben Sie ein vollständiges XML-Dokument, das eine Studentendatenbank mit dem Datensatz für einen Studenten enthält (ein Prüfungsfach genügt).

(Platz zur Lösung von Aufgabe 6)