Universität Mannheim
Fakultät für Mathematik und Informatik
Dr. Heinz Kredel

Teilprüfung
Software- und Internettechnologie
Programmierkurs 1
Wintersemester 2005/2006

Name: .....
Vorname: .....
Matrikel-Nr.: .....
Studienfach: .....

Hinweise:

1. Überprüfen Sie die Klausur auf Vollständigkeit (**11** einseitig bedruckte Seiten).
2. Tragen Sie die Lösungen direkt in die Klausur ein. Benutzen Sie ggf. auch die Rückseiten der Aufgabenblätter.
3. Unterschreiben Sie die Klausur auf dem letzten Blatt.
4. Zugelassene Hilfsmittel: keine
5. Die Bearbeitungszeit beträgt 66 Minuten.

Aufgabe	max. Punktzahl	erreichte Punktzahl
1	12	
2	16	
3	10	
4	10	
5	18	
Summe	66	

Aufgabe 1: Verständnisfragen [12 Punkte]

a) [2 Punkte] Ist jedes wohlgeformte XHTML-Dokument auch gültig? Begründen Sie `kurz` anhand eines Beispiels.

b) [2 Punkte] Warum verlangt der HTML-Standard für `img`-Elemente die Angabe einer alternativen textuellen Beschreibung in Form des `alt`-Attributs?

c) [3 Punkte] Wandeln Sie in folgendem Java-Codefragment die `for`-Schleife in eine äquivalente `while`-Schleife um.

```
for (int i=0 ; i < 10 ; i++){
  System.out.println(i);
  i++;
}
```

d) [3 Punkte] Betrachten Sie folgende fehlerhafte Java-Funktion, die alle Einträge eines `int`-Felds in umgekehrter Reihenfolge ausgeben soll:

```
void revPrintFeld(int[] a){
  int i = a.length-1;
  while (i > 0)
    System.out.println(a[--i]);
}
```

Welche Ausgabe bewirken folgende Anweisungen?

```
int[] a = {2,3,4,5};
revPrintFeld(a);
```

Wie muss die Funktion verändert werden, damit sie das Gewünschte leistet?

e) [2 Punkte] Gegeben sei die folgende Java-Klasse `CheckSense`.

```
public class CheckSense{

    public static boolean check(String ans[]){
        for (int i=0 ; i < ans.length ; i++){
            if (ans[i]=="42") return true;
        }
        return false;
    }

    public static void main(String args[]){
        System.out.println(check(args));
        System.out.println(check(new String[] {"4", "2", "42", "3", "7"}));
    }
}
```

Welche Ausgabe produziert der Programmaufruf `java CheckSense 4 2 42 3 7` ?

Aufgabe 2: XHTML und CSS [16 Punkte]

a) [11 Punkte] Schreiben Sie in Anlehnung an die folgende Abbildung eine gültige XHTML 1.0 Strict Formularseite, mit der man Flüge von Frankfurt nach Toronto buchen kann.

Genauer soll das Formular

- eine passende Überschrift enthalten,
- die in der Abbildung aufgeführten Flugverbindungen in einer Tabelle darstellen, in der die Tabelleneinträge in den Zellen zentriert sind,
- mit einer Checkbox abfragen, ob der Passagier einen Gangplatz wünscht,
- eine entsprechend beschriftete Textarea für Bemerkungen enthalten, die 4 Zeilen hoch und 50 Zeichen breit ist,
- einen mit `Buchen` beschrifteten Submit-Button enthalten
- sowie die Formulardaten an das CGI-Programm `http://www.lufthansa.de/cgi-bin/flug.cgi` übermitteln.

Fügen Sie Ihren HTML-Code in die folgende Datei ein:

```
<?xml version="1.0" encoding="iso-8859-1"?>
<!DOCTYPE html
     PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="de" lang="de">
  <head>
      <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" />
      <title>Flugverbindungen Frankfurt-Toronto</title>
  </head>
```

```
<!-- Fortsetzung der Formularseite -->
```

```
</html>
```

b) [3 Punkte] Geben Sie eine CSS2 style-rule an, mit der man orange als Hintergrundfarbe einer einzelnen Tabellenzelle definieren kann und beschreiben Sie kurz **eine** Möglichkeit, diese style-rule in die Formularseite aus Aufgabe a) zu integrieren.

c) [2 Punkte] Sind die folgenden `style` Elemente äquivalent (kurze Begründung)?

```
<style>
  ul { font-weight : bold }
</style>
```

```
<style>
  li.nav { font-weight : bold }
</style>
```

Aufgabe 3: Java-Programmierung [10 Punkte]

Schreiben Sie eine **vollständige** Java-Application `MuSig`, der über die Kommandozeile ganze Zahlen $a_0, \ldots, a_{n-1}$ übergeben werden. Das Programm soll den Mittelwert μ und die Varianz σ^2 der übergebenen Zahlen bestimmen und ausgeben. Dabei gilt

$$\mu = \frac{1}{n}\sum_{i=0}^{n-1} a_i \text{ und } \sigma^2 = \frac{1}{n}\sum_{i=0}^{n-1}(a_i - \mu)^2 \ .$$

Beispiel:

```
java MuSig 2 4 6 8 10
Der Mittelwert ist 6.0.
Die Varianz ist 8.0.
```

Gehen Sie davon aus, dass der Benutzer als Argumente nur ganze Zahlen eingibt, und reagieren Sie auf nicht sinnvolle Eingaben mit einer geeigneten Fehlermeldung.

Aufgabe 4: Objektorientierte Grundlagen [10 Punkte]

a) [2 Punkte] Erläutern Sie **kurz** an einem Beispiel den Unterschied zwischen Klassen und Objekten.

b) [3 Punkte] Gegeben seien die Java-Klassen

```
public class A {
  public double add(double a, double b) {
    System.out.println("A.add");
    return a+b;
  }
  public double divide(double a, double b) {
    System.out.println("A.divide");
    return a/b;
  }
}

public class B extends A {
  public int divide(int a, int b) {
    System.out.println("B.divide");
    return a/b;
  }
}
```

sowie die Deklarationen

```
int w=5,x=2;
double y=5,z=2;
B b = new B();
```

Welche Ausgabe produzieren die folgenden Anweisungen?

```
System.out.println(b.add(w,x));
System.out.println(b.divide(w,x));
System.out.println(b.divide(y,z));
```

c) [3 Punkte] Zeichnen Sie ein UML-Diagramm für die folgende Java-Klasse.

```
public class Auto extends Fahrzeug {
  private int leistung;
  private int anzahlTueren;

  public boolean starte() { /* ... */ }
}
```

d) [2 Punkte] Betrachten Sie das Java-Interface `Tutor`

```
public interface Tutor {
  public void korrigiere(Uebungsblatt blatt);
  public void fuehreDurch(Tutorium tutorium);
}
```

und die Java-Klasse `StudentischerTutor`

```
public abstract class StudentischerTutor implements Tutor {
  /* (1) */
}
```

Welcher Code muss an der mit `/* (1) */` markierten Stelle **mindestens** eingefügt werden, damit die Klasse übersetzt wird?

Aufgabe 5: Objektorientierte Programmierung [18 Punkte]

Gegeben sei die folgende Java-Klasse `Konto` zur Verwaltung eines Bankkontos.

```
import java.util.List;
import java.util.LinkedList;
import java.util.Date;

public class Konto {

    /* letzte vergebene Kontonummer */
    private static long letzteNummer = 0;

    /* Kontonummer */
    protected final long nummer;

    /* aktueller Kontostand */
    protected double kontostand;

    /* minimal moeglicher Kontostand */
    protected double kreditlimit;

    /* Liste aller Buchungen auf dem Konto */
    protected List<Buchung> buchungen;

    public Konto(double kreditlimit) { /* ... */ }

    public void fuehreAus(Buchung b)
        throws InsufficientBalanceException { /* ... */ }

    public List<Buchung> getKontoauszug(Date von, Date bis) { /* ... */ }

} /* Konto */
```

Eine Buchung auf einem Bankkonto wird verwaltet durch die Java-Klasse `Buchung`:

```
import java.util.Date;

public class Buchung {

    /* Konto, auf dem die Buchung stattfindet */
    protected Konto kto;

    /* Betrag der Kontobewegung */
    protected double betrag;

    protected Date ausfuehrungsdatum;

    protected String buchungstext;

    public Buchung(Konto kto, double betrag, String buchungstext) {
      this.kto = kto;
      this.betrag = betrag;
      this.buchungstext = buchungstext;
    }

    public double getBetrag() { return betrag; }

    public void ausgefuehrt() { ausfuehrungsdatum = new Date(); }

    public Date getDatum() { return ausfuehrungsdatum; }

} /* Buchung */
```

a) [3 Punkte] Geben Sie eine Implementation des Konstruktors der Klasse `Konto` an, der die Attribute geeignet initialisiert. Die Konten sollen fortlaufend nummeriert werden.

b) [1 Punkt] Geben Sie ein Java-Codefragment an, das ein `Konto`-Objekt mit dem Kreditlimit -1000 erzeugt.

c) [8 Punkte] Implementieren Sie die Methode `fuehreAus` der Klasse `Konto`, die die übergebene Buchung auf dem Konto in folgender Weise durchführt:

Falls durch die Buchung der Kontostand unter das Kreditlimit sinken würde, soll der Vorgang abgebrochen und eine `InsufficientBalanceException` geworfen werden. Andernfalls wird der Kontostand um den Buchungsbetrag verändert, die Buchung der Liste der durchgeführten Buchungen hinzugefügt und die Buchung als ausgeführt gekennzeichnet.

Geben Sie ebenfalls eine Implementation der `InsufficientBalanceException` an.

d) [2 Punkte] Geben Sie ein Java-Codefragment an, das eine Bareinzahlung in Höhe von 50 auf das unter b) erzeugte Konto durchführt und dabei eventuell auftretende `InsufficientBalanceException`s abfängt und geeignet behandelt.

e) [4 Punkte] Implementieren Sie die Methode `getKontoauszug` der Klasse `Konto`, die eine Liste aller Buchungen zurückgibt, die zwischen den Daten `von` und `bis` (einschließlich) auf dem Konto durchgeführt wurden.

Hinweise:

- Für ein `Date d` gibt `d.after(Date otherDate)` genau dann `true` zurück, wenn `d` später als `otherDate` ist. Analog liefert `d.before(Date otherDate)` genau dann `true`, wenn `d` früher als `otherDate` ist, und `d.equals(Date otherDate)` genau dann `true`, wenn `d` und `otherDate` gleich sind.
- Benutzen Sie einen geeigneten `Iterator`.