Universität Mannheim
Fakultät für Mathematik und Informatik
Dr. Heinz Kredel

Teilprüfung
Software- und Internettechnologie
Programmierkurs 1
Wintersemester 2004/2005

Name: .....

Vorname: .....

Matrikel-Nr.: .....

Studienfach: .....

Hinweise:

1. Überprüfen Sie die Klausur auf Vollständigkeit (**13** 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. Lösungen auf farbigem Konzeptpapier werden **nicht** bewertet.
5. Zugelassene Hilfsmittel: nicht programmierbarer Taschenrechner
6. Die Bearbeitungszeit beträgt 66 Minuten.

Aufgabe	max. Punktzahl	erreichte Punktzahl
1	14	
2	16	
3	10	
4	18	
5	8	
Summe	66	

Aufgabe 1: Verständnisfragen [14 Punkte]

a) [2 Punkte] Erläutern Sie **kurz** zwei Unterschiede zwischen HTML 4.01 und XHTML 1.0.

b) [2 Punkte] Erläutern Sie **kurz** den Zusammenhang zwischen HTML-Elementen und HTML-Tags.

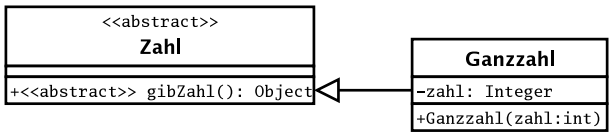
c) [3 Punkte] Wandeln Sie in folgendem Java-Codefragment die `for`-Schleife in eine äquivalente `while`-Schleife um, die keine `break`-Anweisung verwendet.

```
Iterator it;
/* ... */
for (;;){
   if (!it.hasNext()) break;
   System.out.println(it.next());
}
```

d) [3 Punkte] Korrigieren Sie die Fehler in folgender Java-Methode, die alle Einträge des übergebenen Felds `a` ausgeben soll:

```
public void printArray(int[] a){
  for (int i=1 ; i <= a.length() ; i++){
    System.out.println(a[i]);
  }
}
```

e) [4 Punkte] Geben Sie eine Java-Implementation für die durch das folgende UML-Diagramm beschriebene Klasse `Ganzzahl` an, die eine ganze Zahl repräsentieren soll.



Aufgabe 2: XHTML und CSS [16 Punkte]

a) [6 Punkte] Schreiben Sie eine gültige *XHTML 1.0 Strict* Formularseite, mit der man sich für den Mobile Business Workshop an der Uni Mannheim am 22.03.2005 hätte anmelden können. Das Formular soll

- den Namen, den Vornamen und die Firma des Teilnehmers jeweils in einem Texteingabefeld abfragen, das Platz für 30 Zeichen in der Anzeige hat,
- über eine Checkbox abfragen, ob der Teilnehmer ein vegetarisches Mittagessen wünscht,
- einen mit `Abschicken` beschrifteten Submit-Button enthalten
- und die Formulardaten mit `GET` an das CGI-Programm `http://www.informatik.uni-mannheim.de/cgi-bin/m-business.cgi` übermitteln.

Fügen Sie Ihren Code in die folgende Datei ein:

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









</html>
```

b) [6 Punkte] Schreiben Sie eine *XHTML 1.0 Strict* Seite, die einen Ausschnitt aus dem Vortragsprogramm des Workshops mit Hilfe einer zweispaltigen Tabelle darstellt, wobei

- in der linken Spalte mit der Überschrift *Uhrzeit* die Uhrzeiten und in der rechten Spalte mit der Überschrift *Vortrag* die Vortragstitel sowie der Referent stehen und
- die Vortragstitel Hyperlinks sind, die auf die jeweiligen Vortragsfolien verweisen.

Nehmen Sie die folgenden Programmpunkte in Ihre Tabelle auf:

- 09.00 Uhr: Prof. Dr. Wolfgang Effelsberg, Uni Mannheim: *Begrüßung und Einführung*, Folien unter `http://www.uni-mannheim.de/pi4/folien.html`
- 9.15 Uhr: Dr. Udo Urbanek, SAP AG: *What works - What doesn't Work in Mobile Enterprise Business*, Folien unter `http://www.sap.com/pub/slides.html`
- 10.00 Uhr: *Kaffeepause*

Fügen Sie Ihren Code in die folgende Datei ein:

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









</html>
```

c) [2 Punkte] Geben Sie CSS2 style-rules an, um in der Programmseite des Workshops

- alle Hyperlinks in Fettschrift und
- die Tabellenüberschriften und -einträge zentriert in den Tabellenzellen darzustellen.

d) [2 Punkte] Beschreiben Sie kurz **eine** Möglichkeit, diese style-rules in die Seite zu integrieren.

Aufgabe 3: Java-Programmierung [10 Punkte]

Schreiben Sie eine **vollständige** Java-Application `MinMaxAvg`, der über die Kommandozeile ganze Zahlen aus dem Intervall $[-100, 100]$ übergeben werden. Das Programm soll die kleinste und die größte der übergebenen Zahlen sowie den Durchschnitt der übergebenen Zahlen bestimmen und am Bildschirm ausgeben.

Beispiel:

```
> java MinMaxAvg 10 15 -3 4 17
min = -3
max = 17
avg = 8.6
```

Sie können davon ausgehen, dass die übergebenen Zahlen korrekt formatiert sind.

Aufgabe 4: Objektorientierung in Java [18 Punkte]

Die folgende Klasse verwaltet einen Angestellten.

```
public class Angestellter{

    private static int aktPersNr; /* zuletzt vergebene Personalnummer */

    private String name;
    private final int personalnummer;
    private int resturlaub; /* Anzahl der verbleibenden Urlaubstage */
    protected int grundgehalt;
    private Vorgesetzter vorgesetzter; /* direkter Vorgesetzter */

    /* erzeugt einen neuen Angestellten */
    public Angestellter(String name, Vorgesetzter dirVor, int grundgehalt,
                        int urlaub){
        /* ... */
    }

    /* gibt genau dann true zurueck, wenn der Urlaub von anzahlTage Tagen
     * erfolgreich beantragt wurde.
     */
    public boolean beantrageUrlaub(int anzahlTage){ /* ... */ }

    public int getJahresgehalt() { return grundgehalt; }

    public int getResturlaub() { return resturlaub; }

    public Vorgesetzter getVorgesetzter() { return vorgesetzter; }
}
```

Das Interface `Vorgesetzter` ist wie folgt definiert:

```
public interface Vorgesetzter{

    /* gibt genau dann true zurueck, wenn dieser Vorgesetzte dem
     * Angestellten m anzahlTage Urlaub genehmigt.
     */
    public boolean genehmigeUrlaub(Angestellter m, int anzahlTage);

} /* Vorgesetzter */
```

a) [4 Punkte] Implementieren Sie den Konstruktor der Klasse `Angestellter` in folgender Weise:

- Falls eine `null`-Referenz als direkter Vorgesetzter übergeben wird, wirft der Konstruktor eine `NullPointerException`.
- Die Angestellten erhalten eine fortlaufende Personalnummer.

b) [2 Punkte] Geben Sie ein Java-Codefragment an, in dem ein `Angestellter`-Objekt für Klaus Schulz mit einem Grundgehalt von 32000 und 18 Tagen Resturlaub erzeugt wird. Herr Schulz arbeitet in der Abteilung von Frau Schmidt, die durch das `Vorgesetzter`-Objekt `schmidt` repräsentiert wird.

c) [2 Punkte] Implementieren Sie die Methode `beantrageUrlaub`, indem Sie genau dann `true` zurückgeben, wenn der direkte Vorgesetzte den Urlaub genehmigt.

d) [10 Punkte] Schreiben Sie eine `Angestellter`-Subklasse `LeitenderAngestellter`, die das `Vorgesetzter`-Interface implementiert und folgende Eigenschaften besitzt:

- Das Jahresgehalt eines leitenden Angestellten setzt sich zusammen aus dem Angestellten-Grundgehalt und einer zusätzlichen Stellenzulage, die bei der Objekterzeugung festgelegt wird.
- Ein leitender Angestellter genehmigt seinen Mitarbeitern nach erfolgreicher Prüfung des Resturlaubsanspruchs alle Urlaubsanträge für bis zu 21 Tage. Anträge von Angestellten, die einen anderen direkten Vorgesetzten als ihn selbst haben, lehnt er unbesehen ab.

Aufgabe 5: Java Collections Framework [8 Punkte]

Während eine `Map` einem Schlüssel höchstens einen Wert zuordnet, können in einer `MultiMap` einem Schlüssel beliebig viele Elemente zugeordnet werden.

Analog zum Interface `java.util.Map` sei das folgende Java-Interface einer `MultiMap` gegeben:

```
public interface MultiMap{

    /** gibt genau dann true zurueck, wenn diese MultiMap ein mapping fuer
     *  den uebergebenen key enthaelt.
     */
    public boolean containsKey(Object key);


    /** gibt die Werte zurueck, die dem uebergebenen key in dieser MultiMap
     *  zugeordnet sind. Gibt null zurueck, falls fuer den key in dieser
     *  MultiMap keine Werte gespeichert sind.
     */
    public java.util.Set get(Object key);


    /** fuegt den Werten, die dem uebergebenen key in dieser MultiMap
     *  zugeordnet sind, den uebergebenen value hinzu
     */
    public void put(Object key, Object value);
}
```

Implementieren Sie eine Klasse `HashMultiMap`, die mit Hilfe von `java.util.HashSet`s und einer `java.util.HashMap` das `MultiMap` Interface implementiert. Falls benötigt, finden Sie auf den nächsten Seiten Auszüge aus den Dokumentationen der Klassen `java.util.HashMap` und `java.util.HashSet`.

`java.util.HashMap` Constructor Summary

`HashMap()`	Constructs an empty HashMap with the default initial capacity (16) and the default load factor (0.75).
`HashMap(int initialCapacity)`	Constructs an empty HashMap with the specified initial capacity and the default load factor (0.75).
`HashMap(int initialCapacity, float loadFactor)`	Constructs an empty HashMap with the specified initial capacity and load factor.
`HashMap(Map m)`	Constructs a new HashMap with the same mappings as the specified Map.

`java.util.HashMap` Method Summary

`void clear()`	Removes all mappings from this map.
`Object clone()`	Returns a shallow copy of this HashMap instance: the keys and values themselves are not cloned.
`boolean containsKey(Object key)`	Returns true if this map contains a mapping for the specified key.
`boolean containsValue(Object value)`	Returns true if this map maps one or more keys to the specified value.
`Set entrySet()`	Returns a collection view of the mappings contained in this map.
`Object get(Object key)`	Returns the value to which the specified key is mapped in this identity hash map, or null if the map contains no mapping for this key.
`boolean isEmpty()`	Returns true if this map contains no key-value mappings.
`Set keySet()`	Returns a set view of the keys contained in this map.
`Object put(Object key, Object value)`	Associates the specified value with the specified key in this map.
`void putAll(Map m)`	Copies all of the mappings from the specified map to this map These mappings will replace any mappings that this map had for any of the keys currently in the specified map.
`Object remove(Object key)`	Removes the mapping for this key from this map if present.
`int size()`	Returns the number of key-value mappings in this map.
`Collection values()`	Returns a collection view of the values contained in this map.

`java.util.HashSet` Constructor Summary

`HashSet()`	Constructs a new, empty set; the backing HashMap instance has default initial capacity (16) and load factor (0.75).
`HashSet(Collection c)`	Constructs a new set containing the elements in the specified collection.
`HashSet(int initialCapacity)`	Constructs a new, empty set; the backing HashMap instance has the specified initial capacity and default load factor, which is 0.75.
`HashSet(int initialCapacity, float loadFactor)`	Constructs a new, empty set; the backing HashMap instance has the specified initial capacity and the specified load factor.

`java.util.HashSet` Method Summary

`boolean add(Object o)`	Adds the specified element to this set if it is not already present.
`void clear()`	Removes all of the elements from this set.
`Object clone()`	Returns a shallow copy of this HashSet instance: the elements themselves are not cloned.
`boolean contains(Object o)`	Returns true if this set contains the specified element.
`boolean isEmpty()`	Returns true if this set contains no elements.
`Iterator iterator()`	Returns an iterator over the elements in this set.
`boolean remove(Object o)`	Removes the specified element from this set if it is present.
`int size()`	Returns the number of elements in this set (its cardinality).