Fakultät IV
Elektrotechnik/Informatik

Klausur Einführung in die Informatik II (Technikorientierung)

26. Februar 2007

Name: .....

Matr.-Nr. .....

Bearbeitungszeit: 120 Minuten

Bewertung

Aufgabe	Punkte	Erreichte Punkte
1	5	
2	6	
3	6	
4	14	
5	7	
6	9	
7	6	
8	5	
9	2	
Summe	60	

Spielregeln:

- Benutzen Sie für die Lösung der Aufgaben **nur** das mit diesem Deckblatt ausgeteilte Papier. Lösungen, die auf anderem Papier geschrieben werden, können **nicht** bewertet werden. Schreiben Sie ihre Lösung auch auf die Rückseiten der Blätter; benötigen Sie für eine Lösung mehr als ein Blatt, finden Sie am Ende der Klausur Leerblätter. Zusätzliches Papier können Sie von den Tutoren bekommen.
- Tragen Sie jetzt (vor Beginn der eigentlichen Bearbeitungszeit!) auf **allen** Blättern ihren Namen und ihre Matrikelnummer ein.
- Schreiben Sie deutlich! Unleserliche oder mehrdeutige Lösungen können nicht gewertet werden.
- Schreiben Sie **nicht** mit Bleistift und **nicht** mit rotem oder grünem Stift (das sind die Farben für die Korrektur).
- Lesen Sie die Aufgaben jeweils bis zum Ende durch; oft gibt es hilfreiche Hinweise!
- Erlaubt sind nur die beiden beidseitig beschrifteten A4-Blätter und blaue oder schwarze Kugelschreiber bzw. Füller. Die Benutzung aller anderen Hilfsmittel ist untersagt.

Viel Erfolg!

Aufgabe 1 (5 Punkte) Allgemeiner Teil.

Kreuzen Sie bei den folgenden Fragen die richtige Antwort an.

Beachten Sie folgende Punkteregelung:

- Für jede richtige Antwort erhalten Sie eine Punkteinheit.
- Für jede **falsche Antwort** wird Ihnen eine Punkteinheit **abgezogen**.
- Sie können dabei jedoch keine negativen Punkte erzielen, auch wenn die falschen Antworten die richtigen überwiegen.

1. Wie viele Referenzen kann es auf ein einziges Objekt geben?

   ◯ Nur eine.
   ◯ Zwei. Eine vom Aufrufer und eine von der aufgerufenen Methode.
   ◯ Drei. Die ursprüngliche Referenz und je eine Referenz für den aktuellen Parameter und den Rückgabewert der aufgerufenen Methode.
   ◯ Es kann eine beliebige Anzahl von Referenzen geben, die in einer beliebigen Anzahl von Variablen und Parametern (des entsprechenden Typs) gehalten werden.

2. Eine neue Klasse `Milch` soll von der Superklasse `Getraenk` abgeleitet werden. Welche der folgenden Anweisungen ist syntaktisch korrekt?

   ◯ `class Milch isa Getraenk{ . . . }`
   ◯ `class Milch extends Getraenk{ . . . }`
   ◯ `class Milch implements Getraenk{ . . . }`
   ◯ `class Milch defines Getraenk{ . . . }`

3. Eine Klasse `Tier` hat eine Subklasse `Saeugetier`. Welche der folgenden Aussagen ist wahr?

   ◯ Auf Grund der Einfachvererbung kann `Saeugetier` keine Subklassen haben.
   ◯ Auf Grund der Einfachvererbung kann `Saeugetier` keine andere Superklasse haben als `Tier`.
   ◯ Auf Grund der Einfachvererbung kann `Tier` nur eine Subklasse haben.
   ◯ Auf Grund der Einfachvererbung kann `Saeugetier` keine Geschwisterklassen haben.

4. Erbt eine Subklasse sowohl Variablen als auch Methoden?

   ◯ Nein, nur die Variablen werden geerbt.
   ◯ Nein, nur Methoden werden geerbt.
   ◯ Ja, sowohl Variablen als auch Methoden werden geerbt.
   ◯ Ja, aber es wird entweder nur das Eine oder das Andere geerbt.

5. Kann eine abstrakte Methode in einer nicht-abstrakten Klasse definiert sein?

   ◯ Nein, wenn eine Klasse eine abstrakte Methode definiert, muss die Klasse selbst abstrakt sein.
   ◯ Nein, nur Klassen sind abstrakt, nicht Methoden.
   ◯ Ja, eine Methode kann in einer beliebigen Superklasse als abstrakt deklariert sein, solange sie die Subklassen ebenfalls als abstrakt deklarieren.
   ◯ Ja, es gibt keine Beschränkungen wo abstrakte Methoden definiert werden können.

6. Was muss wahr sein, wenn eine Subklasse einer abstrakten Superklasse die abstrakten Methoden der Superklasse nicht überschreibt?

   ◯ Das ist immer ein Fehler.
   ◯ Die Subklasse muss selbst als abstrakt deklariert werden.
   ◯ Subklassen sind automatisch nicht-abstrakt, also ist das korrekt.
   ◯ Die Superklasse ist ein Fehler.

7. Welche Auswirkung hat es, wenn ein Klassenelement (Variable oder Methode) den Modifizierer `private` hat?

   ◯ Wenn ein Klassenelement als `private` deklariert ist, kann es nur an einer Stelle im Programm verwendet werden.

   ◯ Wenn ein Klassenelement als `private` deklariert ist, kann es nur von Methoden verwendet werden, die Mitglied der Klasse sind.

   ◯ Wenn ein Klassenelement als `private` deklariert ist, kann es nur von anderen `private` Mitgliedern anderer Klassen verwendet werden.

   ◯ Wenn ein Klassenelement als `private` deklariert ist, gibt es von ihm nur eine Instanz, ganz egal wie viele Objekte instantiiert sind.

8. Welchen Aufwand hat Quicksort?

   ◯ im Mittel $\mathcal{O}(NlogN)$

   ◯ im Mittel $\mathcal{O}(3N + 4)$

   ◯ im Mittel $\mathcal{O}(N^2)$

   ◯ im Mittel $\mathcal{O}(2^N)$

9. Ein Sortierverfahren heißt *stabil*,

   ◯ wenn es keinen zusätzlichen Speicherplatz benötigt.

   ◯ wenn es linearen Aufwand hat.

   ◯ wenn es die relative Anordnung von Elementen mit dem gleichen Sortierschlüssel erhält.

   ◯ wenn es zusätzlichen Speicherplatz benötigt.

10. In einem *balancierten* Baum

    ◯ hat jeder Knoten maximal drei Kindknoten.

    ◯ hat jeder Knoten maximal zwei Kindknoten.

    ◯ gibt es eine gerade Anzahl von Knoten.

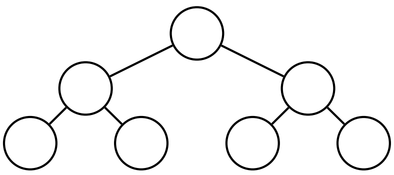
    ◯ unterscheiden sich die Längen der einzelnen Pfade des Baumes höchstens um 1.

Aufgabe 2 (6 Punkte) Heapsort.

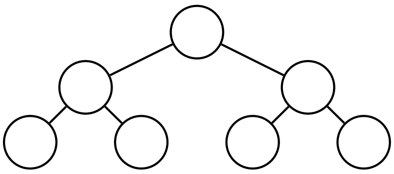
Stellen Sie den Verlauf einer Sortierung durch Heapsort dar.
Kennzeichnen Sie dabei:

- **einmalig**, welche Indizes des Arrays welchen Knoten im Heap entsprechen.
- zu tauschende Elemente.
- im Heap nicht mehr zu berücksichtigende Elemente (durch weglassen) und den bereits sortieren Bereich im Array (durch anstreichen).

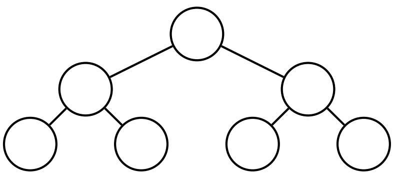
Hinweis: Es müssen nicht alle Vorgaben verwendet werden, wenn das Array bereits sortiert ist.



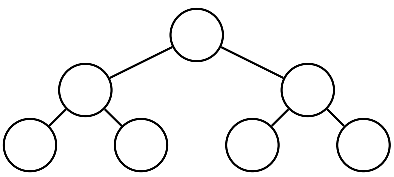
0	1	2	3	4	5
6	4	8	3	7	5



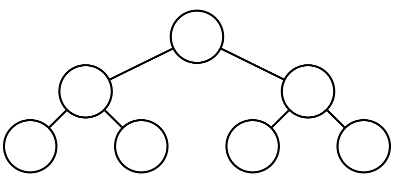
0	1	2	3	4	5



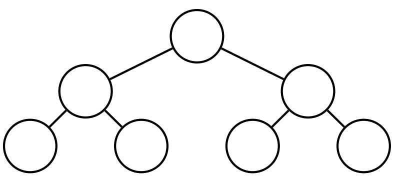
0	1	2	3	4	5



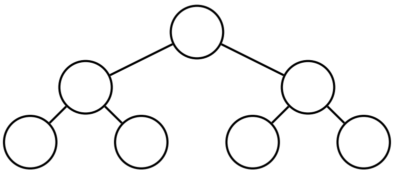
0	1	2	3	4	5



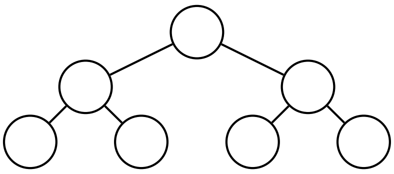
0	1	2	3	4	5



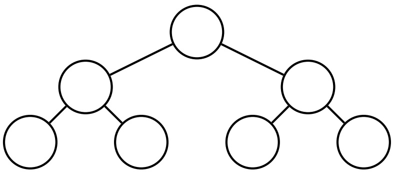
0	1	2	3	4	5



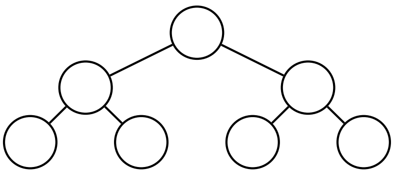
0	1	2	3	4	5



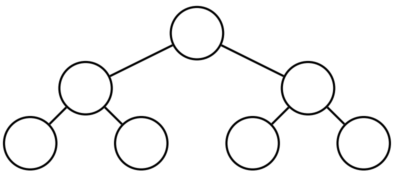
0	1	2	3	4	5



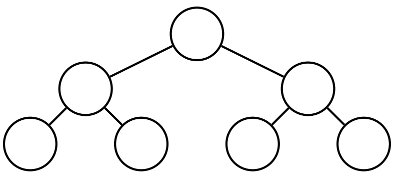
0	1	2	3	4	5



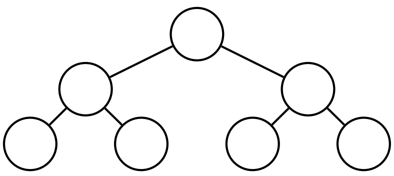
0	1	2	3	4	5



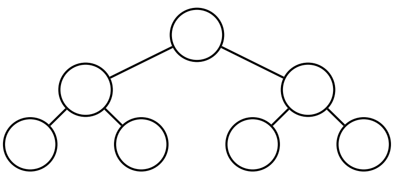
0	1	2	3	4	5



0	1	2	3	4	5



0	1	2	3	4	5



0	1	2	3	4	5

Aufgabe 3 (6 Punkte) Java - Vererbung.

Betrachten Sie die folgenden Java - Klassen.

```
1   class A{
2           String string;
3           A(){ string     = "AAA"; }
4           String getString(){ return string; }
5           void setString(String s){ string = s; }
6   }
7   class B extends A{
8           String string;
9           B(){ string     = "BBB"; }
10          String getString(){ return string; }
11          void setString(String s){ string = s; super.string = "auch " + s; }
12  }
13  class C extends A{
14          String string;
15          C(){ string     = "CCC"; }
16          String getString(){ return string; }
17          void setString(String s){ string = s; }
18  }
```

Welche Ausgabe wird durch die Schleifen an den Punkten (1), (2), (3) und (4) beim Ausführen des folgenden Programmcodes erzeugt? Tragen Sie die Ausgaben in den unten angegebenen Bereichen ein!

```
1   A[] a = {new A(), new B(), new C()};
2   //(1)
3   for(int i = 0; i < a.length; i++){
4           Terminal.println(a[i].getString());
5   }
6   //(2)
7   for(int i = 0; i < a.length; i++){
8           Terminal.println(a[i].string);
9   }
10  for(int i = 0; i < a.length; i++){
11          a[i].setString("neu");
12  }
13  //(3)
14  for(int i = 0; i < a.length; i++){
15          Terminal.println(a[i].getString());
16  }
17  //(4)
18  for(int i = 0; i < a.length; i++){
19          Terminal.println(a[i].string);
20  }
```

Ausgabe an Position (1)	Ausgabe an Position (2)
Ausgabe an Position (3)	Ausgabe an Position (4)

Aufgabe 4 (14 Punkte) Listen - Stack.

In dieser Aufgabe soll ein Datentyp `Lagerstapel` erstellt werden. Es soll damit ein Kistenstapel in einem Lager repräsentiert werden, für den eine maximale Höhe festgelegt werden kann. Sie dürfen für die Lösung dieser Aufgabe kein Array verwenden, sondern sollen die Klassen `LStapel` und `SElement` entwerfen.
Für die Kisten müssen Sie folgende Java-Klasse verwenden, die Sie nicht verändern dürfen.

```
1  class Kiste{
2          int ID;
3          int gewicht;
4          Kiste(int ID, int gewicht){
5                  this.ID = ID;
6                  this.gewicht = gewicht;
7          }
8  }
```

1. (2 Punkte) Schreiben Sie die Klasse `SElement`, die jeweils eine auf dem Stapel vorhandene Kiste repräsentiert. Definieren Sie einen geeigneten Konstruktor.

2. (12 Punkte) Schreiben Sie die Klasse `LStapel`, die den Kistenstapel repräsentiert.

   - Überlegen Sie sich, welche Attribute die Klasse LStapel benötigt.
   - In der Klasse `LStapel` soll es einen Konstruktor geben, der einen leeren Stapel erzeugt und dabei die maximale Höhe für den anzulegenden Stapel mit dem als Parameter übergebenen Wert initialisiert.
   - In der Klasse `LStapel` soll es eine Methode `boolean istLeer()` geben, die `true` liefert, wenn keine Kiste auf dem Stapel liegt und sonst `false`.
   - In der Klasse `LStapel` soll es eine Methode `boolean rauf(Kiste k)` geben, die die als Parameter angegebene Kiste oben auf den Stapel stellt. Dies darf jedoch nur dann erfolgen, wenn die maximale Höhe noch nicht erreicht wurde. Der Rückgabewert der Funktion zeigt dabei an, ob es möglich war, die Kiste oben auf den Stapel zu stellen.
   - In der Klasse `LStapel` soll es eine Methode `Kiste runter()` geben, die die oberste Kiste vom Stapel entfernt und als Rückgabewert liefert. Wenn sich keine Kiste auf dem Stapel befindet, soll dies durch den Rückgabewert `null` angezeigt werden.
   - In der Klasse `LStapel` soll es eine Methode `Kisten[] runter(int anzahl, int gewicht)` geben, die vom Stapel Kisten herunternimmt. Heruntergenommen werden Kisten immer von oben! Dabei wird durch den

Parameter `anzahl` angegeben, wieviele Kisten heruntergenommen werden sollen. Dabei darf das Gesamtgewicht der heruntergenommenen Kisten den durch den Parameter `gewicht` angegebenen Wert nicht überschreiten. Sollte dies jedoch der Fall sein, werden weniger Kisten vom Stapel entfernt. Sollten auf dem Stapel weniger Kisten vorhanden sein, als durch den Parameter `anzahl` angefordert, soll der Stapel vollständig abgeräumt werden. Bei einem leeren Stapel soll auch durch diese Methode der Wert `null` zurückgegeben werden.

- In der Klasse `LStapel` soll es eine Methode `int suche(int ID)` geben, die prüft, ob sich auf dem Stapel eine Kiste mit der durch den Parameter `ID` angegebenen Identifikationsnummer (ID) befindet. Rückgabewert ist die Position der Kiste, wobei zu beachten ist, dass die oberste Kiste die Nummer 1 hat. Ist die Kiste nicht auf dem Stapel, soll dies durch den Wert -1 angezeigt werden.

Aufgabe 5 (7 Punkte) Bäume.

Betrachten Sie die Klasse `TreeNode`

```
class TreeNode {
    private int data;
  private TreeNode left, right;
  TreeNode(int data, TreeNode l, TreeNode r){
  this.data = data;
  this.left = l;
  this.right = r;
  }
}
```

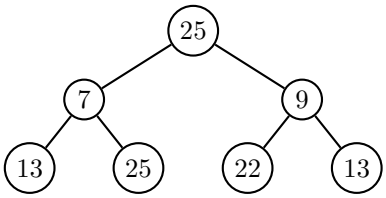
und folgenden (unvollständigen) Programmcode!

```
...
public static void main(String args[]){
  ...
  TreeNode root = new TreeNode(4, null, null);
  ...
  int x = 7;
  boolean b = root.suche(x);
  int hoehe = root.berechneHoehe();
  ...
```

1. (3 Punkte) Definieren Sie in der Klasse `TreeNode` die fehlende Methode `boolean suche(int a)`, die `true` liefert, wenn die Zahl a in einem der Knoten des Baumes enthalten ist, andernfalls `false`. Ist das Element mehrfach vorhanden, soll die Suche abgebrochen werden, wenn das erste Element gefunden wurde.

2. (3 Punkte) Definieren Sie die fehlende Methode `int berechneHoehe()`, die die Höhe eines Baumes ermittelt.

   Der als Beispiel angegebene Baum hat die Höhe 2. Gehen Sie dabei davon aus, dass es sich um beliebige Binärbäume handelt, die aber immer mindestens einen Knoten enthalten.



3. (1 Punkt) Wir betrachten einen Binärbäum, in dem als Werte Zahlen abgespeichert sind und der *linksvoll* ist. Linksvoll ist ein Binärbaum dann, wenn der Baum bis auf die unterste Schicht voll besetzt ist und in der untersten Schicht von links her keine Lücken vorhanden sind. Beim Durchlaufen eines linksvollen Binärbaums in *Post-Order* entsteht die Folge `7, 9, 23, 154, 7, 8, 9, 21, 14, 16, 22`. Zeichnen Sie den entsprechenden Baum.

Aufgabe 6 (9 Punkte) Abstrakte Klassen.

In dieser Aufgabe sollen Klassen modelliert werden, die Kisten eines Lagers repräsentieren. Jede Kiste hat eine eindeutige Nummer und ein Ankunftsdatum im Lager. Kisten werden in zwei Arten unterteilt: Gefrierkisten und normale Kisten. Für eine Gefrierkiste muss es die Möglichkeit geben, die Lagertemperatur zu vermerken. Normale Kisten können auch außerhalb der Lagerhalle gelagert werden. Schreiben Sie für die Klassen Konstruktoren, die sicher stellen, dass keine Objekte ohne die angegebenen Merkmale (Nummer, Ankunftsdatum, Temperatur bei Gefrierkisten, Lagerort bei normalen Kisten) erzeugt werden können.
Für jede Kistenart muss es die Möglichkeit geben, die Verweildauer und die Lagergebühr für den Lagerzeitraum zu berechnen. Diese Funktionalität soll durch die Methoden `int berechneDauer()` und `int berechnePreis()` zur Verfügung gestellt werden. Die Lagergebühren können Sie der folgenden Tabelle entnehmen.

	Gefrierkiste unter −15 Grad	Gefrierkiste zw. −7 und −15 Grad	Normale Kiste Halle	Normale Kiste Lagerplatz
Preis/Tag	560 Euro	280 Euro	180 Euro	70 Euro

Um das Ankunftsdatum darzustellen, steht Ihnen die Klasse `Datum` zur Verfügung. Sie dürfen vorraussetzen, dass die Methode `berechneDifferenzInTagen()` die Differenz zwischen dem durch ein Datumsobjekt repräsentierten Datum und dem aktuellen Datum als Anzahl von Tagen zurückgeliefert wird.

```
class Datum{
  int tag, monat, jahr;
  Datum(int t, int m, int j){ ... }
  int berechneDifferenzInTagen( ){ ... return d;}
}
```

1. (3 Punkte) Schreiben Sie eine abstrakte Klasse `Kiste`.

   - Überlegen Sie, welche der geforderten Attribute bereits in dieser Klasse definiert werden können und definieren Sie sie gegebenenfalls.
   - Überlegen Sie, ob die geforderten Methoden `int berechneDauer()` und `int berechnePreis()` bereits in dieser abstrakten Klasse implementiert werden können. Wenn dies der Fall ist, implementieren Sie die Methoden in dieser Klasse.

```
abstract class Kiste{




}
```

2. (3 Punkte) Schreiben Sie eine von `Kiste` abgeleitete Klasse `Gefrierkiste`. Fügen Sie dabei die Attribute und Methoden ein, die nicht in die abstrakte Klasse `Kiste` eingefügt werden konnten.

```
class Gefrierkiste extends Kiste{

}
```

3. (3 Punkte) Schreiben Sie eine von `Kiste` abgeleitete Klasse `NormaleKiste`. Fügen Sie dabei die Attribute und Methoden ein, die nicht in die abstrakte Klasse `Kiste` eingefügt werden konnten.

```
class NormaleKiste extends Kiste{

}
```

Aufgabe 7 (6 Punkte) Java.

Das folgende Java-Programm ist fehlerhaft. Geben Sie die Zeilennummer an und beschreiben Sie den Fehler. Folgefehler (also Fehler, die aus anderen Fehlern resultieren), sollen ignoriert werden. Sie erhalten für jeden korrekt erkannten Fehler eine Punkteinheit. Für korrekten Programmcode, den Sie als falsch markieren, wird Ihnen eine Punkteinheit abgezogen. Negative Punkte werden dabei jedoch nicht vergeben.

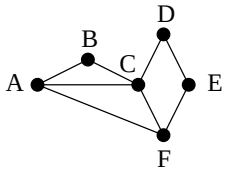
```
1   abstract class Fehler{
2     final long MAX = 4;
3     void print(){ Terminal.println("MAX ist: " + MAX); }
4     abstract void run();
5   }
6   class UnterFehler implements Fehler{
7     private int min;
8     UnterFehler(int m){ this.min = min; super.MAX = min * 10000l; }
9   }
10  class Testprogramm{
11    public static void main(String args[]){
12      Fehler f = new Fehler();
13      UnterFehler uf = new UnterFehler(4);
14      uf.min = Terminal.askInt("Neues Minimum eingeben: ");
15      for(int i = 0; i < 10; i++){
16        Terminal.println("i hat den Wert: " + i);
17      }
18      Terminal.println("Jetzt hat i den Wert: " + i);
19    }
20  }
```

Aufgabe 8 (5 Punkte) Java.

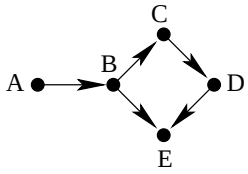
Zwei Wörter sind Anagramme, wenn sie aus denselben Buchstaben in beliebiger Reihenfolge bestehen. So sind zum Beispiel die Wörter *Lampe* und *Palme* Anagramme.
Schreiben Sie eine Funktion `boolean sindAnagramme(String s1, String s2)`, die für zwei als Strings übergebene Wörter prüft, ob es sich dabei um Anagramme handelt.
Sie dürfen dabei die Funktionen `char[] toCharArray(String s)` vorraussetzen, die für den als Parameter übergebenen String s ein Array aller enthaltenen Zeichen zurückgibt. Der Aufruf `toCharArray("ABA")` z.B. liefert das Array `{A, B, A}`.

Aufgabe 9 (2 Punkte) Graphen.

Geben Sie die Adjazenzmatrix und Adjazenzliste der folgenden Graphen an.



Graph G



Graph H

Name: .....

Matr.-Nr. .....