Name: Matrikelnummer:

Klausur: Java (Liste P)

SS05

Erlaubte Hilfsmittel:
Gebundene! Unterlagen (Skript mit Anmerkungen, eigene Mitschrift) und maximal ein Buch. Bitte keine losen Blätter.

Lösung ist auf den Klausurbögen anzufertigen. (eventuell Rückseiten nehmen)

Bitte legen Sie einen Lichtbildausweis und den Studentenausweis auf den Tisch.

Bearbeitungszeit: 90 Minuten

Unterschrift

Benotung

Aufgabe:	1	2	3	4	5/6		Gesamt	Note
Punkte:	20	20	20	20	20		100	
erreicht:								

Aufgabe 1 (Punkte)
(Java Syntax und Semantik)
Die folgenden Javaprogramme enthalten Fehler. Beschreiben Sie die Fehler und korrigieren Sie sie:

a)
.java
```
class Aufgabe2a {
  String s = "hallo";
  static public void main(String [] args){
    System.out.println(s);
  }
}
```

b)
.java
```
class Aufgabe2b {
  private static final String name = "Shakespeare";
  public static void main(String [] args) throws Exception{
    name.toUpperString();
    name="William "+name;
    System.out.println(name);
  }
}
```

c)
.java
```
class Aufgabe2c {
  private static String name = "Shakespeare";
  public static void main(String [] args){
    System.out.println(name.charAt(3)=="l");
    name=n.length();
    System.out.println(name.charAt(3)=="l");
  }
}
```

d)
.java
```
class Aufgabe2d{
  String name ;
  int i = 0,

  Aufgabe2d(String name){this.name=name;}

  String getName() {
    if (null==name) throw new Exception("null value in name");
    else  {
      name = name+i;
      return name;
    }
  }
}
```

e)
.java
```

interface Descriptable {
  abstract String getDescription();
}
```

```
5
6  class Aufgabe2e implements Descriptable {
7    String getDescription(){return "Aufgabe2e";}
8    public static void main(String [] args){
9      System.out.println(new Aufgabe2e().getDescription());
10   }
11 }
```

.java

f)

```
1  class Aufgabe2f extends Object{
2    int sum(int x) {
3      if (x==0) return 1;
4      else x=x+sum(x-1);
5    }
6  }
```

Aufgabe 2 (Punkte)
(Modellieren und Schreiben von Klassen)

a) Modellieren Sie Klassen für die Beschreibung von Webseiten. Eine Webseite habe eine Liste von Links. Ein Link sei dabei ein Objekt, das zwei Zeichenketten enthält: einmal die Zieladresse (URL) und einmal eine Beschreibung des Links.

Implementieren Sie die benötigten Klasse in Java.

b) Ergänzen Sie Ihre Klasse um eine Methode
`public String getDescription(String url);`,
die zu einer Zieladresse die gespeicherte Beschreibung zurück gibt.

c) Welches allgemeine Konzept realisiert ihre Klasse aus Aufgabe b).

d) Fügen Sie Ihrer obigen Klasse zur Darstellung von Linkseiten eine Methode hinzu, die es erlaubt neue Links in die Seite einzufügen:
`public void addLink(String url,String description);`

e) Fügen Sie Ihrer Klasse zur Darstellung von Linkseiten eine Methode zur Erzeugung eines HTML-Strings hinzu. Für das Webseitenobjekt soll ein String erzeugt werden, der in HTML die Seite mit allen in der Liste enthaltenen Links darstellt.
`public String toHTML();`

Aufgabe 3 (Punkte)
(einfach verkettete Listen)
Gegeben sei die folgende abstrakte Klasse für Listen, gemäß unserer Spezifikation aus der Vorlesung. Listenelemente sind Strings:

.java

```
1 abstract class AbList{
2   abstract public AbList empty();
3   abstract public AbList cons(String x, AbList xs);
4   abstract public boolean isEmpty();
5   abstract public String head();
6   abstract public AbList tail();
7 }
```

Schreiben Sie für die Klasse folgende Methoden, die ihr hinzugefügt werden können:

a) `boolean containsIgnoreCase(String o )`: ist genau dann wahr, wenn das Argument `o` unter Vernachlässigung der Groß- und Kleinschreibung in der Liste enthalten ist.
Beispiel: `("friNDS","romans","contrymen").containsIgnoreCase("friends")` ergibt: `true`.

b) `int maxStringLength(int currentMaxLength)` berechnet die Länge des längsten Strings, der in der Liste gespeichert ist. Das Argument `currentMaxLength` soll dabei eine bisher schon gefundene Länge mit angeben.
Beispiel: `("friends","romans","countrymen").maxStringLength(0)` ergibt: `10`.

c) Implementieren Sie für die Klasse `AbList` die Methode:
`boolean thisIsLonger(AbList other)`: das Ergebnis sei genau dann wahr, wenn die Liste länger ist, als die Argumentliste. Eine Methode `length` auf Listen steht Ihnen dabei nicht zur Verfügung.
Beispiel: `("a","b","c").thisIsLonger(("a","b","c","d"))` ergibt: `false`.

Aufgabe 4 (Punkte)
(GUI Programmierung)

a) Schreiben Sie eine GUI-Komponente `AutomaticCounter`, die eine Zahl anzeigt, die jede Sekunde um eins erhöht wird.

b) Ergänzen Sie Ihre GUI-Komponente `AutomaticCounter` um einen Knopf. Durch Drücken des Knopfes soll die angezeigte Zahl wieder auf `0` zurückgesetzt werden.

Aufgabe 5 (Punkte)
(generische Typen)
Benutzen Sie in dieser Aufgabe das generische Typsystem von Java 5:

a) Schreiben Sie eine generische Schnittstelle `Function` die einstellige Funktionen darstellt. Die Schnittstelle soll eine einstellige Methode mit Namen `apply` enthalten.

b) Implementieren Sie den Typ `Function<String,Integer>`, so daß die Methode `apply` die Länge eines Stringparameters zurückgibt.

c) Schreiben Sie eine statische Methode `map`, mit zwei Parametern:

- einem Objekt der Klasse `Function`
- einem Objekt der Standardschnittstelle `java.util.List`.

Das Ergebnis von `map` soll eine neue Liste sein, die ensteht, wenn man das Funktionsobjekt auf jedes Element des Listenparameters anwendet.

Achtung: Wiederholer, die bereits bei Herrn Dreher im WS04/05 die Klausur geschrieben haben, können alternativ die Aufgabe 6 lösen.

Aufgabe 6 (Punkte)
(Threads und Simulation)
Achtung nur für Studenten, die bereits bei Herrn Dreher die Klausur geschrieben haben. Alternativ zu Aufgabe 5

Erstellen Sie eine Anwendung zur Simulation einer Ampel.

- Die Ampel hat eine Warteschlange.
- Die Ampel steht jweils 45Sekunden auf grün und 45 Sekunden auf rot.
- Während der Grünphase kann alle zwei Sekunden ein Auto die Ampel passieren.
- Der zeitliche Abstand zwischen zwei Fahrzeugen die auf die Ampel zufahren liegt zwischen 2 und 16 Sekunden.