Name: Matrikelnummer:

Bachelorprüfung: Objektorientierte Softwareentwicklung

WS12/13

Erlaubte Hilfsmittel: keine

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		Gesamt	Note
Punkte:	20	20	20	20	20		100	
erreicht:								

Name: Matrikelnummer:

Aufgabe 1 (20 Punkte)
(Modellieren und Schreiben von Klassen)
Sie sollen in dieser Aufgabe Klassen entwickeln, die es erlauben, Informationen über Prüfungsordnungen zu speichern.

a) Entwickeln Sie eine Klasse `PO`. Ein PO-Objekt dieser Klasse repräsentiert eine Prüfungsordung für einen Studiengang. Es soll die folgenden Eigenschaften haben: der Name des Studiengangs, der durch eine Zeichenkette dargestellt ist, ein bool'schen Wert, der angibt, ob es sich um einen Bachelor oder um einen Masterstudiengang handelt und eine Standardliste von Stringobjekten, die die einzelnen Paragraphen der Prüfungsordnung darstellen.

Schreiben Sie einen Konstruktor und überschreiben Sie die Methode `equals` auf adequate Weise. Überschreiben Sie auch die Methode `toString` auf sinnvolle Weise.

b) Schreiben Sie eine Unterklasse `BISPO` der Klasse `PO`. `BISPO`-Objekte repräsentieren Prüfungsordnungen für berufsbegleitende Studiengänge. Diese enthalten zusätzlich in einem ganzzahligen Feld die Anzahl der Praxisstunden im Betrieb pro Semester.

Die Methode `toString` soll überschrieben werden, um die zusätzliche Information mit zu berücksichtigen.

c) Gegeben sei die folgende Schnittstelle:

Comp.java

```
1 public interface Comp<A>{
2   boolean greaterThan(A a1, A a2);
3 }
```

Schreiben Sie eine Klasse, die diese Schnittstelle für PO-Objekte so implementiert, dass die Methode `greaterThan` true ergibt, wenn `a1` mehr Paragraphen hat als `a2`.

```
class POComp implements Comp<PO> {
  public boolean greaterThan(PO p1, PO p2){
    return p1.pars.size() > p2.pars.size();
  }
}
```

```
import java.util.List;
```

Name: Matrikelnummer:

```
class PO{
  String name;
  boolean bachelor;
  List<String> pars;
  PO(String n, boolean b, List<String> ps){
      name = n;
      bachelor = b;
      pars = ps;
  }

  @Override
  public boolean equals(Object o){
    if (o.getClass() != PO.class)
                    return false;
    PO that = (PO) o;
    return this.name.equals(that.name)
          && this.bachelor == that.bachelor
          && this.pars.equals(that.pars)
  }
```

Name: Matrikelnummer:

```
class BISPO extends PO {
  int praxis;
  BISPO(String n, boolean b, List<Kind> ps, int p) {
    super(n, b, ps);
    praxis = p;
  }
  @Override
  public String toString() {
    return super.toString()
      + " praxisbegleitend mit "
      + praxis + " Praxisstunden";
  }
}
```

```
@Override
public String toString(){
  return (bachelor? "Bacelor": "Master")
          +" Student name: "+ name;
}
}
```

Name: Matrikelnummer:

Aufgabe 2 (20 Punkte)
(Programmfluss)

a) Führen Sie die folgende Klasse von Hand aus. Schreiben Sie dabei auf, in welcher Reihenfolge die Zeilen durchlaufen werden und mit welchen Werten die einzelnen Variablen während des Programmdurchlaufs belegt sind. Schreiben Sie auf, was auf dem Bildschirm ausgegeben wird.

Aufgabe2a.java

```
1  class Aufgabe2a{
2    public static void main(String [] args){
3      int y = 9;
4      for (int x = 43; x > y; x = x-2){
5        System.out.println(x+" "+y);
6        switch (x%5){
7           case 4: x = x-4;
8           case 3: x = x-1;
9                   y = y+1;
10          default: x = x-1;
11       }
12     }
13   }
14 }
```

Name: Matrikelnummer:

b) Betrachten Sie folgende Klasse:

Aufgabe2b.java

```
1  class Aufgabe2b{
2    static int f(int x,int y1,int y2){
3       return g(y1*y2,y2,x);
4    }
5    static int g(int x1,int x2,int y){
6      if (0==y) return x1;
7      return f(y-1,x1,x2);
8    }
9    static int g(int x,int y){
10     return g(x,x,y);
11   }
12   public static void main(String [] args){
13    System.out.println(g(2,4));
14   }
15 }
```

Berechnen Sie schrittweise das Ergebnis des Ausdrucks `g(2,4)`. Was berechnet die Methode `g`.

Name: Matrikelnummer:

Aufgabe 3 (20 Punkte)
(SwingGUI)
Gegeben Sei folgende Schnittstelle:

Logik.java

```
1 interface Logik{
2   String rechne(String);
3 }
```

a) Schreiben Sie eine Klasse `Dialog`, die von der Klasse `JPanel` erbt. Sie enthalte drei Felder: ein Objekt des Typs `Logik`, ein `JButton`-Objekt und zwei `JTextField`-Objekte.

Implementieren Sie die Klasse so, dass bei Drücken des Knopfes der Text des einen Textfeldes gelesen wird (`getText()`), der Text der Methode `rechne` des `Logik`-Objekts übergeben wird und das Ergebnis dieses Methodenaufrufs dem zweiten Textfeld als Text geschrieben wird (`setText(String s)`).

Der Kontruktor der Klasse soll genau einen Parameter des Typs `Logik` haben.

b) Schreiben Sie eine Applikation mit einer `main`-Methode, in der ein Fenster, mit einem `Dialog`-Objekt geöffnet wird. Bei Drücken des Knopfes soll der Text des Eingabefeldes in Großbuchstaben umgewandelt im Ausgabefeld erscheinen.

Name: Matrikelnummer:

Aufgabe 4 (20 Punkte)
(rekursive Strukturen)
Gegeben sei folgende Klasse von einfach verketteten Listen, wie aus der Vorlesung bekannt.

Li.java

```
1  public class Li<A> {
2    private final A element;
3    private final Li<A> next;
4    public Li(A element, Li<A> next){
5     this.element = element;
6     this.next = next;
7    }
8    public Li(){
9     element = null;
10    next = null;
11   }
12
13   A getElement(){return element;}
14   Li<A> getNext(){return next;}
15
16   boolean isEmpty(){return element==null&&next==null;}
17 }
```

Implementieren Sie folgende Funktionen rekursiv:

a) `int howMany(A a)`
Es soll als Ergebnis berechnet werden, wie viel Elemente in der Liste gleich zu dem übergebenen Parameter a sind.

Lösen Sie diese Aufgabe rekursiv.

Name: Matrikelnummer:

b) Gegeben sei zusätzlich folgende Schnittstelle (wie schon aus Aufgabe 1 bekannt):

Comp.java

```
1 public interface Comp<A>{
2   boolean greaterThan(A a1, A a2);
3 }
```

Sie soll zum Vergleichen von zwei Objekten benutzt werden. Die Methode `greaterThan` gibt true zurück, wenn das Argument `a1` größer ist als das Argument `a2`.

Schreiben Sie jetzt in der Klasse `Li` folgende Methode iterativ:
`static A greatest(Comp<A> comp);`
Für eine leere Liste soll `null` zurück gegeben werden. Ansonsten soll das in der übergebenen Relation `Comp` größte Element der Liste zurück gegeben werden.

Name _____ Matrikelnummer _____

Aufgabe 5 (20 Punkte)
Zusammenhänge
Erklären Sie in kurzen Worten.

a) Wie unterscheidet sich bei Feldern das Verhalten in Gegensatz zu Methoden in Bezug auf Vererbung. Was kann man programmiertechnisch dagegen machen?

b) Nennen Sie je drei Beispiele für Ausdrücke und Befehle.

Name: Matrikelnummer:

c) Erklären Sie den Unterschied zwischen Überschreiben und Überladen.

d) Nennen Sie zwei Klassen, die die Schnitstelle `java.util.List` implementieren und geben jeweils einen Vorteil der Implementierung an.

Name: Matrikelnummer:

e) Was muss bei jedem Konstruktor als erster Befehl gemacht werden und warum?