Name: Matrikelnummer:

Bachelorprüfung: Objektorientierte Softwareentwicklung

WS17/18

Erlaubte Hilfsmittel: keine

Jeder Griff zu einem elektronischen Gerät (z.B. Smartphone) wird als Täuschungsversuch gewertet.

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

Bitte legen Sie den Studentenausweis auf den Tisch.

Bearbeitungszeit: 90 Minuten

Unterschrift

Benotung

Aufgabe:	1	2	3	4	5	6		Gesamt	Note
Punkte:	20	20	7	7	26	20		100	
erreicht:									

Aufgabe 1 (Modellieren und Schreiben von Klassen)

Sie sollen in dieser Aufgabe Klassen entwickeln, die es erlauben, Informationen über Kunstwerke zu speichern.

a) Entwickeln Sie eine Klasse `Kunstwerk`. Ein Objekt dieser Klasse repräsentiert einen künstlerisches Werk. Es soll drei Eigenschaften haben: einen Namen des Künstlers, der durch eine Zeichenkette dargestellt ist, den geschätzen Wert des Werls in €, als Datentyp `int` dargestellt und den Titel des Werks als Zeichenkette.

Schreiben Sie einen Konstruktor, der alle Felder initialisiert.

Schreiben Sie einen zweiten Konstruktor, der keinen Wert für den geschätzen Wert übergeben bekommt, sondern den Standardwert 42 setzt.

Überschreiben Sie die Methode `equals` auf adequate Weise.

b) Schreiben Sie eine Unterklasse `Gemälde` der Klasse `Kunstwerk`. Ein Gemälde habe zusätzlich eine Höhe und Breite in cm. Außerdem sei es von einer der folgenden Kategorien: öl, radierung, aquarell, zeichnung.

c) Gegebe Sei die Klasse `Versteigerung`:

```
1 class Versteigerung extends java.util.ArrayList<Kunstwerk>{
2 }
```

Listing 1: Versteigerung.java

Schreiben Sie für diese Klasse eine Methode `long gesamtWert()`, die die Summe der geschätzten Werte aller Werke der Versteigerung zurück gibt.

Schreiben Sie für diese Klasse eine Methode

`Kunstwerk teuerstesKunstwerk()`,

die das Werk mit dem höchsten geschätzten Wert der Versteigerung zurück gibt.

Aufgabe 2 (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.

```
1  class Aufgabe2a{
2    public static void main(String[] args){
3      int x = 42;
4      int y = 17;
5      while (x>y){
6        switch (x){
7        case 56:;
8        case 18:;
9        case 42:;
10       case 9: x=x-y-1;
11       case 19: y--;break;
12       default: x=x-2;
13       }
14       System.out.println(x+" "+y);
15     }
16   }
17 }
```

Listing 2: Aufgabe2a.java

b) 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.

```
1  class Aufgabe2b{
2    public static void main(String [] args){
3      int y = 5;
4      int x = 14;
5
6      do{
7        for (int i=y;i>0;i=i-4){
8          System.out.println("I:"+i+" "+x+" "+y);
9          y += 3;
10         i--;
11       }
12       if (x == 15){
13        x++;
14        continue;
15       }
16       x--;
17       System.out.println("A:"+x+" "+y);
18     }while (x>=y);
19   }
20 }
```

Listing 3: Aufgabe2b.java

c) Betrachten Sie folgende Klasse:

```
1  class Aufgabe2c{
2    static int e(int a, int b) {
3      if (a == 0) return b;
4      if (b == 0) return a;
5      return a > b ? e(a % b, b) : e(a, b % a);
6    }
7
8    public static void main(String [] args){
9      System.out.println(e(19,51));
10   }
11 }
```

Listing 4: Aufgabe2c.java

Berechnen Sie schrittweise das Ergebnis des Ausdrucks `e(19,51)`.

d) Betrachten Sie folgende Klasse:

```
1  class Aufgabe2d{
2    static int p(int a, int b) {
3      if (!(a>b)) return q(a);
4      return (a+b)%2==1  ? (a+p(a-1,b+2)) : (a+p(a-2,b+1));
5     }
6     static int q(int a ) {
7      return a/2 <=1 ? -50 : a+q(a-18);
8     }
9    public static void main(String [] args){
10     System.out.println(p(38,36));
11   }
12 }
```

Listing 5: Aufgabe2d.java

Berechnen Sie schrittweise das Ergebnis des Ausdrucks `p(38,36)`.

Aufgabe 3 (Ausdrücke und Anweisungen)

Gegeben sei eine einfache Klasse für Datumsobjekte:

```
1  class Date{
2    int  day;
3    int month;
4    int year;
5
6    Date(int newDay, int newMonth, int newYear){
7      day = newDay;
8      month = newMonth;
9      year = newYear;
10   }
11
12   public String toString(){
13     return (day+"."+month+"."+year);
14   }
15 }
```

Listing 6: Date.java

a) Betrachten Sie folgende Methode für diese Klasse:

```
1  public boolean isEarlierThan(Date that){
2    boolean isEarlier = false;
3    if(this.year < that.year){
4      isEarlier = true;
5    }else if(this.year == that.year && this.month < that.month){
6      isEarlier = true;
7    }else if (this.year == that.year && this.month == that.month &&
       this.day < that.day){
8      isEarlier = true;
9    }else{
10     isEarlier = false;
11   }
12   return isEarlier;
13 }
```

Schreiben Sie eine Version dieser Methode, die außer einem `return` keine Anweisung sondern nur einen Ausdruck verwendet.

b) Schreiben Sie für die Klasse eine Methode, die anzeigt, dass das this-Objekt im Kalender nach dem Parameter `that` liegt:

```
boolean isLaterThan(Date that)
```

Hierbei sollen Sie einen Einzeiler schreiben und keinen Code aus der Methode aus Teil a) kopieren.

Aufgabe 4 Gegeben sei folgende einfache Schnittstelle:

```
1  public interface ButtonLogic {
2    String eval(String str);
3    }
```

Zusätzlich sei folgende JavaFX-Klasse gegeben:

```
1  import javafx.application.Application;
2  import javafx.scene.Scene;
3  import javafx.scene.control.Button;
4  import javafx.scene.control.TextField;
5  import javafx.scene.layout.BorderPane;
6  import javafx.stage.Stage;
7
8  public class Dialogue extends Application {
9    ButtonLogic logic;
10   public Dialogue(ButtonLogic logic) {
11     this.logic = logic;
12   }
13
14   @Override
15   public void start(Stage primaryStage) {
16     Button btn = new Button();
17     TextField out = new TextField();
18     TextField in = new TextField();
19     out.setEditable(false);
20     btn.setText("Drücken");
21     btn.setOnAction((event) -> {
22       out.setText(logic.eval(in.getText()));
23     });
24     BorderPane root = new BorderPane();
25     root.setTop(in);
26     root.setCenter(btn);
27     root.setBottom(out);
28     primaryStage.setScene(new Scene(root));
29     primaryStage.show();
30   }
31   }
```

Schreiben Sie eine Unterklasse von `Dialogue`, so dass eine FX-Applikation entsteht, bei der eine Zahl eingegeben werden kann und auf Knopfdruck die Fakultät dieser Zahl in einem Ausgabefeld angezeigt wird.

Aufgabe 5 (Arraylisten)

Gegeben sei folgende Klasse, die eine einfache Array-basierte Liste realisiert, wie aus der Vorlesung bekannt.

```
1  public class AL<A> {
2    private int size = 0;
3    private A[] array = (A[])new Object[10];
4
5    public void add(A el){
6      if (size >= array.length){
7        enlargeArray();
8      }
9      array[size++] = el;
10   }
11
12   private void enlargeArray(){
13       A[] newarray = (A[])new Object[array.length + 10];
14       for (int i=0; i<array.length; i++) newarray[i] = array[i];
15       array = newarray;
16   }
17
18   public A get(int i) {
19     return array[i];
20   }
21   public int size(){return size;}
22   public boolean isEmpty(){return size == 0;}
23 }
```

Listing 7: AL.java

Implementieren Sie folgende Methoden für diese Listenklasse:

a) `boolean startsWith(AL<A> that);`

Sie soll genau dann true ergeben, wenn die that-Liste ein Präfix der this-Liste ist, also wenn die that-Liste der Anfang von der this-Liste ist. Für jedes Element x_i der that-List also gilt, dass es auch das i-te Element der this-Liste ist.

b) Gegeben sei zusätzlich folgende Schnittstelle:

```
1 public interface Property<A>{
2   boolean test(A a1);
3 }
```

Listing 8: Property.java

Sie soll verwendet werden, um zu testen, ob ein Objekt eine bestimmte Eigenschaft hat.

Schreiben Sie jetzt in der Klasse `AL` folgende Methode:

```
boolean all(Property<A> p);
```

Sie soll genau dann true ergeben, wenn alle Elemente der Liste, die übergebene Eigenschaft haben.

c) Schreiben Sie jetzt in der Klasse `AL` folgende Methode:

```
AL<A> filter(Property<A> p);
```

Es soll eine neue Liste erzeugt werden, die aus allen Elementen der Ursprungsliste, die die übergebene Eigenschaft erfüllen, besteht.

d) Was lässt sich über folgende zusätzliche Methode in `AL` sagen? Optimieren Sie die Methode:

```
1 boolean m(Property<A> p){
2   return this.filter(p).all(p);
3 }
```

e) Gegeben sei zusätzlich folgende Schnittstelle:

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

Listing 9: Comp.java

Schreiben Sie folgende Methode: `public boolean isSorted(Comp<A> comp)`

Sie soll genau dann true ergeben, wenn für alle Elemente der Liste gilt, dass sie kleiner im Bezug auf die Methode `isSmaller` des Comp-Objektes sind als alle nachfolgenden Elemente der Liste.

Aufgabe 6 Zusammenhänge

Erklären Sie in kurzen Worten.

a) Was hat jede abstrakte Klasse, das keine Schnittstelle hat?

b) Was versteht man unter dem Encoding?

c) Welche dieser Zuweisungen sind korrekt? Erklären Sie warum:

1. `List<Long> ls = new ArrayList<Long>();`
2. `ArrayList<Object> ls = new ArrayList<Long>();`
3. `Object[] ls = new Long[42];`
4. `List<List<String>> xs = new ArrayList<>();`

d) Was ist beim Überschreiben einer Methode bezüglich Ausnahmen zu beachten?

e) Was versteht man unter einem statischen Typcheck?