

Bachelorprüfung: Objektorientierte Softwareentwicklung

WS19

Studiengang (AI, AI Dual): _____

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

Aufgabe 1 (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 Objekt dieser Klasse repräsentiert eine Prüfungsordnung eines Studiengangs. Es soll drei Eigenschaften haben: einen Namen des Studiengangs, der durch eine Zeichenkette dargestellt ist, ein Aufzählungsobjekt, der angibt, ob es sich um einen Bachelor- oder Masterstudiengang handelt und das Jahr, in dem die Prüfungsordnung in Kraft getreten ist.

Sie benötigen dabei auch die Definition einer Aufzählungsklasse.

Schreiben Sie einen Konstruktor, der alle Felder initialisiert.

Überschreiben Sie die Methode `equals` auf adequate Weise.

Überschreiben Sie die Methode `toString`.

- b) Schreiben Sie eine Unterklasse `Dual` der Klasse `PO`, die eine Prüfungsordnung dualer Studiengänge repräsentiert. Ein dualer Studiengang habe zusätzlich eine Angabe über die Anzahl der in einem Betrieb abzuleistenden Stunden.

- c) Gegeben Sei die Klasse `Fachbereich`, die eine Liste von Studiengängen enthält:

```
1 class Fachbereich{
2     java.util.List<PO> pos = new java.util.ArrayList<>();
3 }
```

Listing 1: `Fachbereich.java`

Schreiben Sie für diese Klasse eine Methode

```
int anzahlDerDualenStudiengänge()
```

die die Anzahl der der dualen Studiengänge in der Liste zurück gibt.

- d) Schreiben Sie für diese Klasse eine Methode

```
PO ältestePO(),
```

die die am längsten gültige Prüfungsordnung berechnet.

Sie dürfen davon ausgehen, dass die Liste nicht leer ist.

Name:

Matrikelnummer:

Name:

Matrikelnummer:

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 y = 4;
4         for (int x = 25;!(y>x-1);x--){
5             switch (x-1){
6                 case 20;;
7                 case 8;;
8                 case 42;;
9                 case 9: x=x-y-2;
10                case 11: y--;break;
11                case 1: y++;return;
12                default: x--;
13            }
14            x -=1;
15            System.out.println(x+" "+y);
16            x--;
17        }
18        System.out.println(y);
19    }
20 }
```

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 [] xs = {5,13};
4     int y = 21;
5
6     for (var x:xs){
7       for (int i=y;i>0;i=i-6){
8         System.out.println("I:" +i+ " "+x+ " "+y);
9         x += 3;
10        y--;
11        if (y==19){
12          y=y+1;
13          break;
14        }
15      }
16      if (x == 13){
17        x=15;
18        continue;
19      }
20      x--;
21      System.out.println("A:" +x+ " "+y);
22    }
23  }
24 }
```

Listing 3: Aufgabe2b.java

c) Betrachten Sie folgende Klasse:

```
1 class Aufgabe2c{
2     static int f(int a) {
3         return a == 0 ? 0 : (f(a/2)+a%2) ;
4     }
5
6     public static void main(String [] args){
7         System.out.println(f(27));
8     }
9 }
```

Listing 4: Aufgabe2c.java

Berechnen Sie schrittweise das Ergebnis des Ausdrucks $f(27)$.

d) Betrachten Sie folgende Klassen:

```
1 class A2dO{
2     int f1(int x){
3         return f2(x);
4     }
5     int f2(int x){
6         return x+42;
7     }
8 }
```

Listing 5: A2dO.java

```
1 class A2dU extends A2dO{
2     int f2(int x){
3         return x+17;
4     }
5
6     int f1(int x){
7         return x-17;
8     }
9
10    public static void main(String [] args){
11        A2dO x1 = new A2dO();
12        A2dO x2 = new A2dU();
13        System.out.println(x1.f1(2));
14        System.out.println(x2.f1(2));
15        System.out.println(x1.f2(2));
16        System.out.println(x2.f2(2));
17    }
18 }
```

Listing 6: A2dU.java

Welche Ausgabe tätigt die Klasse A2dU bei Ausführung auf der Kommandozeile. Erklären Sie, wie es zu dieser Ausgabe kommt.

Aufgabe 3 (Ausdrücke und Anweisungen)

Gegeben sei eine einfache Klasse für Längenangaben im angloamerikanischen Maßsystem. 12 Inch sind dabei ein Foot.

```
1 class Distance{
2     int foot;
3     int inch;
4     Distance(int newFoot, int newInch){
5         foot = newFoot;
6         inch = newInch;
7     }
8
9     public String toString(){
10        return (foot+", "+inch);
11    }
12 }
```

Listing 7: Distance.java

- a) Schreiben Sie für die Klasse `Distance` eine Methode `isShorterThan`, die angibt, dass das Objekt eine kürzere Entfernung beschreibt, als der übergebene Parameter. Verwenden Sie dabei außer einem `return` keine weitere Anweisung und verwenden Sie nicht den Bedingungsoperator.

```
13 boolean isShorterThan(Distance that){
14     return
15
16
17
18
19
20
21
22
23
24
25
26
27
28 }
```

Listing 8: Distance.java

Name:

Matrikelnummer:

- b) Schreiben Sie für die Klasse eine Methode, die anzeigt, dass das `this`-Objekt eine längere Distanz als `that` beschreibt:

Verwenden Sie hierbei die Methode aus Aufgabenteil a).

```
29 boolean isLongerThan(Distance that){
30
31
32
33
34
35
36
37
38
39
40 }
```

Listing 9: Distance.java

- c) Schreiben Sie für die Klasse `Distance` eine Methode `addInch`, die das Objekt modifiziert, so dass es die als Parameter übergebene Länge in Inch länger ist:

```
41 void addInch(int inch){
42
43
44
45
46
47
48
49
50
51
52
53
54 }
```

Listing 10: Distance.java

Name:

Matrikelnummer:

- d) Ein Inch ist die Länge von 25,4 Millimeter. Schreiben Sie für die Klasse `Distance` eine Methode, die die beschriebene Länge in Millimetern berechnet:

```
55 float getDistanceInMillimeter () {  
56  
57  
58  
59  
60  
61  
62  
63  
64  
65  
66  
67  
68  
69  
70  
71 }
```

Listing 11: Distance.java

Aufgabe 4 (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    private void enlargeArray(){
12        A[] newarray = (A[])new Object[array.length + 10];
13        for (int i=0; i<array.length; i++) newarray[i] = array[i];
14        array = newarray;
15    }
16    public A get(int i) {
17        return array[i];
18    }
19    public int size(){return size;}
20    public boolean isEmpty(){return size == 0;}
21 }
```

Listing 12: AL.java

Implementieren Sie folgende Methoden für diese Listenklasse:

Name:

Matrikelnummer:

a) Schreiben Sie in der Klasse `AL` eine Methode `firstIndexOf`:

Es soll der erste Index aus der Liste berechnet werden, an dem ein Objekt gespeichert ist, das gleich zum Parameter `o` ist. Ist kein solches Objekt enthalten, so soll `-1` zurück gegeben werden.

```
22 int firstIndexOf(A o){  
23  
24  
25  
26  
27  
28  
29  
30  
31  
32  
33  
34  
35  
36  
37  
38  
39  
40 }
```

Listing 13: `AL.java`

Name:

Matrikelnummer:

b) Gegeben sei zusätzlich folgende Schnittstelle:

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

Listing 14: Property.java

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

Schreiben Sie jetzt in der Klasse `AL` eine Methode `firstIndexOf(Property<A> p)`.

Es soll der erste Index aus der Liste berechnet werden, an dem ein Objekt gespeichert ist, das die Eigenschaft `p` hat. Ist kein solches Objekt enthalten, so soll `-1` zurück gegeben werden.

```
41 int firstIndexOf(Property<A> p){  
42  
43  
44  
45  
46  
47  
48  
49  
50  
51  
52  
53  
54  
55  
56  
57 }
```

Listing 15: AL.java

Name:

Matrikelnummer:

- c) Schreiben Sie jetzt die Methode aus Aufgabenteil a) neu, unter Verwendung der Methode von Aufgabenteil b).

```
58 int firstIndexOf(A o){  
59  
60  
61  
62  
63  
64  
65 }
```

Listing 16: AL.java

- d) Schreiben Sie folgende Methode `deleteAt(int i)`. Die Liste soll so verändert werden, dass das Element am Index `i` aus der Liste gelöscht wird und die Liste anschließend ein Element weniger hat. Ist der Index `i` kein gültiger Index für ein Element, so bleibt die Liste unverändert.

```
66 void deleteAt(int i){  
67  
68  
69  
70  
71  
72  
73  
74  
75  
76  
77  
78  
79  
80  
81  
82  
83  
84  
85 }
```

Listing 17: AL.java

Aufgabe 5 Zusammenhänge

Erklären Sie in kurzen Worten.

a) Betrachten Sie folgende Methode;

```
1 static int f(int x){
2   while (x<0){
3     x *= -1;
4   }
5   return x;
6 }
```

Mit welcher sonst so verpönten Bezeichnung kann die Schleife zu Recht benannt werden. Formulieren Sie die Methode um, so dass keine Schleife verwendet wird.

b) Was gilt für alle abstrakten Methoden einer Schnittstelle und ist bei der Implementierung einer Schnittstelle zu beachten.

Name:

Matrikelnummer:

c) Welche Sichtbarkeiten für Methoden hat Java.

d) Geben Sie drei Beispiele für statische Analysen, die der Javacompiler durchführt.

Name:

Matrikelnummer:

e) Erklären Sie, welche Funktion Konstruktoren in Aufzählungsklassen haben.