

Bachelorprüfung: Objektorientierte Softwareentwicklung

SS22

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:	22	20	14	24	20		100	
erreicht:								

Aufgabe 1 (Modellieren und Schreiben von Klassen)

Sie sollen in dieser Aufgabe Klassen entwickeln, die es erlauben, Rauminformationen zu speichern.

- a) Entwickeln Sie eine Klasse **Raum**. Ein Objekt dieser Klasse repräsentiert einen Raum in einem Gebäude. Es soll zwei Eigenschaften haben: die Raumnummer, die durch eine Zeichenkette dargestellt ist und die Anzahl der Sitzplätze in dem Raum.

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 **Rechnerraum** der Klasse **Raum**. Rechnerräume haben an jedem Arbeitsplatz einen PC. Die Klasse **Rechnerraum** soll als zusätzliche Eigenschaft einen Aufzählungswert haben, der angibt, welches Betriebssystem auf den Rechnern installiert ist. Gültige Aufzählungswerte sollen sein: Unix, Linux, Windows, MacOS.

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

- c) Schreiben Sie eine Klasse **Gebäude**. Diese Klasse soll als Eigenschaft eine Liste von Räumen enthalten. Benutzen Sie hierzu Klassen aus dem Paket `java.util`.

Schreiben Sie eine Methode `int wievielPlatz(String raum)`, die angibt, wieviel Sitzplätze der Raum mit der übergebenen Raumnummer hat.

Name:

Matrikelnummer:

Name:

Matrikelnummer:

Aufgabe 2 (Programmfluss)

a) Was wird ausgegeben, wenn der folgende Code ausgeführt wird?

```
1 int number = 9;
2 while (number > 0){
3     number -= 4;
4     System.out.print(number + ' ');
5 }
```

Listing 1: Aufgabe2.java

b) Welchen Wert haben die Variable item und sigma, wenn die Schleife terminiert?

```
6 int sigma = 0;
7 int item = 0;
8 while (item < 8){
9     item += 1;
10    if (sigma >= 6) continue;
11    sigma += 1;
12 }
```

Listing 2: Aufgabe2.java

c) Wie oft druckt das folgende Programm das Wort "java" aus?

```
13 for (int i1=2; i1 < 14; i1++){
14     for (int i2=3; i2 < 9; i2++)
15         System.out.println("java");
16     if (i1 > 6) break;
17 }
```

Listing 3: Aufgabe2.java

d) Welchen Wert hat die Variable x, nach dem Durchlaufe dieses Code-Fragments?

```
18 var x = 10;
19 switch (x){
20     case 9: x+=1;
21     case 10: x+=2;
22     case 11: x+=3;
23     case 12: x+=4;
24     case 13: x+=5;
25 }
```

Listing 4: Aufgabe2.java

e) Gegeben sei folgende Klasse:

```
1 class Aufgabe2b{
2     static int q(int a,int b){
3         if (b==0) return a;
4         if (a==0) return b;
5         if (a>b) return q(a-b, b);
6         return q(a, b-a);
7     }
8
9     public static void main(String [] args){
10        System.out.println(q(51,12));
11    }
12 }
```

Listing 5: Aufgabe2b.java

Berechnen Sie schrittweise das Ergebnis des Ausdrucks $q(51, 12)$; . Notieren Sie jeweils, mit welchen Werten die Methode q aufgerufen wird.

Was berechnet die Funktion q ?

Aufgabe 3 Gegeben sei folgende Klasse für ein TicTacToe Spiel. In diesem Spiel gibt es ein 3x3 Felder großes Spielfeld. Dort kann eine Position mit 1,2 oder 0 belegt sein, je nachdem, ob ein Stein des ersten oder zweiten Spielers dort liegt, oder noch gar keine.

```
1 class TTT{
2     private int [][] spielfeld = new int [3][3];
3     boolean spielerEinsIstDran;
4 }
```

Listing 6: TTT.java

- a) Schreiben Sie eine Methode, die genau dann true ergibt, wenn die in den Argumenten übergebene Position einen gültigen Spielzug darstellt, weil es eine Position ist, an der noch kein Stein liegt.

```
5 boolean legalMove(int row, int column){
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29 }
```

Listing 7: TTT.java

Name:

Matrikelnummer:

- b) Schreiben Sie eine Methode, die die Zahl des Spielers berechnet, der gewonnen hat, weil er eine komplette Spalte, Zeile oder Diagonale mit einem Stein belegt hat. Wenn noch kein Spieler gewonnen hat, ist 0 zurück zu geben.

```
30 int theWinnerIs () {  
31  
32  
33  
34  
35  
36  
37  
38  
39  
40  
41  
42  
43  
44  
45  
46  
47  
48  
49  
50  
51  
52  
53  
54  
55  
56  
57  
58  
59  
60  
61  
62  
63  
64 }
```

Listing 8: TTT.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 9: AL.java

Implementieren Sie in den Teilaufgaben Methoden für diese Listenklasse:

Name:

Matrikelnummer:

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

Es soll angegeben werden, welches der höchste Index in der Liste ist, an dem ein Element enthalten ist, das gleich zum Parameter `o` ist. Ist kein solches Element enthalten, dann sei das Ergebnis `-1`.

```
22 int lastIndexOf(A o){
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51 }
```

Listing 10: `AL.java`

Name:

Matrikelnummer:

- b) Schreiben Sie jetzt die Methode `endsWith`. Sie soll genau dann wahr zurück geben, wenn die übergebene Liste eine Endliste dieser Liste ist.

```
52 boolean endsWith(List<E> that {  
53  
54  
55  
56  
57  
58  
59  
60  
61  
62  
63  
64  
65  
66  
67  
68  
69  
70  
71  
72  
73  
74  
75  
76  
77  
78 }
```

Listing 11: AL.java

Name: _____

Matrikelnummer: _____

c) Gegeben sei zusätzlich folgende Schnittstelle:

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

Listing 12: 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 `filter`.

Es soll eine neue Liste erzeugt werden, die aus allen Elementen der `this`-Liste besteht, für die die Eigenschaft des Parameter `p` `true` ergibt.

```
79 public AL<E> filter(Property<? super E> p){
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102 }
```

Listing 13: AL.java

Name:

Matrikelnummer:

- d) Wie könnten Sie unter Verwendung der Methode `filter` eine naive Implementierung der Methode `contains` in einem Ausdruck umsetzen, um zu testen, ob ein bestimmtes Element in der Liste enthalten ist. Was ist bei dieser Umsetzung nachteilhaft?

```
103 boolean contains(A o){  
104     return  
105  
106  
107  
108  
109  
110  
111 }
```

Listing 14: AL.java

Name:

Matrikelnummer:

Aufgabe 5 Zusammenhänge

Erklären Sie in kurzen Worten.

a) Was ist späte Bindung (*late binding*)?

b) Was unterscheidet die Klasse `RuntimeException` von allgemeinen `Exception`-Klassen.

Name:

Matrikelnummer:

c) Was ist der Unterschied zwischen Anweisungen und Ausdrücken?

d) Welche zwei Anwendungsfälle hat das Schlüsselwort `super`? Was ist dabei zu beachten?

e) Was sind anonyme Klassen?