Name: Matrikelnummer:

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:								

Name: Matrikelnummer:

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`. Rechneräume 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.

```
class Raum {
    String nr;
    int plätze;
    Raum(String n, int p) {
        nr = n; plätze = p;
    }
```

```
@Override
public boolean equals(Object o){
  if(o instanceof Raum that){
    return this.nr.equals(that.nr)
      && this.plätze == that.plätze;
  }
  return false;
}
@Override
public String toString(){
  return "Raum "+nr+" mit "
    +plätze+" Plätzen";
}
}
```

b) `enum OS {unix, linux, windows, macOs;}`

```
class Rechnerraum extends Raum {

    OS os;
    Rechnerraum(String nr, int plätze, OS o){
      super(nr, plätze);
      os=o;
    }
    @Override
    public String toString() {
      return super.toString()
          + " mit OS: " + os;
    }

}
```

c)

```
class Gebäude {
    List<Raum> räume;
    int wieviel Platz(String raum) {
      for(Raum r : räume){
        if(r.nr.equals(raum)) return r.plätze;
      }
    return 0;
}
```

Name: Matrikelnummer:

Aufgabe 2 (Programmfluss)

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

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

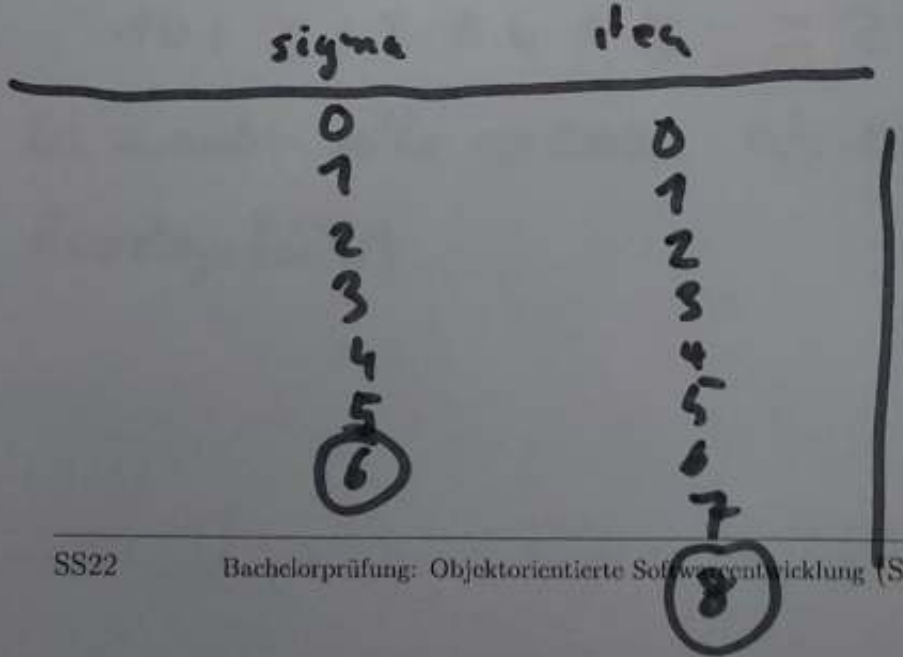
Listing 1: Aufgabe2.java

"5 1 -3"

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

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

Listing 2: Aufgabe2.java



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

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

Listing 3: Aufgabe2.java

i1 i2

2 3 → "java" } 6 x

: →

4 ... 8

3 ... 8 } 36 x

innere Schleife läuft immer 6 x } insgesamt 30 x "java"

äußere Schleife läuft 5 x

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

```
var x = 10;
switch (x){
  case 9: x+=1;
  case 10: x+=2;
  case 11: x+=3;
  case 12: x+=4;
  case 13: x+=5;
}
```

Listing 4: Aufgabe2.java

10 + 2 + 3 + 4 + 5 = 24

Es werden alle cases ab case 10 durchgeführt.

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? ✓

q(57,12)

→ q(57-12,12) → q(39,12)

→ q(39-12,12) → q(27,12)

→ q(27-12,12) → q(15,12)

→ q(15-12,12) → q(3,12)

→ q(3,12-3) → q(3,3)

→ q(3,6) → q(3,3)

→ q(3,0) → 3

Es wird GGT berechnet.

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    return spielfeld[row][column]
7    ==0;
8  }
```

Listing 7: TTT.java

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

```
int theWinnerIs(){ var sf = spielfeld;
  for(var r=0;r<3;r++){
    if(spielfeld[r][0]==spielfeld[r][1]
      && spielfeld[r][0]==spielfeld[r][2]
      && spielfeld[r][0]!=0)
        return spielfeld[r][0];
  }
  for(var c=0;c<3;c++){
    if(spielfeld[0][c]==spielfeld[1][c]
      && spielfeld[0][c]==spielfeld[2][c]
      && spielfeld[0][c]!=0)
        return spielfeld[0][c];
  }
  if(sf[0][0]==sf[1][1]
    && sf[0][0]==sf[2][2] && sf[0][0]
                                !=0)
    return sf[0][0]
} if(sf[0][2]==sf[1][1]
    && sf[2][0]==sf[1][1]
    && sf[1][1]!=0)
      return sf[1][1];
  return 0;
```

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:

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.

```
int lastIndexOf(A o){
  for(var i=size-1; i>=0; i--){
    if(get(i).equals(o))
      return i;
  }
  return -1;
}
```

Listing 10: AL.java

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

```
boolean endsWith(List<E> that{
    if(that.size() > this.size())
        return false;
    var p = this.size() - that.size();
    for(var i=0; i<that.size(); i++){
      if(!this.get(p+i).equals(that.get(i)))
        return false;
    }
    return true;
}
```

Listing 11: AL.java

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   var rs= new AL<E>();
81   for(var i=0; i<size(); i++){
82     if(p.test(get(i))
83       rs.add(get(i));
84   }
85   return rs;
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102 }
```

Listing 13: AL.java

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 !filter(x -> x.equals(o))
105           .isEmpty();
106
107
108
109
110
111 }
```

Listing 14: AL.java

Nachteilhaft ist, dass man eine komplette neue Liste im Speicher anlegt.

Aufgabe 5 Zusammenhänge

Erklären Sie in kurzen Worten.

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

Bei einem Methodenaufruf wird immer die spezifischste zuletzt überschriebene Methode der Klasse ausgeführt, von der das Objekt erzeugt wurde.
Bei Vererbung ist das wichtig.

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

RuntimeException Objekte sind nicht checked. Sie können geworfen werden, ohne im throws deklariert zu sein oder in einem catch abgefangen zu werden.

c) Was ist der Unterschied zwischen Anweisungen und Ausdrücken?

Ausdrücke berechnen immer ein Ergebnis.
Anweisungen steuern den Kontrollfluss
oder weisen Variablen neue Werte zu.

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

Mit Parametern wird der Konstruktor der
Oberklasse aufgerufen. Dort muss erste
Anweisung in einem Konstruktor sein.
Ohne Parameter kann auf eine geerbte oder
überschriebene Methode zugegriffen werden.

e) Was sind anonyme Klassen?

Klasse ohne Namen, für die gerade ein
Objekt mit new erzeugt wird.
Bei der Objekterzeugung können direkt
inline Methoden überschrieben werden.