

Datum -Zeit: 24.01.2019 – Beginn 10:00 Uhr

Nettobearbeitungszeit TEIL A: 60 Min

Punkteverteilung

	Punkte	Erreicht
Teil A		
a1.geoReihelt	5	
a1. geoReiheRek	5	
a1. n_fuer_eps	5	
a1.miniTabCalc	10	
a1.merge	10	
a2.VereinReader	17	
a2.Verein		
fruehesterKurs	4	
gruppierenNachJahrMonat	4	
a2.Verband		
gruppierenNachJahrMonat	7	
Teil A Gesamt	67	

Hinweise

- Die maximal gewerteten Punkte für Teil A sind **60** Punkte. Alles was > 60 Punkte ist, wird nicht gewertet!
- Verwenden Sie unbedingt die in den Projekten vorgegebenen Methoden.

Schreiben Sie in jede Datei Ihre Matrikelnummer und wenn Sie mögen Ihren Namen.

Viel Erfolg! ☺

Teil A: Verstehen und Programmieren

A1 Einzelaufgaben

1. **Iteration und Rekursion:** Gegeben die allgemeine Summenformel für geometrische Reihen:

$$\text{sum}(a, q, n) = \sum_{k=0}^n a * q^k;$$

Die geometrische Reihe konvergiert gegen $a/(1-q)$ für $|q| < 1$

- a. Berechnen Sie die Formel iterativ (statische Methode `geoReiheIt(double a, double q, int n)`) (**5Pkt**)
 - b. Berechnen Sie die Formel rekursiv (statische Methode `geoReiheRek(double a, double q, int n)`) (**5Pkt**)
 - c. Prüfen Sie in beiden Methoden die Gültigkeit der Eingabewerte und generieren Sie eine `IllegalArgumentException`, wenn der übergebene Wert nicht im geforderten Wertebereich liegt. (Punkte in a./b. enthalten)
 - d. Berechnen Sie für ein gegebenes Epsilon `eps` (ein sehr kleiner Wert) `a` und `q` das `n`, für das `/geoReiheIt(a,q,n)- a/(1-q) / < eps` wird. **Achtung:** Verwenden Sie nur die Methode mit der iterativen Lösung! (**5Pkt**)
2. **Arrays:** Gegeben ein regelmäßiges 2-dimensionales `double[][]` Array. Schreiben Sie die statische Methode `miniTabCalc(double[][], in)`,
- die jede Zeile des Arrays um zwei Spalten verlängert und
 - in den neuen Spalten 1.) die Reihensumme und 2.) den Mittelwert der Reihe einträgt. Die Reihensumme ist die Summe der Werte einer Reihe, der Mittelwert die Reihensumme geteilt durch die Anzahl der Elemente.
 - Wenn die Matrix nur leere Zeilen enthält, dann ist das Ergebnis eine 2-spaltige Matrix. In der ersten Spalte steht überall 0.0, in der zweiten `NaN`.
 - Sie müssen **nicht** prüfen, ob das Argument eine gültige Matrix ist.

Beispiel1:

Eingabe:

`[[1.0, 1.0, 1.0], [1.0, 1.0, 1.0], [1.0, 1.0, 1.0]]`

Ergebnis:

`[[1.0, 1.0, 1.0, 3.0, 1.0], [1.0, 1.0, 1.0, 3.0, 1.0], [1.0, 1.0, 1.0, 3.0, 1.0]]`

Beispiel2:

Eingabe:

`[[1.0, 2.0, 3.0, 4.0], [2.0, 3.0, 4.0, 5.0], [3.0, 4.0, 5.0, 6.0], [4.0, 5.0, 6.0, 7.0]]`

Ergebnis:

`[[1.0, 2.0, 3.0, 4.0, 10.0, 2.5], [2.0, 3.0, 4.0, 5.0, 14.0, 3.5], [3.0, 4.0, 5.0, 6.0, 18.0, 4.5], [4.0, 5.0, 6.0, 7.0, 22.0, 5.5]]`

3. **Maps (10 Pkt):** Schreiben Sie eine statische Methode, die zwei Maps **m1** und **m2**, deren Schlüssel **Integer** und deren Werte Listen von **Integer** sind, wie folgt mischt:

- Wenn **m1** und **m2** einen gleichen Schlüssel haben, dann sollen die Listen kopiert, zusammengehängt und in die Ergebnis-Map eingetragen werden.
- Für Keys, die nur in **m1** oder **m2** auftreten, sollen die Key-Value-Paare in die Ergebnis-Map übertragen werden. Auch hier müssen die Werte (die Listen) als Kopie übertragen werden.

Beispiel:

Eingabe:

m1: {1=[1, 2, 3], 2=[2, 3, 4], 3=[3, 4, 5]}

m2: {3=[4, 5, 6], 4=[5, 6, 7], 5=[6, 7, 8]}

Ergebnis:

{1=[1, 2, 3], 2=[2, 3, 4], 3=[3, 4, 5, 4, 5, 6], 4=[5, 6, 7], 5=[6, 7, 8]}

Tests zum ersten Aufgabenteil finden Sie in A1Test. Sie dürfen die Tests an keiner Stelle modifizieren!!!!

A2 Dateien einlesen / Auswertungen mit Hilfe von Streams

Gegeben ein einfaches Modell für Vereine und Verbände, das aus den Klassen **Verein**, **Kurs** und **Verband** besteht.

- Vereine werden mit einem Vereinsnamen (→ Enum **VereinsName**) erzeugt
- Kurse werden mit einer Sportart (→ Enum **Sportart**), einer Zeitangabe für den Beginn (**LocalDateTime**), einer Dauer (**Duration**) und der maximalen Teilnehmerzahl (**int**) erzeugt.
- Mit **addAll** können die Kurse dem Verein hinzugefügt werden.
- Mehrere Vereine sind in einem Verband organisiert. Vereine können mit **addAll** dem Verband hinzugefügt werden.

TODO's:

1. Implementieren Sie in der Klasse **VereinReader** die statische Methode **List<Verein> readData()**, die den Inhalt einer Datei parst und diesen in eine Liste von Vereinen überführt. **(17 Pkt)**

- a. Das Öffnen der Datei, Dateiname und -pfad sind bereits vorgegeben.
- b. Der Inhalt der Datei sieht wie folgt aus:

```
Fit12::Leichtathletik%01.12.2018 19:00%1H30M%22::Fußball%02.10.2018
20:30%2H%8
```

```
Lazy Crowd::Turnen%01.10.2018 22:30%1H30M%16::Handball%01.02.2018
15:00%2H%8::Turnen%02.01.2018 11:00%1H30M%12::Laufen%01.06.2018
17:00%1H%16::Schwimmen%01.05.2018 20:00%2H%9
```

```
Couch Potatoes::Fußball%01.04.2018 17:00%2H%9
Strong Performers::Turnen%02.04.2018 21:00%1H%26::Laufen%02.02.2018
14:30%1H30M%28
```

```
Movers::Volleyball%02.06.2018 16:00%2H%15::Fußball%01.11.2018
22:00%1H30M%12::Leichtathletik%02.12.2018
20:00%1H%21::Fußball%02.09.2018
14:00%1H%19::Leichtathletik%01.02.2018
20:00%1H30M%26::Handball%02.03.2018
15:00%2H%14::Leichtathletik%01.01.2018 15:00%1H30M%17
```

- c. Die Separatoren sind **::** und **%%**. Beide Separatoren sind als Konstanten der Klasse **GeneratorKonstanten** hinterlegt.
- d. Tests zu diesem Aufgabenteil finden Sie in der Datei **VereinReaderTest**.

Hinweise:

- Die Klasse **DateTimeUtil** enthält Methoden für
 - Das Umwandeln von **LocalDateTime**-Objekten in Strings.
 - Das Umwandeln von Strings in **LocalDateTime**-Objekte.
 - Das Umwandeln von **Duration**-Objekten in Strings.
 - Das Umwandeln von Strings in **Duration**-Objekte.
- Die Enum-Klassen haben Methoden, um aus dem Namen des Vereins das zugehörige Enum zu ermitteln.

2. Schreiben Sie Methoden für das Auswerten der Daten. Schreiben Sie die Methoden **nur** unter Verwendung des Streaming-API's. Wenn Sie die Aufgabe ohne das Streaming-API lösen, wird nur die Hälfte der Punktzahl gewertet.
- Für die Klasse **Verein** die Methoden
 - fruehsterKurs()**: Berechnet den Kurs mit dem kleinsten Start-Datum. (**4 Pkt**)
 - gruppierenNachJahrMonat()**: Erzeugt eine Map, deren Schlüssel Zeichenketten mit Jahr und Monatsinformation sind und deren Werte Listen von Kursen sind. (**4Pkt**)
 - Die Methode **getLocalDateYearMonthOnly** der Klasse **Kurs** überführt ein **LocalDateTime**-Objekt in eine Zeichenkette mit dem Format "**yyyy-(M)M**" (Beispiel "2018-2")
 - Für die Klasse **Verband** die Methode:
 - gruppierenNachJahrMonat()**: Erzeugt eine Map, deren Schlüssel Zeichenketten mit Jahr und Monatsinformation sind und deren Werte Listen von Kursen sind. (**7Pkt**)
 - Tests zu dem Aufgabenteil finden Sie in der Klasse **VereinUndVerbandTest**.

Überblick über die Klassen und Packages von A2

Klassen, die von Ihnen ergänzt werden müssen

Klasse	Package	Implementieren/Ändern	Aufgabenteil
VereinReader	a2.modelreader	readData()	2.1
Verein	a2.model	gruppierenNachJahrMonat fruehsterKurs()	2.2
Verband	a2.model	gruppierenNachJahrMonat()	2.2

Klassen, die Sie nutzen müssen (sollten)

Klasse	Package	Verwenden	Aufgabenteil
Kurs	a2.donottouch.model	in VereinReader , Verein und Verband	2.1 und 2.2
Sportart	a2.donottouch.model.enum	in VereinReader	2.1
VereinsName	a2.donottouch.model.enum	in VereinReader	2.1
DateTimeUtil	a2.donottouch.utils	in VereinReader	2.1
GeneratorKonstanten	a2.donottouch.modelgenerator	in VereinReader um die Konstanten für die Trenner zu ermitteln	2.1

Alle anderen Klassen des von mir mitgelieferten Quelltexts

Klasse	Package	Verwendung
PathUtil	a2.donottouch.utils	wird im VereinReader verwendet
RandomUtil	a2.donottouch.utils	wird von den Generator-Klassen und dem DateTimeUtil verwendet

<i>VereinGenerator</i>	<i>a2.donottouch.modelgenerator</i>	wird vom <i>VereinWriter</i> und den Testklassen <i>VereinReaderTest</i> / <i>VereinUndVerbandTest</i> verwendet
<i>KursGenerator</i>	<i>a2.donottouch.modelgenerator</i>	wird vom <i>VereinGenerator</i> verwendet
<i>VereinWriter</i>	<i>a2.donottouch.modelgenerator</i>	erzeugt die Datei für den <i>VereinReader</i>

Tests zu A2.1 finden sich in der Klasse *VereinReaderTest*.

Tests zu A2.2 finden Sie in der Klasse *VereinUndVerbandTest*.

Sie dürfen die Tests auf keinen Fall modifizieren!!!!

A2.1 und A2.2 bauen *NICHT* aufeinander auf und können unabhängig voneinander bearbeitet werden.

Datum -Zeit: 24.01.2019 - Beginn 10:00 Uhr

Nettobearbeitungszeit TEIL A: 60 Min

Punkteverteilung

	Punkte	Erreicht
Teil A		
a1.geoReihelt	5	5
a1. geoReiheRek	5	5
a1. n_fuer_eps	5	5
a1.miniTabCalc	10	10
a1.merge	10	10
a2.VereinReader	17	17
a2.Verein		
fruehesterKurs	4	4
gruppierenNachJahrMonat	4	4
a2.Verband		
gruppierenNachJahrMonat	7	7
Teil A Gesamt	67	67 max 60

Hinweise

- Die maximal gewerteten Punkte für Teil A sind **60** Punkte. Alles was > 60 Punkte ist, wird nicht gewertet!
- Verwenden Sie unbedingt die in den Projekten vorgegebenen Methoden.

Schreiben Sie in jede Datei Ihre Matrikelnummer und wenn Sie mögen Ihren Namen.

Viel Erfolg! ☺

$$\sum A+B = M7 \quad \text{Note} \quad 15$$

WWD

27.1.2019

```

1 // MatNr.
2 package a1;
3
4 import java.util.ArrayList;
5 import java.util.HashMap;
6 import java.util.List;
7 import java.util.Map;
8
9 public class A1 {
10
11     // 5/5Pkt 26.1.2019 WND
12     public static double geoReiheIt(double a0, double q,
13         int n) {
14         if (n < 0 || a0 <= 0){ // a0 nichht prüfen
15             throw new IllegalArgumentException();
16         }
17
18         double sum = 0.0;
19         for (int k = 0; k <= n ;k++) {
20             sum += a0*Math.pow(q, k);
21         }
22         return sum;
23     }
24
25     // 5/5Pkt 26.1.2019 WND
26     public static double geoReiheRek(double a0, double q,
27         int n) {
28         if (n < 0 || a0 <= 0){
29             throw new IllegalArgumentException();
30         }
31
32         if (n == 0){
33             return a0*Math.pow(q, n);
34         }
35         else{
36             return a0*Math.pow(q, n) + geoReiheRek(a0, q, n-1
37         );
38     }
39
40     // 5/5Pkt 26.1.2019 WND
41     public static int n_fuer_eps(double eps, double a0,
42         double q) {
43         int n = 0;

```

Anlage zu Teil A

Pt2 27.1.2019 WND

```

42         while(Math.abs(geoReiheIt(a0,q,n)-(a0/(1-q)))>=
43             eps) {
44             n++;
45         }
46     }
47
48     // 10/10 Pkt 26.1.2019 WND
49     public static double[][][] miniTabCalc(double[][][] table)
{
50         int xSize = table.length;
51         int ySize = table[0].length;
52         double[][] result = new double[xSize][ySize+2];
53         for (int i = 0;i<table.length;i++) {
54             double rowsum = 0;
55             for(int j = 0;j< table[0].length;j++) {
56                 result[i][j] =table[i][j];
57                 rowsum+= result[i][j];
58             }
59             result[i][table[0].length] = rowsum;
60             result[i][table[0].length+1] = rowsum / (table
61 [0].length);
62         }
63     return result;
64
65     }
66
67     // 10/10 Pkt 26.1.2019 WND sehr schöne Lösung
68     public static Map<Integer, List<Integer>> merge(Map<
Integer, List<Integer>> m1, Map<Integer, List<Integer>> m2)
{
69         Map<Integer, List<Integer>> result = new HashMap<>(
    );
70         for (Map.Entry<Integer, List<Integer>> entry : m1.
entrySet()) {
71             result.computeIfAbsent(entry.getKey(), noVal
-> new ArrayList<Integer>());
72             result.get(entry.getKey()).addAll(entry.
getValue());
73         }
74         for (Map.Entry<Integer, List<Integer>> entry : m2.
entrySet()) {
75             result.computeIfAbsent(entry.getKey(), noVal
-> new ArrayList<Integer>());

```

```
76             result.get(entry.getKey()).addAll(entry.  
    getValue());  
77         }  
78     return result;  
79 }  
80  
81 }  
82
```

```
1 // MatNr.
2 package a2.modelreader;
3
4 import a2.donottouch.model.Kurs;
5 import a2.donottouch.model.enums.Sportart;
6 import a2.donottouch.model.enums.VereinsName;
7 import a2.donottouch.modelgenerator.GeneratorKonstanten;
8 import a2.donottouch.utils.DateTimeUtil;
9 import a2.donottouch.utils.PathUtil;
10 import a2.model.Verein;
11
12 import java.io.IOException;
13 import java.net.URL;
14 import java.nio.charset.StandardCharsets;
15 import java.nio.file.Paths;
16 import java.time.Duration;
17 import java.time.LocalDateTime;
18 import java.util.ArrayList;
19 import java.util.List;
20 import java.util.Scanner;
21
22 // Gesamt: 17/17 Pkt 26.1.2019 WND
23 public class VereinReader {
24
25     private static final String RESOURCE_NAME = "VereinsDaten";
26
27
28     public static List<Verein> readData() throws
29         IOException {
30         List<Verein> result = new ArrayList<>();
31         String vereineURI = Paths.get(PathUtil.
32             getRelativePathToCompiledProjectResources() +
33             RESOURCE_NAME).toAbsolutePath().toUri().toString();
34         Scanner scanner = new Scanner(new URL(vereineURI).
35             openStream(), StandardCharsets.UTF_8);
36         scanner.useDelimiter("\n");
37         while(scanner.hasNext()) {
38             String row = scanner.next();
39             if(!row.trim().isEmpty()) {
40                 String[] parts = row.split(
41                     GeneratorKonstanten.DELIM1);
42                 Verein rowVerein = new Verein(VereinsName.
43                     fromNameString(parts[0].trim()));
44                 for(int i = 1;i<parts.length;i++) {
```

```
39             String[] kursTeile = parts[i].split(
40                 GeneratorKonstanten.DELIM2);
41             Sportart spArt = Sportart.
42                 fromNameString(kursTeile[0].trim());
43             LocalDateTime date = DateTimeUtil.
44                 fromString(kursTeile[1].trim());
45             Duration dauer = DateTimeUtil.
46                 durationFromString(kursTeile[2].trim());
47             int teilehmer = Integer.valueOf(
48                 kursTeile[3].trim());
49             Kurs kurs = new Kurs(spArt, date, dauer,
50                 teilehmer);
51             rowVerein.add(kurs);
52         }
53     }
54 }
```

```
1 // MatNr.
2 package a2.model;
3
4 import a2.donottouch.model.Kurs;
5 import a2.donottouch.model.enums.VereinsName;
6
7 import java.util.*;
8 import java.util.stream.Collectors;
9
10 public class Verein {
11     private final VereinsName vereinsName;
12     private List<Kurs> kurse;
13
14     public Verein(VereinsName name) {
15         this.vereinsName = name;
16         kurse = new ArrayList();
17     }
18
19     public Verein(Verein vorlage){
20         this(vorlage.vereinsName);
21         this.addAll(vorlage.kurse);
22     }
23
24     public void add(Kurs kurs) {
25         this.kurse.add(kurs);
26     }
27
28     public void addAll(List<Kurs> kurse) {
29         this.kurse.addAll(new ArrayList<>(kurse));
30     }
31
32     @Override
33     public boolean equals(Object o) {
34         if (this == o) return true;
35         if (o == null || getClass() != o.getClass())
36             return false;
37         Verein verein = (Verein) o;
38         return vereinsName == verein.vereinsName &&
39                 Objects.equals(kurse, verein.kurse);
40
41     // 4/4 Pkt 26.1.2019 WND
42     public Map<String,List<Kurs>> gruppierenNachJahrMonat()
43     {
44         return kurse.stream().collect(Collectors.
```

```
44 groupingBy(Kurs::getLocalDateYearMonthOnly));
45     }
46
47     // 4/4 Pkt 26.1.2019 WND
48     public Kurs fruehesterKurs() {
49         return kurse.stream().min(Comparator.comparing(
50             Kurs::getStart)).orElse(null);
51     }
52
53
54     public List<Kurs> getKurse() {
55         return new ArrayList<>(kurse);
56     }
57
58     @Override
59     public int hashCode() {
60         return Objects.hash(vereinsName, kurse);
61     }
62
63     @Override
64     public String toString() {
65         return String.format("%s[%s]", vereinsName, kurse);
66     }
67
68     public String toFormattedString(String delim1, String
69         delim2) {
70         List<String> formatierteKurse = kurse.stream().map
71             (k -> k.toFormattedString(delim2)).collect(Collectors.
72                 toList());
73         String formatierteKurseString = formatierteKurse.
74             stream().collect(Collectors.joining(delim1));
75         return String.format("%s%s%s", vereinsName, delim1,
76             formatierteKurseString);
77     }
78 }
```

```
1 // MatNr.  
2 package a2.model;  
3  
4 import a2.donottouch.model.Kurs;  
5  
6 import java.util.ArrayList;  
7 import java.util.List;  
8 import java.util.Map;  
9 import java.util.stream.Collectors;  
10  
11 public class Verband {  
12  
13     List<Verein> vereine;  
14     public Verband(){  
15         this.vereine = new ArrayList<>();  
16     }  
17     public void add(Verein verein){  
18         this.vereine.add(verein);  
19     }  
20  
21     public void addAll(List<Verein> vereine){  
22         this.vereine.addAll(vereine);  
23     }  
24  
25     // 7/7 Pkt 26.1.2019 WND  
26     public Map<String,List<Kurs>> gruppierenNachJahrMonat()  
{  
27         return vereine.stream().flatMap(v -> v.getKurse().  
stream()).collect(Collectors.groupingBy(Kurs::  
getLocalDateYearMonthOnly));  
28     }  
29 }  
30
```

Datum -Zeit: 24.01.2019 – Beginn 10:00 Uhr

Nettobearbeitungszeit TEIL B: 60 Min

Punkteverteilung

	Punkte	Erreicht
Teil B		
1	4	4
2	6	6
3	4	1
4	2	2
5	2	2
6	3	1.5
7	3	3
8	2	2
9	3	3
10	4	3
11	5	4.5
12	3	3
13	4	3.5
14	10	6
15	4	3.5
16	3	2
17	4	4
18	1	1
19	2	2
Teil B Gesamt	69	57

Schreiben Sie auf jedes Blatt Ihre Matrikelnummer und wenn Sie mögen Ihren Namen.

Maximale erreichbare Punktzahl für Teil B sind 60 Punkte.

Teil B: Wissen und Verstehen

1. Geben Sie je ein Beispiel für Coercion und Typkompatibilität. (4Pkt)

double d = 3; // coercion ✓

Object o = Integer.valueOf(1) ✓ // Typkompatibilität

4/4

2. Gegeben die 2 Arrays *iAry* und *iAryClone* und 3 Zeilen, die Arrays auf Gleichheit prüfen.
(Summe 6Pkt)

6/6

```
Integer[][] iAry = {{1, 2, 3}, {7, 8, 9, 10}};  
Integer[][] iAryClone = iAry.clone();  
  
*1 System.out.println(iAry.equals(iAryClone));  
*2 System.out.println(Arrays.equals(iAry, iAryClone));  
*3 System.out.println(Arrays.deepEquals(iAry, iAryClone));
```

Nur der zweite und dritte Vergleich liefert *true*. Erklären Sie die Ergebnisse.

*1 liefert false (1Pkt): *iAry* und *iAryClone* sind zwei unterschiedliche Instanzen. Prüfung auf Identität. ✓

*2 liefert true (3Pkt): Es wird jedes Element auf einer Ebene auf *Identität* geprüft. *clone* auf Arrays nur eine Flache Kopie erzeugt, sind die beiden inneren Arrays die selben. (*equals* auf den inneren Arrays prüft wieder auf Identität). ✓

*3 liefert true (2Pkt): *deepEquals* prüft rekursiv jedes Array auf Wertgleichheit, durch rekursiven Aufruf von *Arrays.deepEquals*. Somit wird jeder Integer auf Wertgleichheit mit dem im anderen Array verglichen. Diese ist dann *clone* gegeben. ✓

3. Gegeben folgendes Codeschnipsel. (Summe 4Pkt)

1/4

```
Object[] oAry = new String[4];  
oAry[0] = 34;
```

- a. Welchen Fehler erzeugt das Codeschnipsel? (1Pkt)

ArrayStoreException 0/1

- b. Wann tritt dieser Fehler auf? (1Pkt) Wenn ein genereller statischer Typ enthält, und ein Objekt einer inkompatiblen Vererbungshierarchie zugewiesen wird.

- c. Welche Regel kennen Sie für generische Typen, die diese Art von Fehlern unterbindet? (2 Pkt)

1/2

Der Komponententyp von generischen Typen ist (ohne Wildcards) Typ invariant, es dürfen also keine Spez. mehr also genau der angegebene Komponententyp zugewiesen werden.

Was ist mit kovarianten WCT's?

Ungeschützte
Kovarianz
von Arrays

Laufzeit
0/1

4. Welche Technik wenden Sie an, um Objekte von Collection-Klassen ineinander umzuwandeln? (2 Pkt)

Eine neue Instanz des Zieltyps erzeugen, ~~wie~~ indem die alte Collection als Parameter im Konstruktor übergeben wird.
(Kopierkonstruktor) ✓

5. Welche Transformationen wendet der Compiler bei der nachfolgenden Zuweisung an? (2Pkt)

float d = new Float(2) + new Long(17);

1: Autounboxing von Basisdatentypen zu erhalten. ✓
2: Coercion um ~~sie aus dem~~ float und long in einen identischen Typen zu casten.

2/2

6. Nennen Sie 3 Regeln für Konstruktoren. (3Pkt)

Σ 1.5/3

- In der ersten Zeile wird immer der ~~super~~-Konstruktor der Superklasse aufgerufen. Einmal und implizit
- Wenn nicht anders angegeben wird der Default-Konstruktor der Superklasse aufgerufen. (Kann zu Fehlern führen wenn es keinen gibt) ✓
- Können nicht überschrieben werden. | kein Punkt
- sind statisch gebunden.

7. Gegeben das nachfolgende Codebeispiel. Ergänzen Sie die Initialisierung mit Hilfe einer Konstruktor-Kaskade. (3Pkt)

Σ 3/3

```
public class Tupel {  
  
    private Object elem1;  
    private Object elem2;  
  
    public Tupel(){  
        this(null); ✓  
    }  
    public Tupel(Object elem1){  
        this(elem1,null); ✓  
    }  
    public Tupel(Object elem1, Object elem2){  
        this.elem1 = elem1;  
        this.elem2 = elem2;  
    }  
}
```

8. Wenn Sie in Java einen diskreten endlichen Wertebereich modellieren wollen, welchen Datentyp verwenden Sie dann? (2Pkt)

Σ 2/2

Dazu werden Enum (Enumerationen) verwendet. ✓

9. Überführen sie den Methodenkopf der statischen Methode in eine statische generische Definition ohne Typebounds. (3 Pkt.)

✓ 3/3

```
public static Map<Integer, List<Double>>
    merge(Map<Integer, List<Double>> m1,
          Map<Integer, List<Double>> m2) {...}

public static <T> <V> Map<T, List<V>>
    merge (Map<T, List<V>> m1,
           Map<T, List<V>> m2) {...} ✓
```

- ? 10. Wandeln Sie die Definition des abstrakten Datentyps *Queue* in eine generische Definition um. Sie dürfen die Vorlage ergänzen/modifizieren. (4Pkt)

✓ 3/4

```
public class Queue<T> {
    private List<Object> intern;
    public Queue(){
        this.intern = new ArrayList<>();
    }

    public boolean isEmpty(){
        return intern.isEmpty();
    }

    public void enqueue(Object elem){
        this.intern.add(elem);
    }

    public Object dequeue() throws EmptyQueueException {
        if (isEmpty()) throw new EmptyQueueException();
        return intern.remove(0);
    }
}
```

11. Was wird in Java statisch gebunden? (5Pkt)

✓ 4.5/5

Konstruktor ✓
Alle Attribute ✓
final-Methoden ✓ ~~final Attribute~~
static-Methoden und Attribute
private-Methoden ✓

12. Was versteht man unter statischem Binden einer Methode. (3Pkt)

✓ 3/3

Es steht bereits zur Compilerzeit fest, welche Methodendefinition aufgerufen wird. Dies ist ausschließlich vom statischen Typs des Objekts abhängig. Es findet keine dynamische Methodensuche zur Laufzeit statt. ✓

13. Was versteht man unter dynamischem Binden einer Methode. (4Pkt)

Z 2.5/4

Es wird zur Laufzeit eine dynamische Methodensuche, abhängig vom dynamischen Typ des Objekts ausgeführt.
Beginnen wird dabei in der Klasse des dynamischen Typs. (objekt)

JVR?

14. Welche der folgenden Aussagen sind für die Sprache Java wahr, welche falsch? Markieren Sie die wahren Aussagen mit einem (w), die falschen mit einem (f). Nicht korrekte Markierungen geben 1 Punkt Abzug. Korrekte Markierung werden mit 1 Punkt gewertet. Keine Markierungen werden mit 0 Punkten gewertet. Ist die Gesamtpunktzahl negativ, wird auf 0 Punkte aufgerundet. (10Pkt)

Z 6/10

✓	<input checked="" type="checkbox"/> Ein Cast ändert den Objekttyp. (aus dem statischen)	+1
✓	<input checked="" type="checkbox"/> Der Typ <code>List<? extends Number></code> ist kompatibel zu dem Typ <code>List<Number></code> .	+1
✓	<input checked="" type="checkbox"/> Ein Objekt hat in der Regel mehr als einen Typ.	+1
F	<input checked="" type="checkbox"/> Abstrakte Klassen und Interfaces können Instanzvariablen enthalten. interfaces nor public static Variablen -1	-1
✓	<input checked="" type="checkbox"/> Ein Cast ist zwischen Geschwisterknoten einer Vererbungshierarchie möglich.	+1
W	<input checked="" type="checkbox"/> Generische Wildcardtypen sind gültige Javatypen.	+1
F	<input checked="" type="checkbox"/> Wenn eine Klasse eine Methode einer Superklasse überschreibt, darf die Klasse die Methode nicht überladen. Es wird dann die überschreibende Methode überladen.	-1
✓	<input checked="" type="checkbox"/> Zwei Methoden haben unterschiedliche Signaturen, wenn Sie sich nur im Typ des Rückgabewertes unterscheiden.	+1
✓	<input checked="" type="checkbox"/> Custom-Konstruktoren erzwingen die Implementierung von Konstruktoren in Subklassen. (nicht, wenn trotzdem ein default konstruktur vorhanden ist.)	+1
✓	<input checked="" type="checkbox"/> Der Typ <code>Integer[]</code> ist kompatibel zum Typ <code>Number[]</code> .	+1

15. Gegeben die nachfolgenden Klassendefinitionen. (4Pkt)

Σ 3.5/4

```
class Person {  
    private String name;  
  
    public Person(String name){  
        this.name = name;  
    }  
}  
  
class Student extends Person {  
    private String name;  
    private String matNr;  
  
    public Student(String name, String matNr){  
        super(name);  
        this.matNr = matNr;  
    }  
  
    @Override  
    public String toString(){  
        return "Student{" + name + ", " + matNr + '}';  
    }  
}  
  
public class PersonStudentMain {  
    public static void main(String[] args) {  
        1: Student newBee = new Student("NewBee", "88");  
        2: System.out.println(newBee);  
    }  
}
```

Das **main** der Klasse **PersonStudentMain** erzeugt die Ausgabe **Student{null, "88"}**. Erklären Sie das Ergebnis.

In Zeile 1 wird der Konstruktor in **Student** aufgerufen.
Dieser übergibt den Namen an den Konstruktor in **Person**.
Dort wird der Name in die private Instanzvariable der Klasse **Person** hinterlegt.

Die Matrikelnummer wird an die private Instanzvariable **matNr** in **Student** gebunden.

In Zeile 2 wird **toString** in **Student** aufgerufen. Dabei werden die beiden Instanzvariablen von **Student** ausgegeben. Da Name in **Student** nie gesetzt wurde ist der Wert null. Dieses Verhalten entsteht durch das Überschatten der **Instanzvariablen**, "name".

16. Welche Exceptions (Kategorien nennen) (Summe 3Pkt)

- müssen Sie in Java behandeln oder weiterreichen?
Exception (handelt) checked
- müssen / sollten Sie in Java nicht behandeln?
Runtime Exception (unhandled)
- dürfen Sie in Java auf keinen Fall behandeln?
Errors

2/3

hierbei ist statischer Binden

17. Gegeben die Registrierung eines *ChangeListeners* auf Selektionsänderungen für das Modell einer *TreeView*. Übersetzen Sie die Registrierung mittels einer anonymen inneren Klasse in eine Registrierung mittels Lambda-Ausdruck. (4Pkt)

```
tree.getSelectionModel().selectedItemProperty().  
    addListener(new ChangeListener<TreeItem<Object>>() {  
        @Override  
        public void changed(  
            ObservableValue<? extends TreeItem<Object>> observable,  
            TreeItem<Object> oldValue,  
            TreeItem<Object> newValue) {  
                // ...  
            }  
        );  
    ... .add Listener (ActionEvent e) > changed(e);
```

changed

add Listener (Observable, oldValue, newValue) → { ... }) ✓

24/4

18. Wie heißt der *ChangeListener* im MVC Pattern? (1Pkt)

1/1

View
Controller

19. Welche Bedingung muss ein Interface erfüllen, damit die Ersetzung einer anonymen inneren Klasse durch einen Lambda-Ausdruck möglich ist? (2Pkt)

2/2

Es darf nur eine Methodendeklaration enthalten.

Methodendeklaration enthalten. ✓