Quiz zur Nacht der Wünsche

Wiederholung ausgewählter Vorlesungsthemen mit Gewinnmöglichkeit (siehe Übungsblatt)
Offline

Test und Assessment – Druckansicht

Quiz zur Nacht der Wünsche

Datum: Heute, 16:40 Maximale Punktezahl: 25

Frage 1 - Compile & Run (1 Punkt) [ID: 1808963]

Die Datei `Hello.java` ist im Verzeichnis `/home/progra/vl1` gespeichert. Sie wollen die Klasse kompilieren und ausführen. Vervollständigen Sie dazu die folgenden Befehle.

```
/home %
cd




 progra/vl1
/home/progra/vl1 %
javac




 Hello.java
/home/progra/vl1 %
ls o




Hello.class Hello.java
/home/progra/vl1 %
java



 Hello
```

Frage 2 - Hello World! (1 Punkt) [ID: 1808964]

Vervollständigen Sie die folgende Java-Klasse in der Datei `Hello.java`, sodass sie eine `main`-Methode enthält, die Ihre individuelle Codezahl, die Sie am 12.12.25 per Uni-Mail von uns erhalten haben, ausgibt.

Achtung: Ob Sie diese Aufgabe (teilweise) richtig gelöst haben, kann das Ilias leider nicht anzeigen – das Ilias wird für die Aufgabe immer 0 Punkte anzeigen, wir werden Ihnen den Punkt für die Verlosung aber manuell geben, wenn die Antwort korrekt ist. Falls Sie keine E-Mail bekommen haben, schreiben Sie uns von der Uni-Mail aus (nicht via Ilias, das ist ein separates Mail-System) an progra@cs.hhu.de.

```
class
Hello
{
   static
void
main() {
       System.
out
.println(
28221312345673915 o
)
;
    }
}
```

Frage 3 - FizzBuzz - Ausgabe (1 Punkt) [ID: 1808965]

Diese Aufgabe ist eine typische Aufgabe aus Bewerbungsgesprächen, um grundlegende Programmierfähigkeiten zu prüfen.

Schreiben Sie ein Programm `FizzBuzz`, welches alle Zahlen **von 1 bis 42** durch Kommata getrennt auf der Standardausgabe ausgibt.
Beachten Sie hierbei die folgenden Regeln:

- Falls die Zahl ein Vielfaches von 3 ist, soll anstelle der Zahl das Wort „Fizz“ ausgegeben werden.
- Falls die Zahl ein Vielfaches von 5 ist, soll anstelle der Zahl das Wort „Buzz“ ausgegeben werden.
- Ist die Zahl sowohl ein Vielfaches von 3 als auch von 5, soll anstelle der Zahl das Wort „FizzBuzz“ ausgegeben werden.
- Als Beispiel ist hier die Ausgabe des Programms für die Zahlen von 1 bis 15 angegeben:

```
1,2,Fizz,4,Buzz,Fizz,7,8,Fizz,Buzz,11,Fizz,13,14,FizzBuzz
```

Achten Sie darauf, in Ihrem Quelltext die **idiomatische** Schleifenart zu benutzen. Geben Sie die **Ausgabe** Ihres Programms an (nicht den Quelltext!).

Ausgabe:

```
1,2,Fizz,4,Buzz,Fizz,7,8,Fizz,Buzz,11,Fizz,13,14,FizzBuzz,16,17,Fizz,19,Buzz,Fizz,22,23,Fizz,Buzz,26,Fizz,28,29,FizzBuzz,31,32,Fizz,34,Buzz,Fizz,37,38,Fizz,Buzz,41,Fizz ode
```

Frage 4 - Lineare Suche (1 Punkt) [ID: 1808968]

Vervollständigen Sie die folgende `List`-Klasse, sodass die Methode `linearSearch` den übergebenen Wert mithilfe einer linearen Suchen in der Liste sucht. Die Methode soll den ersten Index, an dem das Element gefunden wurde, zurückgeben; das erste Listenelement hat den Index 0. Falls das Element nicht in der Liste ist, soll eine `IllegalArgumentException` geworfen werden.

```
public class List {


    private class Node {
        private int value;
        private Node next;


        private Node(int element, Node next) {
            this.value = element;
            this.next = next;
        }
    }


    private Node head;


    public List() {
        head = null;
    }


    public void add(int value) {
        head = new Node(value, head);
    }


    public int linearSearch(int needle) {
        Node current =
```
head
```


;
        int index =
```
0
```


;
        while(current !=
```
null
```


) {
            if(current.
```

```
value
```

```
 == needle) {
                return
```

```
index
```

```
;
            }
            current = current.
```

```
next
```

```
;
```

```
index
```

```
++;
        }
```

```
throw
```

```
 new IllegalArgumentException();
    }

}
```

Frage 5 - Heinenacci-Zahlen (1 Punkt) [ID: 1808973]

Die *n*-te Heinenacci-Zahl (ja, die haben wir uns ausgedacht) ist definiert als

$$f(n) = \begin{cases} f(n-1) + 2nf(n-2) + 1797 & n > 1 \\ 1 & \text{sonst.} \end{cases}$$

Füllen Sie die Lücken, sodass die Java-Methode diese Formel umsetzt.

```
private static int heinenacci(int
```
n
```
) {
    if(
```
n
```
 > 1) {
        return
```
heinenacci
```
(n - 1) + 2 * n *
```
heinenacci
```
(n - 2) + 1797;
    }
    return
```
1
```
;
}
```

Die 2. Heinenacci-Zahl (n=2) ist 1802 .

Frage 6 - Primitive Datentypen (1 Punkte) [ID: 1808969]

Aus der Übungsaufgabe *Referenzen* wissen wir, dass Änderungen an den Werten von Methodenparametern auch außerhalb der Methode Auswirkungen haben können. Dies ist nur dann möglich, wenn der Wert des Parameters nicht primitiv ist (und damit nicht im Stack abgelegt ist – auf dem Stack liegt „nur“ eine Referenz auf das Objekt im Heap). Um versehentliche, „globale“ Änderungen zu vermeiden, sollten wir also wissen, welche Datentypen (nicht) primitiv sind.

Ordnen Sie richtig zu:

primitiver Datentyp	passt zu	double	*(0.1 Punkte)*
primitiver Datentyp	passt zu	boolean	*(0.1 Punkte)*
primitiver Datentyp	passt zu	int	*(0.1 Punkte)*
primitiver Datentyp	passt zu	long	*(0.1 Punkte)*
Objekttyp	passt zu	String	*(0.1 Punkte)*
Objekttyp	passt zu	int[]	*(0.1 Punkte)*
Objekttyp	passt zu	String[]	*(0.1 Punkte)*
Objekttyp	passt zu	boolean[][]	*(0.1 Punkte)*
Objekttyp	passt zu	Color	*(0.1 Punkte)*
Objekttyp	passt zu	List<String>	*(0.1 Punkte)*

Frage 7 - Heap und Stack (1 Punkt) [ID: 1808970]

Geben Sie für die folgenden Werte jeweils an, wo sie gespeichert sind:

Stack	passt zu	Wert einer lokalen int-Variable	*(0.1 Punkte)*
Stack	passt zu	Wert eines Parameters vom Typ double	*(0.1 Punkte)*
Stack	passt zu	Referenz auf einen String, die in einer lokalen Variablen gespeichert ist	*(0.1 Punkte)*
Heap	passt zu	Inhalt eines String-Objekts	*(0.1 Punkte)*
Heap	passt zu	erstes Element eines int-Arrays	*(0.1 Punkte)*
Heap	passt zu	erstes Element eines String-Arrays	*(0.1 Punkte)*
Heap	passt zu	Daten, die in einer verketteten Liste gespeichert sind	*(0.1 Punkte)*
Heap	passt zu	Wert einer Instanzvariablen vom Typ double	*(0.1 Punkte)*
Heap	passt zu	Wert einer Instanzvariablen vom Typ String	*(0.2 Punkte)*

Frage 8 - Gleichheit (1 Punkt) [ID: 1808971]

Wir haben gelernt, dass wir bei Objekten zwei Arten von Gleichheit unterscheiden müssen:

- Sind die Objekte identisch? (Zeigen zwei Referenzen auf ein und dasselbe Objekt im Heap?)
- Sind die Objektinhalte gleich? (Wobei jede Klasse selbst definieren kann, wann zwei Instanzen gleich sind.)

Ergänzen Sie den Code, sodass er tut, was in den Kommentaren steht:

```
String s1 = new String("Hello");

String s2 = "Hello";



// Sind die Objekte identisch?

System.out.println(s1
```
== oder =
```
s2
```
) o
```

;

```
// Sind die Stringinhalte gleich?

System.out.println(s1
```
.equals(
```
s2
```
))
```

;

(Passen Sie mit der Anzahl der Klammern auf.)

Frage 9 - Referenzen (1 Punkt) [ID: 1808972]

Wir wollen uns nochmal genauer anschauen, wie das mit dem Umbiegen von Referenzen funktioniert. Was gibt dieser Code aus? *(Machen Sie sich im Zweifel eine Skizze, auf welche Speicherstellen im Heap die Referenzen a und b nach jeder Zeile zeigen).*

```
int[] a = {5, 6};
int[] b = {7, 8};
a = b;
b[0] = 1;
System.out.println(a[0]);
System.out.println(a[1]);
System.out.println(b[0]);
System.out.println(b[1]);
```

1

8

1

8

Frage 10 - Sortieren & Kopieren & Heap (1 Punkt) [ID: 1838415]

Gegeben ist folgender Code. Gehen Sie davon aus, dass die LinkedList analog zur Vorlesung funktioniert.

Achtung: Bei der Klasse Person muss gescrollt werden. (Ilias ...)

```java
enum Geschlecht {
    D, M, W, X;
}
```

```java
class Person implements Comparable<Person> {
    private String name;
    private Geschlecht geschlecht;

    Person(String name, Geschlecht geschlecht) {
        this.name = name;
        this.geschlecht = geschlecht;
    }

    void setName(String name) {
        this.name = name;
    }

    void setGeschlecht(Geschlecht geschlecht) {
        this.geschlecht = geschlecht;
    }

    @Override
    public String toString() {
        return String.format("%s {%s}", name, geschlecht);
    }

    @Override
    public int compareTo(Person other) {
        // aufsteigend nach Name
        return this.name.compareTo(other.name);
    }
}
```

```java
class Verwaltung {
    private LinkedList<Person> personen = new LinkedList<>(

    Verwaltung(Person[] personen) {
        // sortiert Array gemäß natürlicher Sortierung
        java.util.Arrays.sort(personen);
        for(Person person: personen) {
            this.personen.add(person); // hinten einfügen
        }
    }

    @Override
    public String toString() {
        return personen.toString();
    }
}
```

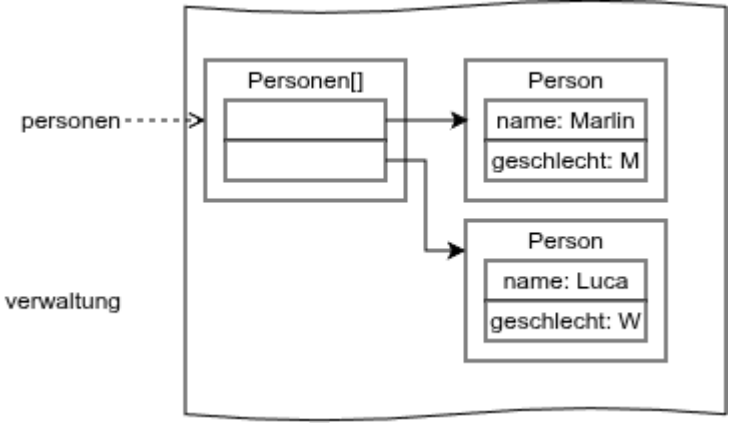
```java
static void main() {
    Person[] personen = {
        new Person("Marlin", Geschlecht.M),
        new Person("Luca", Geschlecht.W)
    };

    // Stelle 1
    Verwaltung verwaltung = new Verwaltung(personen);
    // Stelle 2

    personen[1].setGeschlecht(Geschlecht.D);

    System.out.println(verwaltung);
    System.out.println(java.util.Arrays.toString(personen))
}
```

An Stelle 1 sieht der Heap vereinfacht so aus:



Wenn Sie sich gut für die Klausur vorbereiten wollen, beantworten Sie folgende Fragen, bevor Sie die Zuordnungsfrage unten beantworten:

- Skizzieren Sie den Heap-Zustand an Stelle 2; beachten Sie insbesondere:
  - Wohin zeigt die Instanzvariable des Objekts verwaltung?
  - Welche Objekte werden von dieser Liste referenziert?
  - In welcher Reihenfolge sind die Personen im Array und in der Liste gespeichert?
- Welche Ausgabe machen die println-Statements?

Ordnen Sie den println-Statements Ihre Ausgabe zu:

System.out.println(verwaltung);	passt zu	[Luca {W}, Marlin {D}]	*(0.5 Punkte)*
System.out.println(java.util.Arrays.toString(personen));	passt zu	[Luca {W}, Marlin {D}]	*(0.5 Punkte)*

Frage 11 - Fahrradladen (1 Punkte) [ID: 1808974]

Für einen Fahrradladen soll eine kleine Produktverwaltung geschrieben werden.

Es soll eine öffentliche Klasse `Fahrrad` geben, die ein Fahrrad repräsentiert. Jedes Fahrrad hat zwei Räder, eine beliebige Farbe und kann einen Motor haben oder nicht. Farbe und Vorhandensein eines Motors sollen beim Konstruktoraufruf angegeben werden und später über Setter geändert werden können. Es soll außerdem einen Konstruktor ohne Argumente geben, der ein schwarzes Fahrrad ohne Motor erstellt. Anzahl der Räder, Farbe und Vorhandensein eines Motors sollen über Getter abgefragt werden können. Konstruktoren, Setter und Getter sollen paket-privat sein; alle Instanzvariablen sollen minimale Sichtbarkeit haben.

Weiterhin gibt es eine Klasse `Fahrradladen`, in der Sie in einer `main`-Methode ein Array mit genau zwei Fahrrädern anlegen. Das erste Fahrrad soll Blau sein und einen Motor haben, das zweite Fahrrad schwarz und ohne Motor.

Ergänzen Sie den folgenden Code, sodass er die Anforderungen erfüllt. Die in der Vorlesung gezeigte Klasse `Color` wird als vorhanden angesehen.

```
class Fahrrad {



private 




 boolean hatMotor;

private 




 Color farbe;
    private static final int ANZAHL_RAEDER = 2;


    Fahrrad(Color farbe, boolean
hatMotor 




) {
        this.
farbe 
 = farbe;

this 




.hatMotor = hatMotor;
    }


    Fahrrad() {
        farbe = Color.BLACK;
        hatMotor =
false 
```

```
;
    }

    Color getFarbe() {

return

 farbe;
    }

boolean

 hatMotor() {
        return
hatMotor oder t

;
    }

    int getAnzahlRaeder() {

return

 ANZAHL_RAEDER;
    }

void
```

```
setFarbe(Color farbe) {
       this.
```
`farbe`
```
= farbe;
   }

   void setHatMotor(boolean hatMotor) {
```
`this`
```
.hatMotor = hatMotor;
    }

}
```

```
class Fahrradladen {




static 
```

```
 void main() {
```

```
Fahrrad[] 
```

```
 raeder = 
```

```
new 
```

```
 Fahrrad[2];
```

```
        raeder[
```

```
0 
```

```
] = 
```

```
new 
```

```
 Fahrrad(Color.BLUE, 
```

```
true 
```

```
);
        raeder[
```

```
1 
```

```
] =
```

`new`

```
 Fahrrad();
    }

}
```

Frage 12 - Konstruktoren (1 Punkt) [ID: 1808975]

Gegeben seien die folgenden (nutzlosen) Klassen:

```
public class KlasseA {

}
```

```
public class KlasseB {
    public KlasseB() {
        // ...
    }
}
```

```
public class KlasseC {
    public KlasseC() {
        // ...
    }

    public KlasseC(int a) {
        // ...
    }
}
```

```
public class KlasseD {
    public KlasseD(int a) {
        // ...
    }
}
```

```java
public class KlasseE extends KlasseB {
    public KlasseE(int a) {
        // ...
    }
}
```

Auch wenn wir selbst keinen Konstruktor für eine Klasse erstellen, können wir Objekte einer Klasse erstellen.

Wie viele Konstruktoren haben die obigen Klassen jeweils?

Anzahl Konstruktoren von `KlasseA`: 1

Anzahl Konstruktoren von `KlasseB`: 1

Anzahl Konstruktoren von `KlasseC`: 2

Anzahl Konstruktoren von `KlasseD`: 1

Anzahl Konstruktoren von `KlasseE`: 1

Frage 13 - Performance (1 Punkt) [ID: 1808976]

In der Praxis ist es wichtig, dass Sie die Vor- und Nachteile verschiedener Datenstrukturen und Algorithmen kennen. Vor allem bei der Verarbeitung großer Datenmengen (z. B. beim Durchsuchen eines Kundenstamms eines Energieversorgers) ist es relevant, die Geschwindigkeit verschiedener Lösungen grob richtig einordnen zu können. Vervollständigen Sie die Sätze so, dass sie richtig sind.

Das Zugreifen auf einen bestimmten Array-Index ist schneller als der Zugriff auf ein Element an einem bestimmten Index in einer einfach verketteten Liste; das liegt daran, dass bei einem Array alle Elemente direkt hintereinander im Heap liegen, wohingegen bei Listen den next-Referenzen gefolgt werden muss. Anders als Arrays können aber Listen zur Laufzeit länger oder kürzer werden, wodurch Listen gut für Einsatzfälle geeignet sind, bei denen vorab die Menge der zu speichernden Daten nicht bekannt ist.

Mergesort ist bei großen Datenmengen, die sortiert werden sollen, meistens wesentlich schneller als Insertion Sort.

In einem sortierten Integer-Array lässt sich mit binärer Suche schneller feststellen, ob ein bestimmter Wert im Array vorhanden ist, als in einem unsortierten Array unter Verwendung von linearer Suche.

Frage 14 - binarySearch (1 Punkt) [ID: 1838433]

Die im JDK enthaltene Methode `java.util.Arrays.binarySearch(int[], int)` sucht im übergebenen Array mithilfe binärer Suche nach der übergebenen Zahl. Wird die Zahl gefunden, wird ihr Index zurückgegeben. Falls die Zahl nicht gefunden wird, wird ein negativer Wert zurückgegeben.

Erklären Sie, warum der Ausdruck `java.util.Arrays.binarySearch(new int[]{8,-2,1,5,6}, 8)` den Wert `-6` hat, obwohl die 8 im Array enthalten ist.

.

Binäre Suche geht davon aus, dass das Array, in dem gesucht wird, sortiert ist . Wenn das Array wie in diesem Beispiel nicht sortiert ist , sucht die binäre Suche die 8 an der falschen Stelle: Im ersten Schritt wird nämlich geprüft, ob die 8 kleiner oder größer als 8 ist. Da sie kleiner als 8 ist, wird in der Hälfte 5 6 weitergesucht. Dort wird die 8 dann aber nicht gefunden , weshalb die Methode eine negative Zahl zurückgibt.

Frage 15 - Insertion Sort (1 Punkt) [ID: 1808977]

In Java gibt es zwar schon fertige Möglichkeiten, um Listen und Arrays zu sortieren, aber irgendjemand muss es einmal für Java programmiert haben. Wir haben uns in der Vorlesung verschiedene Sortierverfahren angesehen. Eines dieser Verfahren war Insertion Sort.

Die Zahlenfolge 9 7 1 5 4 2 soll mit Insertion Sort sortiert werden. Ordnen Sie die folgenden Zahlenfolgen an, sodass sie den Zwischenschritten von Insertion Sort entsprechen.

9 7 1 5 4 2

7 9 1 5 4 2

1 7 9 5 4 2

1 5 7 9 4 2

1 4 5 7 9 2

1 2 4 5 7 9

Frage 16 - DNA (1 Punkte) [ID: 1808978]

In der Biologie arbeitet man mit DNA. Wir haben zwar keine Ahnung von DNA, aber vervollständigen Sie doch bitte die Methode `containsAC` folgender DNA-Klasse, sodass sie genau dann `true` zurückgibt, wenn es in der DNA(-Liste) die beiden Elemente A und C direkt hintereinander gibt.

```
public class DNA {


    private class Node {
        private char element;
        private Node next = null;


        private Node(char element, Node next) {
            this.element = element;
            this.next = next;
        }
    }


    private Node head = null;


    public void insert(char element) {
        head = new Node(element, head);
    }


    public boolean containsAC() {
        if (head == null
||




head.next == null) {
            return
false




;
        }


Node




 current =
head
```

```
;
        while(current.next
!=
 null) {
            if(current.element == 'A'
&&
 current.next.element ==
'C'
) {
                return
true
;
            }
            current = current.
next
;
        }
        return
false
;
```

```
    }
}
```

Frage 17 - Generics (1 Punkt) [ID: 1808980]

Wir haben generische Datentypen eingeführt, damit wir

- eine generische Listen-Klasse erstellen können, die beliebige Objekte speichern kann,
- wir aber trotzdem bei der Deklaration angeben können, welche Typen in der Liste gespeichert werden können sollen (und damit der Compiler uns eine Fehlermeldung anzeigen kann, falls wir unpassende Objekte in der Liste speichern wollen).

Was stimmt über generische Datentypen?

- ☐ Die Typvariable muss immer `T` heißen. *(Ausgewählt = 0 Punkte, Nicht ausgewählt = 0.25 Punkte)*
- ☐ Auf der linken Seite einer Deklaration darf man auch `<>` verwenden, z. B. `List<> list = new List<Integer>;`. *(Ausgewählt = 0 Punkte, Nicht ausgewählt = 0.25 Punkte)*
- ☐ Auch primitive Datentypen dürfen für die Typvariable eingesetzt werden, z. B. `new List<int>;`. *(Ausgewählt = 0 Punkte, Nicht ausgewählt = 0.25 Punkte)*
- ☑ Generische Datentypen stellen Typsicherheit sicher, d. h. in einer `List<String>` kann ich nur Strings speichern, aber z. B. keine Studi-Objekte. *(Ausgewählt = 0.25 Punkte, Nicht ausgewählt = 0 Punkte)*

Frage 18 - final (1 Punkt) [ID: 1808979]

Gegeben sei folgende Klasse. Gehen Sie davon aus, dass die Klasse `List` der einfach verketteten Liste aus der Vorlesung entspricht.

```java
public class Summierer {

    private final List<Integer> summanden = new List<>();

    public void summandHinzufuegen(int summand) {
        summanden.add(summand);
    }

    public int summe() {
        int ergebnis = 0;
        for(int i = 0; i < summanden.size(); i++) {
            ergebnis += summanden.get(i);
        }
        return ergebnis;
    }

}
```

Füllen Sie die Lücken im Text:

Das Statement `summanden.add(summand);` verändert den Inhalt der Liste summanden . Man könnte glauben, dass die Methode `summandHinzufuegen` so nicht funktioniert, weil `summanden` als final deklariert ist. Die Methode funktioniert aber trotzdem, denn final ist nur die Referenz , die in der Instanzvariablen `summanden` gespeichert ist, nicht aber der Inhalt des List-Objekts. Falls man verhindern möchte, dass auch der Inhalt der Liste geändert werden kann, müsste man eine Klasse für unveränderliche (immutable) Listen schreiben, die keine Methoden anbietet, um ihre Inhalte zu ändern.

Frage 19 - Klassen (1 Punkt) [ID: 1808982]

Welche Aussagen über Klassen stimmen?

- ☑ Eine Klasse kann mehrere Interfaces implementieren (class KlasseA implements InterfaceA, InterfaceB). *(Ausgewählt = 0.2 Punkte, Nicht ausgewählt = 0 Punkte)*
- ☐ Eine Klasse kann direkt von mehreren Klassen erben (class Klasse C extends KlasseA, KlasseB). *(Ausgewählt = 0 Punkte, Nicht ausgewählt = 0.2 Punkte)*
- ☑ Eine Klasse B kann sowohl Oberklasse einer Klasse C, als auch Unterklasse einer Klasse A sein. *(Ausgewählt = 0.2 Punkte, Nicht ausgewählt = 0 Punkte)*
- ☑ Eine Klasse ist abstrakt, wenn sie mindestens eine Methode ohne Implementierung besitzt. *(Ausgewählt = 0.2 Punkte, Nicht ausgewählt = 0 Punkte)*
- ☑ Methoden, die (in einer Klasse oder einem Interface) keine Implementierung haben, heißen „abstrakte Methoden“. *(Ausgewählt = 0.2 Punkte, Nicht ausgewählt = 0 Punkte)*

Frage 20 - Noch mehr Fragen zu Klassen (1 Punkt) [ID: 1808983]

Welche Aussagen über Klassen stimmen?

- ☐ Wenn eine Klasse mit `public class Fahrrad extends Fahrzeug` beginnt, dann ist `Fahrrad` eine Oberklasse von `Fahrzeug`. *(Ausgewählt = 0 Punkte, Nicht ausgewählt = 0.2 Punkte)*
- ☑ Wenn eine Klasse mit `public class Fahrrad extends Fahrzeug` beginnt, dann ist `Fahrrad` eine Unterklasse von `Fahrzeug`. *(Ausgewählt = 0.2 Punkte, Nicht ausgewählt = 0 Punkte)*
- ☑ Wenn eine Klasse mit `public class Fahrrad extends Fahrzeug` beginnt, dann wird beim Erstellen eines `Fahrrad`-Objekts auch ein Konstruktor von `Fahrzeug` aufgerufen (im Zweifel der Default-Konstruktor von `Fahrzeug`, wenn `Fahrzeug` keinen Konstruktor explizit definiert). *(Ausgewählt = 0.2 Punkte, Nicht ausgewählt = 0 Punkte)*
- ☑ Wenn eine Klasse mit `public class Fahrrad {` beginnt, dann erbt sie von der Klasse `Object`. *(Ausgewählt = 0.2 Punkte, Nicht ausgewählt = 0 Punkte)*
- ☑ Die Klasse Object enthält Standardimplementierungen von `equals` und `toString`. *(Ausgewählt = 0.1 Punkte, Nicht ausgewählt = 0 Punkte)*
- ☑ Die Standard-Implementierung von `equals` in der Klasse Object vergleicht standardmäßig nur die Referenzen, verhält sich also wie ==. *(Ausgewählt = 0.1 Punkte, Nicht ausgewählt = 0 Punkte)*

Frage 21 - Rennspiel (1 Punkt) [ID: 1808985]

Wir haben ein kleines Rennspiel (bzw. einen Ansatz dafür, sehr spannend ist das Rennen noch nicht) in Java programmiert, das Sie in den vorgegebenen Dateien zum Übungsblatt finden.

Leider gibt es bei dem Spiel noch Compilerfehler. Geben Sie für alle Programmierfehler an, in welcher Klasse und in welcher Zeile sie vorliegen und wie die korrigierte Codezeile aussieht; falls eine Zeile entfernt werden muss, tragen Sie `-leer-` als Korrektur ein. Geben Sie keine Folgefehler an, die durch Korrektur des eigentlichen Fehlers behoben sind.

Klasse: PolarPoint

Zeilennummer: 21

korrigierte Codezeile*: public PolarPoint subtract(Point otherPoint) {

Klasse: PolarPoint

Zeilennummer: 28

korrigierte Codezeile*: -leer-

Klasse: Race

Zeilennummer: 26

korrigierte Codezeile*: for(Racecar car: cars) {

Klasse: Racecar

Zeilennummer: 2

korrigierte Codezeile*: private PolarPoint velocity = new PolarPoint(0, 0);

Klasse: Racecar

Zeilennummer: 44

korrigierte Codezeile: public String toString() {

Klasse: Racecar

Zeilennummer: 49

korrigierte Codezeile*: public void draw(Draw canvas) {

* vollständige Zeile ohne Einrückung; Leerzeichensetzung vom Originalcode bitte beibehalten

Frage 22 - Records (1 Punkt) [ID: 1838371]

Records sind eine Kurzzschreibweise für Klassen, deren Felder final sind und Gleichheit über die Gleichheit der Felder definieren.

Gegeben sei folgende Klasse:

```
class Paket {
    private final int gewicht;
    private final String adresse;

    public Paket(int gewicht, String adresse) {
        this.gewicht = gewicht;
        this.adresse = adresse;
    }

    public int gewicht() {
        return gewicht;
    }

    public String adresse() {
        return adresse;
    }
}
```

Und folgendes Record:

```java
record Paket(int gewicht, String adresse) {}
```

Kreuzen Sie die Korrekten aussagen an:

- [x] Die Signatur der Konstruktoren der Klasse und des Records sind gleich. *(Ausgewählt = 0.1 Punkte, Nicht ausgewählt = 0 Punkte)*
- [x] Sowohl Klasse als auch Record besitzen eine Instanz-Methode String adresse(). *(Ausgewählt = 0.1 Punkte, Nicht ausgewählt = 0 Punkte)*
- [x] Betrachten wir das Record Paket: (new Paket(1, "foo")).toString() hat den Wert "Paket[gewicht=1, adresse=foo]". *(Ausgewählt = 0.1 Punkte, Nicht ausgewählt = 0 Punkte)*
- [x] Betrachten wir die Klasse Paket: (new Paket(1, "foo")).toString() hat einen Wert, der irgendwie wie Paket@26a1ab54 aussieht. *(Ausgewählt = 0.1 Punkte, Nicht ausgewählt = 0 Punkte)*
- [x] Betrachten wir das Record Paket: (new Paket(1, "foo")).equals(new Paket(1, "foo")) ist true. *(Ausgewählt = 0.1 Punkte, Nicht ausgewählt = 0 Punkte)*
- [x] Betrachten wir die Klasse Paket: (new Paket(1, "foo")).equals(new Paket(1, "foo")) ist false. *(Ausgewählt = 0.5 Punkte, Nicht ausgewählt = 0 Punkte)*

Frage 23 - E-Mailversand (1 Punkte) [ID: 1808986]

(Die Aufgabe basiert auf einen Teil einer alten Klausuraufgabe. In der Klausur war die Aufgabe allerdings kein Lückentext.)

In dieser Aufgabe sollen Sie ein System zum Versand von E-Mails entwickeln.
Grundlage hierfür ist das Interface `Sendable`, welches die Anforderungen an eine verschickbare E-Mail beschreibt:

```java
public interface Sendable {
    // Gibt die Absenderadresse zurück.
    String getSenderAddress();

    // Gibt die Empfaengeradresse zurück.
    String getReceiverAddress();

    // Gibt den Betreff der Mail zurück.
    String getSubject();

    // Gibt den Inhalt (Text) der Mail zurück.
    String getContent();

    // Gibt die Größe der Mail (Anzahl Textzeichen) zurück.
    int getSize();
}
```

Implementieren Sie eine paket-private Klasse `Mail`, welche eine E-Mail beschreibt. Eine E-Mail besteht aus den vier Komponenten Absender, Empfänger, Betreff und Text. Diese sollen als Objektvariablen mit **minimaler** Sichtbarkeit gespeichert werden. Implementieren Sie außerdem einen Konstruktor, der für jede dieser Variablen einen Wert übergeben bekommt und die Instanzvariablen entsprechend initialisiert. Ihre Klasse soll `Sendable` sinnvoll implementieren. Geben Sie in `getSize()` die Länge des Betreffs plus die Länge des Textes zurück.

Methoden und Konstruktoren sollen paket-privat sein, es sei denn, Sie müssen öffentlich sein.

```
class
```

Mail

```
implements
```

Sendable {

```
private
```

String sender;

```
private
```

String receiver;

```
private
```

String subject;

```
private
```

String content;

Mail(

```
String
```

sender, String receiver, String subject, String

`content`

) {

`this`

.sender = sender;

`this`

.receiver = receiver;

`this`

.subject = subject;

`this`

.content = content;

    }

    @Override

`public`

```
 String getSenderAddress() {
        return sender;
    }

    @Override
    public String
```
getReceiverAddress

```


() {
        return receiver;
    }

    @
```
Override

```


    public String getSubject() {

```
return

```


 subject;
    }

    @Override
    public
```
String

```


 getContent() {
        return content;
    }
```

```
    @Override
    public int
getSize
```

```
() {
        return
subject
```

```
.length() + content.
length
```

```
();
    }

}
```

Frage 24 - Interfaces (1 Punkt) [ID: 1808987]

Gegeben sei das Interface `Stack`:

```java
public interface Stack {
    void push(int a);
    int pop();
}
```

Die folgende Klasse `List` (nicht vollständig angegeben) implementiert dieses Interface:

```java
public class List implements Stack {
    private class Node {
        // ...
    }

    private Node head;

    public List() {
        head = null;
    }

    public void push(int value) {
        // ...
    }

    public int pop() {
        // ...
    }

    public void append(int value) {
        // ...
    }
}
```

Welche Statements können bei HIER eingefügt werden, ohne dass es zu einem Compile- oder Laufzeitfehler kommt?

```
public static void main(String[] args) {
    List list = new List();
    Stack stack = new List();

    // HIER
}
```

- [x] list.push(75); *(Ausgewählt = 0.1 Punkte, Nicht ausgewählt = 0 Punkte)*
- [x] stack.push(111); *(Ausgewählt = 0.1 Punkte, Nicht ausgewählt = 0 Punkte)*
- [ ] List.push(114); *(Ausgewählt = 0 Punkte, Nicht ausgewählt = 0.1 Punkte)*
- [x] list.append(114); *(Ausgewählt = 0.1 Punkte, Nicht ausgewählt = 0 Punkte)*
- [ ] stack.append(97); *(Ausgewählt = 0 Punkte, Nicht ausgewählt = 0.1 Punkte)*
- [x] Stack stack2 = new List(); *(Ausgewählt = 0.1 Punkte, Nicht ausgewählt = 0 Punkte)*
- [ ] List list2 = new Stack(); *(Ausgewählt = 0 Punkte, Nicht ausgewählt = 0.1 Punkte)*
- [x] Stack stack3 = list; *(Ausgewählt = 0.1 Punkte, Nicht ausgewählt = 0 Punkte)*
- [x] List list3 = list; *(Ausgewählt = 0.1 Punkte, Nicht ausgewählt = 0 Punkte)*
- [x] Stack stack4 = stack; *(Ausgewählt = 0.05 Punkte, Nicht ausgewählt = 0 Punkte)*
- [ ] List list4 = stack; *(Ausgewählt = 0 Punkte, Nicht ausgewählt = 0.05 Punkte)*

Frage 25 - Meme (1 Punkt) [ID: 1808989]

Erstellen Sie ein Meme, das inhaltlich zur Veranstaltung Programmierung passt. Was Sie darstellen wollen, ist Ihnen überlassen.

Wir werden die Memes, die unseren Hilfskräften am besten gefallen, im Forum vorstellen. (Wenn Sie das nicht wollen, können Sie im Bild z. B. den Zusatz „Bitte nicht veröffentlichen“ unterbringen.) Da beste Meme bekommt außerdem einen Sonderpreis.

Sie dürfen existierende Meme-Formate nutzen, aber kein existierendes Meme 1:1 übernehmen.

Datei hochladen

Välj fil Ingen fil har valts

Maximal erlaubte Upload-Größe: 9.5 MB

(1 Punkt)