Programmieren Präsenzübung 22/23 WiSe Gedächtnisprotokoll

Prüfer: *Prof. Dr.-Ing. Anne Koziolek* am 13.01.2023

Die Prüfung ist unterteilt in:

A	B	C	D	E	Summe
2	2	2	2	2	10

Bearbeitungshinweise: Die Lösung muss in die dafür vorgesehenen Lücken eingetragen werden, dabei stehen die Lücken nicht unbedingt dafür, wie viel eingetragen werden muss. Die abgebene Lösung muss kompilieren können, aber der Checkstyle, der in den Übungsblättern Pflicht ist, muss nicht eingehalten werden. Verwendet werden dürfen Elemente aus **java.lang** der Java SE 17.

Aufgabe A

Implementieren sie das enum Season mit SPRING, SUMMER, FALL, WINTER ohne weitere Methoden in Season zu implementieren. Ergänzen sie die untenstehende Main-Methode, sodass eine Text-Repräsentation von dem Wert FALL von Season ausgegeben wird.

```
public static void main(String[] args) {
    System.out.            ;
}
```

Aufgabe B

Schreiben sie eine Methode, sodass für einen gegebenen instanzierten String word und einen übergebenen char character der Index des letzten Auftreten des character zurückgegeben wird. Wenn character in word nicht vorkommt, geben sie -1 zurück. Mit `String::charAt(int index)` können sie auf den char an einem Index im String zu greifen.

```
public static int lastIndex(char character, String word) {
    int lastIndex =    ;
    for (                ) {
        if (              ) {




        }
    }
    return lastIndex;
}
```

Aufgabe C

Implementieren sie die Methode `sequence456(int[] values)`, sodass für einen bereits instanziert und übergebenen Array aus Ganzzahlen geprüft wird, ob dieser eine Sequenz 4,5,6 enthält und wahr oder falsch zurückgibt. Ein Beispiel: `new int[]{9,4,5,6,3,7}`

```
static boolean sequence456(int[] values) {
        for (int i = 0;         ; i++) {
        if (


                                                                        ) {
            return true;
        }
    }
    return false;
}
```

Aufgabe D

Schreiben sie für die Klasse Container für alle fields (Felder), die veränderbar sind die zugehörigen Setter-Methoden.

```
public class Container {
    private final byte valueOne;
    private short valueTwo;

    Container(byte valueOne, short valueTwo) {
        this.valueOne = valueOne;
        this.valueTwo = valueTwo;
    }

    public byte getValueOne() {
        return valueOne;
    }

    public short getValueTwo() {
        return valueTwo;
    }

    /** hier die Implementierung der Setter-Methoden */









}
```

Aufgabe E

Schreiben sie die Ausgaben, die das folgende Programm ausgibt, nacheinander in den untenstehenden Bereich.

```
public class InheritanceTask {
    public static void main(String[] args) {
        Parent foo = new Child();
        foo.bar();
    }

    static class Parent {
        static int x = 99;
        void foo() { System.out.println(x); qux(); }
        public void bar() { x = 24;}
        static void qux() { System.out.println(x); }
    }

    static class Child extends Parent {
        static int x = 21;
        void bar() { foo();}
        static void qux() { System.out.println(x); }
    }
}
```

```
99
99
```

Lösungen

Ein Hinweis: Es gibt teilweise mehrere gültige Lösungen, ich habe hier eine Variante eingefügt und manchmal andere erwähnt.

Aufgabe A

```
public enum Season {
    SPRING,
    SUMMER,
    WINTER,
    FALL;

}
```

```
public static void main(String[] args) {
    //Season.FALL.toString() geht auch
    System.out.println(Season.FALL);
}
```

Aufgabe B

Man kann hier sowohl den gesamten String durchlaufen und immer wenn ein Charakter gleich character ist, lastIndex anpassen, oder den String von hinten durchlaufen und wenn character vorkommt gleich den Index zurückgeben. Hier ist die letztere Variante.

```
public static int lastIndex(char character, String word) {
    int lastIndex = -1;
    for (int i = word.length() - 1; i >= 0; i--) {
        if (word.charAt(i) == character) {
            return i;
        }
    }
    return lastIndex;
}
```

Aufgabe C

```
static boolean sequence456(int[] values) {
    // da wir auf i+2 in values zugreifen, muss i immer
    // echt kleiner als die Länge von values - 2 sein
    for (int i = 0; i < values.length - 2; i++) {
        if (values[i] == 4 &&
            values[i + 1] == 5 && values[i + 2] == 6) {
            return true;
        }
    }
    return false;
}
```

Aufgabe D

`valueOne` ist final und Setter sollten nur für die veränderbaren Felder geschrieben werden.

```
public class Container {
    private final byte valueOne;
    private short valueTwo;

    Container(byte valueOne, short valueTwo) {
        this.valueOne = valueOne;
        this.valueTwo = valueTwo;
    }

    public byte getValueOne() {
        return valueOne;
    }

    public short getValueTwo() {
        return valueTwo;
    }

    public void setValueTwo(short value) {
        valueTwo = value;
    }
}
```

Aufgabe E

Es wird erst `Child#bar` ausgeführt, dort wird `Parent#foo`. Da Attribute statisch gebunden werden wird `Parent.x` ausgegeben. Und da `qux()` statisch ist, wird statisch gebunden also `Parent#qux` ausgeführt (wo wieder `Parent.x` ausgegeben wird).

```
 99
99
```