Präsenzübung WS 2021/22 Variante 2

> **Gedächtnisprotokoll (!) der Aufgabenstellung der Präsenzübung aus dem Modul Programmieren im Wintersemester 2021/22.** Diese Aufgaben wurden an diejenigen gestellt, die die Präsenzübung um 19:20 Uhr geschrieben haben und unterscheidet sich damit von der Version von 18:20 Uhr. Auszufüllende Lücken sind durch Blockkommentare im Quelltext gekennzeichnet.

> Bearbeitungshinweise (aus Beispielaufgaben von WS 2021/22)
>
> Für eine korrekte Lösung müssen nicht alle Lücken ausgefüllt werden. Die Größe der Lücken steht weiterhin nicht unbedingt in Relation zu den erreichbaren Punkten oder zur Länge des Textes, der für eine korrekte Lösung eingefüllt werden muss. Der von Ihnen eingefüllte Text muss kompilierbar sein. Verwenden Sie nur Elemente des Pakets `java.lang` der Java SE 11 Edition. Achten Sie im Folgenden auch auf Groß- und Kleinschreibung. Setzen Sie nur die in den folgenden Aufgabenstellungen angegebenen Informationen um. Sie müssen bei der Beantwortung der Präsenzübungsaufgaben keine Vorgaben zu Checkstyle-Regeln einhalten wie sonst bei der Bearbeitung der Übungsblätter.

Insgesamt: 10 Punkte in 6 Aufgaben

Aufgabe A: Schleife (2 Punkte)

Vervollständigen Sie folgenden Quelltext so, dass die Methode `countChars` die Anzahl der Vorkommen des Zeichens `character` in der Zeichenkette `word` zurückgibt.
Hinweis: Die Methode `String::charAt(int index)` gibt das Zeichen an der Stelle `index` einer Zeichenkette zurück.

```
public int countChars(String word, char character) {
    int count = 0;
    /*




    */
    return count;
}
```

Aufgabe B: Operatoren (2 Punkte)

Vervollständigen Sie folgenden Quelltext so, dass die Methode `operators`

- "FizzBuzz" zurückgibt, wenn `i` sowohl ein Vielfaches von 5 als auch von 7 ist,
- "Fizz" zurückgibt, wenn `i` nur ein Vielfaches von 5 ist,
- "Buzz" zurückgibt, wenn `i` nur ein Vielfaches von 7 ist,
- und ansonsten die Zahl `i` als Zeichenkette codiert zurückgibt.

Hinweis: Der Operator `%` führt eine Division durch und gibt den Rest (Remainder) zurück.

```
public String operators(int i) {
    if (/*                                  */) {
        return "FizzBuzz";
    } else if (/*                           */) {
        return "Fizz";
    } else if (/*                           */) {
        return "Buzz";
    }
    return /*                               */;
}
```

Aufgabe C: Arrays (2 Punkte)

Vervollständigen Sie folgenden Quelltext so, dass die Methode `copy` eine Kopie des angegebenen Bereichs das Arrays `original` zurückgibt. Dabei soll `copy` die Werte aus `original` vom Index inklusive `from` bis zum Index inklusive `to` enthalten. Sie können davon ausgehen, dass `original` instanziiert ist (`original != null`), dass die Indizes `from` und `to` im Bereich des Arrays `original` liegen (`from >= 0` und `to < original.length`) und dass `from` kleiner als oder gleich `to` ist (`from <= to`).

```
class Wrapper {
    public double[] copy(double[] original, int from, int to) {
        double[] copy = /*                                    */;
        for (/*                                               */) {
            /*


            */
        }
        return copy;
    }
}
```

Aufgabe D: Enum (1 Punkt)

Schreiben Sie den Quelltext für eine Aufzählung (Enum) mit dem Namen `Punctuation`, die die Satzzeichen COLON, COMMA, PERIOD und SEMICOLON enthält.

```
/*




*/
```

Aufgabe E: While-Schleife (1 Punkt)

Gegeben sind die Methoden `query()` mit dem Rückgabetyp `boolean` und `command()` mit dem Rückgabetyp `void`. Vervollständigen Sie folgenden Quelltext so, dass `query` und `command` solange aufgerufen werden, bis `query` das erste Mal `false` zurückgibt. Dabei soll `query` immer vor `command` aufgerufen werden.

```
while (/*                                          */) {
    /*


    */
}
```

Aufgabe F: Dynamische Bindung (2 Punkte)

Geben Sie für den Aufruf der `main`-Methode in folgendem Quelltext die Ausgabe auf der Konsole *zeilenweise* an.

```
public class Main {
    public static void main(String[] args) {
        Parent p = new Child();
        p.g();
    }
}

class Parent {
    static int x = 41;
    void f() { System.out.println(x); h(); }
    void g() { x = 17; }
    static void h() { System.out.println(x); }
}
class Child extends Parent {
    static int x = 29;
    void g() { f(); }
    static void h() { System.out.println(x); }
}
```

Ausgabe:

```

```