Musterlösung

Lösungsblatt zur PiS-Klausur

Bitte nutzen Sie für Ihre Antworten ausschließlich dieses ausgehändigte Lösungsblatt!
Sie geben am Ende der Klausur nur das Lösungsblatt ab!

Nachname, Vorname	**Matrikelnummer**	**Note**
Unterschrift 25.09.2021		

1. Objektvergleich (12 + 4 = 16)

1.

```
@Override public boolean equals(Object other) { // 2
    if (other == null) return false;  // 2
    if (other == this) return true;   // 2
    if (other.getClass() != getClass()) return false; // 2
    Rectangle that = (Rectangle)other; // 2
    return this.height == that.height && this.width == that.width; // 2
}
```

2.

```
@Override public int hashCode() {  // 2
    return Objects.hash(height, width); // 2
}
```

Oft sehe ich Schreibfehler wie z.B. hier Hashcode() oder hashcode(). Das ist unnötig!

2. Collections (4 + 6 + 3 + 2 = 15)

1.

```
List<Set<Integer>> listofSetOfNumbers = List.of(Set.of(2,5,6), Set.of(1,4,5,8)); // 4
// listofSetOfNumbers ==> [[5, 6, 2], [4, 5, 8, 1]]
```

2.

```
Set<Integer> level(Set<Integer> set) {
    return set.stream().map(n -> n % 2 == 1 ? 2 * n : n / 2).collect(Collectors.toSet()); // 6
}
```

Versuche, mit forEach und filter zu arbeiten, funktionieren nicht

3.

```
listofSetOfNumbers.stream().map(set -> level(set)).collect(Collectors.toList()); // 3
```

4.

```
$10 ==> [[1, 3, 10], [2, 4, 10]] // 2
```

Achtung, ein Set kann Elemente nicht doppelt führen!

3. Functional Interfaces (3 + 3 + 3 + 3 = 12)

Sie sollten hier Ausdrücke hinschreiben. Methodenreferenzen sind keine Lambda-Ausdrücke. Und der Cast des Lambda-Ausdrucks muss sein.

Predicate:
```
(Predicate<List>) lst -> lst.size() >= 3;
```

Function:
```
(Function<List, Integer>) lst -> lst.size();
```

Supplier:
```
(Supplier<List<Integer>>) () -> List.of(1,2,3);
```

Consumer:
```
(Consumer<List>) lst -> System.out.println(lst);
```

Achtung: Einige void-Methoden taugen als Consumer nicht, z.B. wenn der Zustand der Liste verändert wird. Das geht bei immutablen Listen nämlich nicht. Seiteneffekte sind super!

4. Fakultät (10)

```
return LongStream.rangeClosed(1, n).reduce(1, (x, y) -> x * y);
```

5. Median (9 + 12 = 21)

5.1 medianA

Für values.length von 10 ist lower 4 und upper 5 – so soll es sein.
Für values.length von 11 ist lower 5 und upper 5. Einfach, gell? ;-)

```
int lower = (values.length - 1) / 2;
int upper = values.length / 2;
return (values[upper] + values[lower]) / 2;
```

5.2 medianB

```
return Arrays.stream(values).sorted().skip(lower).limit(upper - lower + 1)
       .average().getAsDouble();
```

6. Map (4 + 4 = 8)

1.

BinaryOperator<Integer> geht auch.

```
Map<Character,BiFunction<Integer,Integer,Integer>> calc =
    Map.of('+', (x, y) -> x + y,
           '-', (x, y) -> x - y);
```

2.

```
calc.get('+').apply(2,3)
```

7. Fragen (6 x 3 = 18)

	1.	2.	3.	4.	5.	*Punkte*		1.	2.	3.	4.	5.	*Punkte*
A	■	■	□	■	□		D	■	■	■	□	■	
B	□	□	■	■	□		E	■	□	□	■	□	
C	■	□	■	□	■		F	□	□	□	□	■	

Sollten Sie eine Lösung revidieren wollen, streichen Sie die betreffende Reihe durch und notieren Sie hier neben der Tabelle Ihre korrigierte Lösung im Stil von Z13 (die hypothetische Frage Z, Antwort 1 und 3).

Bei vollständiger Übereinstimmung gibt es 3 Punkte. Wenn entweder durch Hinzufügen oder Entfernen eines Kreuzes die vollständig korrekte Antwort erreichbar ist, gibt es 2 Punkte.