Lösungsblatt zur PiS-Klausur

Sie geben am Ende der Klausur nur das Lösungsblatt ab!

Nachname, Vorname	**Matrikelnummer**	**Note**
Unterschrift 22.09.2022		

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
        Ellipse that = (Ellipse)other; // 2
        return this.height == that.height && this.width == that.width; // 2
    }
```

2.

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

2. Integrieren (28)

```
double integrate(DoubleUnaryOperator f, double from, double to, int n) { // jede Zeile 2 Punkte
    assert from < to && n >= 1;
    double deltaX = (to - from) / n;
    double areaR = IntStream.range(0, n)
                            .mapToDouble(i -> from + i * deltaX)
                            .mapToObj(x -> new Rect(x, 0, f.applyAsDouble(x), deltaX))
                            .mapToDouble(Rect::area)
                            .sum();
    double areaL = IntStream.rangeClosed(1, n)
                            .mapToDouble(i -> from + i * deltaX)
                            .mapToObj(x -> new Rect(x - deltaX, 0, f.applyAsDouble(x), deltaX))
                            .mapToDouble(Rect::area)
                            .sum();
    return (areaL + areaR) / 2;
}
```

3. Kreiszahl Pi schätzen (3 + 3 + 9 = 15)

1. `BiPredicate<Double, Double> isInCircle = (x, y) -> Math.sqrt(x * x + y * y) <= 1;` // 3

2. `Supplier<Boolean> hit = () -> isInCircle.test(Math.random(), Math.random());` // 3

3.

```
double estimatePi(long n) {
    long circleHits = Stream.generate(hit)                          // 2
                            .limit(n)                               // 2
                            .filter(b -> b == true)                 // 2
                            .count();                               // 2
    return 4.0 * circleHits / n;                                    // 1
}
```

4. 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

2.

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

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

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

5. Map (4 + 4 = 8)

1.

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

2.

```
assert calc.get('+').apply(2,3) == 5;
```

6. 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).