Java Intro Workshop

6 Aufgaben zu Misc

Aufgabe 1: Enum Basics 15 min

Implementieren Sie zwei Enums:

1. Zuerst sollen Ausrichtungen wie LINKSBÜNDIG, ZENTRIERT und RECHTSBÜNDIG als Enum modelliert werden.

2. Außerdem soll für ein Ticketsystem die Kritikalität mit einem Enum ausgedrückt werden: CRITICAL, HIGH, MEDIUM, LOW. Diese besitzen ein Attribut «attentionLevel» mit 10 für CRITICAL, 7 für HIGH, 3 für MEDIUM und 1 für LOW.

Aufgabe 2: Pizza Enum 15 min

Implementieren Sie einen Enum, der den Herstellungsprozess einer Pizza abbildet, etwa ORDERED, PREPARE_TOPPINGS, BAKING, READY, DELIVERED. Zudem soll der Enum zwei Methoden `isInPreparation()` und `isReadyForDelivery()` erhalten.

Aufgabe 3: Syntaxänderungen bei `switch` 20 min

Vereinfachen Sie folgenden Sourcecode mit einem herkömmlichen switch-case durch die neue Syntax.

```
private static void dumbEvenOddChecker(int value)
{
      String result;

      switch (value)
      {
      case 1, 3, 5, 7, 9:
            result = "odd";
            break;

      case 0, 2, 4, 6, 8, 10:
            result = "even";
            break;
```

```
        default:
                result = "only implemented for values < 10";
        }

        System.out.println("result: " + result);
}
```

Aufgabe 3a

Nutzen Sie zunächst nur die Arrow-Syntax, um die Methode kürzer und übersichtlicher zu schreiben.

Aufgabe 3b

Verwenden Sie nun noch die Möglichkeit, Rückgaben direkt zu spezifizieren und ändern Sie die Signatur in `String dumbEvenOddChecker(int value)`

Aufgabe 3c

Wandeln Sie das Ganze so ab, dass die Spezialform «yield mit Rückgabewert» verwendet wird.

Aufgabe 4 – Record-Grundlagen 10 min

Gegeben seien zwei einfache Klassen, die reine Datencontainer darstellen und somit lediglich ein öffentliches Attribut bereitstellen. Wandle diese in Records um:

```
class Square
{
        public final double sideLength;

        public Square(final double sideLength)
        {
                this.sideLength = sideLength;
        }
}

class Circle
{
        public final double radius;

        public Circle(final double radius)
        {
                this.radius = radius;
        }
}
```

Welche Vorteile ergeben sich – außer der kürzeren Schreibweise – durch den Einsatz von Records statt eigener Klassen?

Aufgabe 5: Person als Record modellieren und Erwachsene aus Liste extrahieren 10 min

Nehmen wir an, es wir wollen Personen in Form eines Records mit den Attributen name und age modellieren. Zudem werden einige Personen als Liste wie folgt bereitgestellt:

```
var persons = List.of(new Person("Mike", 37), new Person("Tim", 49),
                      new Person("Tom", 5), new Person("Michael", 50),
                      new Person("Jim", 7), new Person("James", 17));
```

Filtern Sie die Erwachsenen heraus. Der `Record` soll dazu eine Hilfsmethode `boolean isAdult()` anbieten.

Aufgabe 6 – instanceof und record 30 - 45 min

Vereinfachen Sie den Sourcecode mithilfe der Syntaxneuerungen bei `instanceof` und danach mithilfe der Besonderheiten eigener Methoden sowie Implementieren eines Interfaces bei Records.

```
record Square(double sideLength) {
}

record Circle(double radius) {
}

public double computeAreaOld(final Object figure)
{
      if (figure instanceof Square)
      {
            final Square square = (Square) figure;
            return square.sideLength * square.sideLength;
      }
      else if (figure instanceof Circle)
      {
            final Circle circle = (Circle) figure;
            return circle.radius * circle.radius * Math.PI;
      }
      throw new IllegalArgumentException("figure is not a recognized
figure");
}
```

Analyse

Zwar lässt sich durch modernes `instanceof` sicher eine Verbesserung bezüglich der Lesbarkeit und Anzahl Zeilen erzielen, jedoch deuten mehrere derartige `instanceof`-Prüfungen auf einen Verstoß gegen das sogenannte Open-Closed-Prinzip, eines der SOLID-

Prinzipien guten Entwurfs, hin. Was wäre ein objektorientiertes Design? Die Antwort ist in diesem Fall einfach: Oftmals lassen sich `instanceof`-Prüfungen vermeiden, wenn man einen Basistyp mit passender Methode einführt.

Vereinfachen Sie das Ganze durch ein Interface `BaseFigure` und nutzen dieses. Welche positiven Auswirkungen hat das auf die Flächenberechnung?

Bonus
Führen Sie mit `Rectangle` einen weiteren Typ von Figuren ein. Das sollte aber keine Modifikationen in der Methode `computeArea()` erfordern.