Workshop: Übungen

OO-Design Principles

Aufgabe 1: Law of Demeter 10 min

Betrachte folgende Klasse Person und analysiere diese mit dem neu gewonnenen Wissen zum Law of Demeter: Welche Regeln werden von dem Sourcecode verletzt und wie müsste man die Klasse Person ändern?

```
class Person
{
        final String name;
        final boolean isAdult;
        int age;
        Address homeAddress = new Address()
        Company company = null;

        Person(final String name, final int age)
        {
                this.name = name;
                this.age = age;

                // Regel 1
                this.isAdult = isOlderThan(18);
        }

        // Regel 1
        boolean isOlderThan(final int desiredAge)
        {
                return getAge() > desiredAge;
        }

        boolean sameAge(final Person other)
        {
                // Regel ?
                return getAge() == other.getAge();
        }

      boolean livesIn(final City city)
      {
                // Regel ?
                return getAddress().getCity().getZipCode() ==
                       city.getZipCode()
      }

      boolean isManager()
      {
                // Regel ?
                return getCompany().getStuffMembers().isManager(name)
      }
}
```

Aufgabe 2: OCP 30 - 45 min

Die Übung zum OCP ist separat im Dokument „**Aufgabe2_OCP - Train Table - Teil 1.pdf**“ beschrieben. Es geht darum verschiedene Varianten von Zügen als Informationstafeln zu modellieren. Analysiere die Klassen und Interfaces und zeige mögliche Verbesserungen auf. Führe die Verbesserungen in der Implantierung aus. Beantworte vorab oder beim Modellieren unter anderem folgende Fragen:

- Wieso erfolgt die Berechnung im Konstruktor? Was hat das für Konsequenzen?
- Wie sollten die Änderungen zukunftsweisend erfolgen?
- Was passiert eigentlich, wenn sich die Liste der Züge ändert?
- Wie reflektiert man die aktuellsten Daten korrekt?
- Erkennst du schon ein Entwurfsmuster, was hier eventuell für die Zukunft passt?

Aufgabe 3: LSP 15 - 30 min

Entwickle Klassen oder eine Klassenhierarchie für Rechtecke und Quadrate. Wo könnten Schwachstellen liegen? Finde eine Realisierung für `Rectangle` und `Square`, sodass das LSP eingehalten wird:

a) Mit Vererbung
b) Ohne Vererbung: Was ändert sich dadurch? Was gilt dann?

Aufgabe 4: Immutability 15 - 30 min

Betrachte die folgende Klasse `PeriodOfTime` und betreibe ein Code-Review sowie eine Design-Analyse, um daraus eine Immutable-Klasse zu machen.

a) Was sind die Anforderungen an eine Immutable-Klasse?
b) Finde mögliche Schwachstellen in der bisherigen Umsetzung.
c) Implementiere eine Immutbale-Version dieser Klasse.

```
public final class PeriodOfTime
{
    private final Date start;
    private Date end;

    public PeriodOfTime(final Date start, final Date end)
    {
        Objects.requireNonNull(start, "start must not be null");
        Objects.requireNonNull(end, "end must not be null");

        if (start.after(end))
            throw new IllegalArgumentException(start +
                                               " must be <= " + end);

        this.start = start;
        this.end = end;
    }
```

```
    public Date getStart()
    {
        return start;
    }

    public Date getEnd()
    {
        return end;
    }

    public void withNewEnd(final Date newEnd)
    {
        if (start.after(newEnd))
            throw new IllegalArgumentException(newEnd +
                                               " must be >= " + start);

        this.end = newEnd;
    }
}
```