Tutorium zur Vorlesung Programmieren

11. Tutorium

Joshua Gleitze

30. Januar 2017

IPD Koziolek

Anmerkungen

„Gather together the things that change for the same reasons. Seperate those things that change for different reasons.“

– Robert Martin

Trennung der Zuständigkeiten

Zuständigkeiten nicht getrennt:

Calendar.java

```
public void printAppointmentsThatConflictWith(String name) {
    Appointment appointment;
    for (Appointment app : appointments) {
        if (app.getName().equals(name)) {
            appointment = app;
            break;
        }
    }
    if (appointment == null) {
        Terminal.printLine("Appointment not found.");
    }
    ...
```

Trennung der Zuständigkeiten

Zuständigkeiten nicht getrennt:

Calendar.java

```
public String getAppointmentsThatConflictWith(String name) {
    Appointment appointment;
    for (Appointment app : appointments) {
        if (app.getName().equals(name)) {
            appointment = app;
            break;
        }
    }
    if (appointment == null) {
        return "Appointment not found."
    }
    ...
```

Trennung der Zuständigkeiten

Zuständigkeiten getrennt:

Calendar.java

```
public SortedAppendList<Appointment>
  ↪  getAppointmentsThatConflictWith(String name) throws
  ↪  AppointmentNotFoundException {
    Appointment appointment;
    for (Appointment app : appointments) {
        if (app.getName().equals(name)) {
            appointment = app;
            break;
        }
    }
    if (appointment == null) {
        throw new AppointmentNotFoundException("There is no
          ↪  Appointment called " + name);
    }
    ...
```

Gewichteter Durchschnitt

Course.java

```java
public static <L extends Course> Optional<Double> weightedAverage(Iterable<L>
  ↪   courses,
    Function<L, Optional<Double>> gradeFunction) {
    double grades = 0;
    int pointSum = 0;
    boolean foundLectureWithGrade = false;
    for (L course : courses) {
        Optional<Double> grade = gradeFunction.apply(course);
        if (grade.isPresent()) {
            int points = course.getPoints();
            pointSum += points;
            grades += grade.get() * points;
        }
    }
    return foundLectureWithGrade ? Optional.of(grades / pointSum) : Optional.empty();
}
```

Module.java

```java
@Override
public Optional<Double> getAverageGrade() {
    return Course.weightedAverage(this.lectures, Lecture::getAverageGrade);
}
```

compareTo

Finde den Fehler:

Appointment.java

```java
public int compareTo(Appointment other) {
    if (this.getFrom().compareTo(other.getFrom()) == -1) {
        return -1;
    } else if (this.getFrom().compareTo(other.getFrom()) == 1) {
        return 1;
    } else {
        ...
    }
}
```

compareTo

Richtig:

Appointment.java

```
public int compareTo(Appointment other) {
    int compStart = this.getFrom().compareTo(other.getFrom();
    if (compStart != 0) {
        return compStart;
    else {
        ...
    }
}
```

compareTo

Richtig:

Appointment.java

```
public int compareTo(Appointment other) {
    int compStart = this.getFrom().compareTo(other.getFrom();
    if (compStart != 0) {
        return compStart;
    else {
        ...
    }
}
```

Programmiere gegen Schnittstellen und nicht gegen eine Implementierung!