Department Mathematik/Informatik, Abteilung Informatik
Software & System Engineering Group
Software Technology Group
Lecture Object-Oriented Software Engineering, SS 2024

Alexander Korn, Adrian Bajraktari,
Prof. Dr. Michael Felderer

UNIVERSITÄT ZU KÖLN

Extra Exercises 1. Object-Oriented Programming

General Remark: These extra exercises are intended for you to train your understanding of the course material. There will be no grading of these exercises (points on the sheet $\neq 0$ are just to show you how we could grade the exercise with regard to the exam). Feel free to ask questions about these exercises, but solutions will be provided (Attention! Remember what we said about solutions in the first lecture!).

Exercise 1. Classes and Objects

Implement the following as Java code:

- Each course has a name, a number of credit points, and a number of hours per week (SWS).
- Implement a toString method for the course that returns its details as a string.
- Each student has a name, a matriculation number and a list of courses they attend.
- Implement a print method for the student that prints its details.
- Create a method for student that adds a course to their list of courses.
- Create 3 students that each attend 2 courses and print their details.

[0 points]

Exercise 2. Object-Oriented Programming

Analogous to the `Student` class we had in the lecture, define a class `Professor` that has the following information stored: firstname, lastname, age, chair, list of lectures, list of publications, list of students in direct supervision (bachelor/master thesis, student jobs, phds). We want professors to greet us with their full name and the name of their chair. A lecture has a name, a number of credit points and a number of weekly hours (SWS).

You can get the whole student class from https://github.com/AdrianBajraktari/oose24.git in `src/extra1/Student.java`.

[4.5 points]

Exercise 3. Inheritance and Polymorphism

a) Starting with your solution from ex. 2, extract members of the classes `Student` and `Professor` and change your implementation so that they are not defined in both classes, but shared through a common supertype `Person`. Make sure that you also share initialization of instance variables (-> use a super constructor).

b) Now, create a greet method in the `Person` class and let it print the full name and age of a person.

c) Insert the following code (also available in `src/extra1/ProfCreate.java`):

```
1 Person myProf = new Professor("Andreas", "Vogelsang",
2             "Software & Systems Engineering", 39,
3             new ArrayList<Lecture>(),
4             new ArrayList<Publication>(),
5             new ArrayList<Student>());
```

What will `myProf.greet()` print? Why is that? (If you used arrays or other collections instead of lists, change the above snippet accordingly or directly change your implementation, as you are supposed to do so in the next exercise anyway).

[4.5 points]

Exercise 4. Object Identity

Given the following code, explain the output and modify the code to get a valid result.

```
1  public class Student {
2          private String name;
3          private int matNr;
4
5          public Student(...) {...}
6  }
7
8  Student alice = new Student("Alice", 1);
9  Student bob = new Student("Bob", 2);
10
11 System.out.println(alice == bob);
```

[0 points]

Exercise 5. Object Identity

Inspect the following code snippet. The full program is in `src/extra1/OID.java` in. What will the four print statements output to the console? Is this what we expect? Explain why this happens and how to fix it.

```
1 private static Object[] copy(Object[] array) {
2     return Arrays.copyOf(array, array.length);
3 }
```

```
4
5  public static void main(String[] args) {
6      Student[] students = new Student[] {new Student("Tom"),
7                                          new Student("Alice")};
8      Student[] students2 = (Student[]) copy(students);
9
10     System.out.println(Arrays.toString(students));
11     System.out.println(Arrays.toString(students2));
12     students[0].name = "Tim";
13     System.out.println(Arrays.toString(students));
14     System.out.println(Arrays.toString(students2));
15 }
```

[1 point]

Exercise 6. Object-Based Programming in JavaScript

Expand the delegation example of the lecture by an object `bigModule`, which has the following pairs of properties and values: `lectureHours: 4, exerciseHours: 2, cp: 9`.

Set the `__proto__` reference of `swt` to `bigModule`. Move the `getFullDescription()` method to `bigModule` and change it so that it also prints the above information, e.g.: "Module name: Software Engineering, CP: 9, lecture hours: 4, exercise hours: 2".

Then, change `swt` to have a 3 lecture hours + 3 exercise hours format. For `inf2`, however, we change the format to 2 + 2 and 6 CP.

[0 points]

$\sum$ **10.0 points**