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
Solution

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]

```
1  public class Student {
2          private int matNr;
3          private String name;
4          private List<Course> courses = new ArrayList<>();
5
6          public Student(String name, int matNr) {
7                  this.name = name;
8                  this.matNr = matNr;
9          }
10
11         public void addCourse(Course course) {
12                 this.courses.add(course);
13         }
14
15         public void print() {
```

```
16                  System.out.println(this.name + ", " +
17                  this.matNr + " attends the following courses: "
18                  + this.courses);
19          }
20  }
21
22  public class Course {
23          private int creditPoints;
24          private int sws;
25          private String name;
26
27          public Course(String name, int creditPoints, int sws) {
28                  this.name = name;
29                  this.creditPoints = creditPoints;
30                  this.sws = sws;
31          }
32
33          public String toString() {
34                  return this.name + " (" +
35                          this.creditPoints + " CP, " + this.sws + " SWS)";
36          }
37  }
```

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 find the whole student class in `de.uni.koeln.sse.se.homework_provided_code.oop`.

[4.5 points]

```
1   public class Professor {
2       private String firstName;
3       private String lastName;
4       private String chair;
5       private int age;
6       private List<Lecture> lectures;
7       private List<Publication> publications;
8       private List<Student> students;
9
10      public Professor(String firstName, String lastName, String chair,
11                       int age, List<Lecture> lectures,
12                       List<Student> students) {
13          this.firstName = firstName;
14          this.lastName = lastName;
15          this.chair = chair;
```

```
16          this.age = age;
17          this.lectures = lectures;
18          this.students = students;
19      }
20
21      public void greet() {
22          System.out.println("Hello, my name is Prof. " +
23                  this.firstName + " " + this.lastName +
24                  " and I lead the chair for " + this.chair);
25      }
26  }
27
28  public class Lecture {
29      private String name;
30      private int creditPoints;
31      private int sws;
32
33      public Lecture(String name, int creditPoints, int sws) {
34          this.name = name;
35          this.creditPoints = creditPoints;
36          this.sws = sws;
37      }
38  }
```

Exercise 3. Inheritance and Polymorphism

a) Starting with your solution from homework 4, 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 `de.uni.koeln.sse.se.homework_provided_code.oop`):

```
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]

```
1  public class Person {
2      private String firstName;
```

```
3       private String lastName;
4       private int age;
5
6       public Person(String firstName, String lastName, int age) {
7           this.firstName = firstName;
8           this.lastName = lastName;
9           this.age = age;
10      }
11
12      public void greet() {
13          System.out.println("Hello, my name is " +
14              this.firstName + " " + this.lastName +
15              " and I am " + this.age + " years old.");
16      }
17  }
18
19  public class Student extends Person { ... }
20
21  public class Professor extends Person {
22      private String chair;
23      private List<Lecture> lectures;
24      private List<Student> students;
25
26      public Professor(String firstName, String lastName, String chair,
27                  int age, List<Lecture> lectures,
28                  List<Student> students) {
29          super(firstName, lastName, age);
30          this.chair = chair;
31          this.lectures = lectures;
32          this.students = students;
33      }
34
35      ...
36  }
```

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]

```
1  public class Student {
2          private int matNr;
3          private String name;
4          private List<Course> courses = new ArrayList<>();
5
6          public Student(String name, int matNr) {
7                  this.name = name;
8                  this.matNr = matNr;
9          }
10
11         public void addCourse(Course course) {
12                 this.courses.add(course);
13         }
14
15         public void print() {
16                 System.out.println(this.name + ", " +
17                 this.matNr + " attends the following courses: "
18                 + this.courses);
19         }
20 }
21
22 public class Course {
23         private int creditPoints;
24         private int sws;
25         private String name;
26
27         public Course(String name, int creditPoints, int sws) {
28                 this.name = name;
29                 this.creditPoints = creditPoints;
30                 this.sws = sws;
31         }
32
33         public String toString() {
34                 return this.name + " (" +
35                         this.creditPoints + " CP, " + this.sws + " SWS)";
36         }
37 }
```

Exercise 5. Object Identity

Inspect the following code snippet. The full program is in `de.uni.koeln.sse.se.homework_provided_code.object_identity` in the repository you needed to clone in the last homework (https://github.com/AdrianBajraktari/swt2023). You may need to pull the newest changes. 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]

The output will be
[Studentname='Tom', Studentname='Alice']
[Studentname='Tom', Studentname='Alice']
[Studentname='Tim', Studentname='Alice']
[Studentname='Tim', Studentname='Alice'].
The reason is that Arrays.copyOf(...) will produce a shallow clone of the provided array. Fixed via Arrays.deepCopyOf(...).

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]

```
1  let inf2 = {
2      __proto__: swt,
3      modulename: "Informatik II",
4      lectureHours: 2,
5      exerciseHours: 2,
6      cp: 6
7  };
8
9  let swt = {
10     __proto__: bigModule,
11     modulename: "Software Engineering",
```

```
12      lectureHours: 3,
13      exerciseHours: 3
14  };
15
16  let bigModule = {
17      cp: 9,
18      lectureHours: 4,
19      exerciseHours: 2,
20
21      getFullDescription: function() {
22          return `Module name: ${this.modulename}, CP: ${this.cp}, lecture hours:
23                                              ${this.lectureHours}, exercise hours: ${
24      }
25  }
```

$\sum$ **10.0 points**