Software Engineering WS 2022/23

Exercise 2: Java

Hand in via Moodle: Oct. 25, 14:00

Task 1: IDE

Install the IDE *IntelliJ IDEA* from JetBrains in the Ultimate Edition, which you can get for free as a student: (https://www.jetbrains.com/de-de/community/education/).

We will probably use some features of the Ultimate Edition, so the community version is not sufficient. You can in principle use other IDEs (or a text editor, the console and a JDK), but our support and tasks are tuned to features of IntelliJ and you might have to do some tasks laboriously by hand, which IntelliJ IDEA will do for you with a few mouse clicks.

You can install Git and a JDK from within IntelliJ IDEA, probably the IDE will automatically ask you for it when you get to a place where it is needed.

1.1: Hello World!

This task should not be handed in. Everyone should work on the task alone on their computer. If necessary, help each other in your team. If you are already confident in using IntelliJ IDEA, you can skip this task.

After starting IntelliJ for the first time, start a new project by clicking on the *new project* button. Give the project a meaningful name (e.g. *JavaIntrodcutionSE*), specify the location and make sure *Java* is the selected language. Select the highest JDK version available and click on *Create*.

IntelliJ IDEA will automatically create a project with a class called `Main` containing a `Main` method.

The `main` method will already contain a text output (the famous `Hello World`) analogous to the example from the lecture. Run your project by clicking on the green start button on the top right.

Congratulations, you have executed your first Java program!

More detailed instructions can be found at:
https://www.jetbrains.com/help/idea/creating-and-running-your-first-java-application.html

Task 2: Students

Before you start with your implementation read the whole task.

2.1: `Main`

Create a new Java project with a `Main` class and a method `main()`. You can use the procedure described in the previous task.

2.2: `Student`

Create a `Student` class with the following fields:

- `String firstname;`
- `String lastname;`
- `long studentId;`
- `double weight;`
- `Date birthday;`

Don't forget to import `java.util.date`!

2.3: `Student.SortKey`

Add the following `enum` inside the `Student` class:

```
public enum SortKey {FIRSTNAME, LASTNAME, STUDENT_ID, WEIGHT, BIRTHDAY}
```

2.4: `Student()`

Provide constructors with the following signatures:

- `Student()`
- `Student(String, String, long, double, Date)`

The first constructor should call the second one with parameters `(null, null, studentId, 0, new Date(0))`. Inside the second constructor the objects fields are initialized with the passed parameters.

2.5: `Student` getter & setter

Provide getter and setter for each field. Almost all of them should be accessible from any other class. Only the `studentId` should not be able to be changed from the outside. It can be read from any other class of course.

In addition provide a method `getName()`, which provides the full name, separated by a space, as a `String`.

2.6: `StudentList`

Create a class `StudentList`, which contains an `ArrayList` containing students. `StudentList` should not inherit from `ArrayList`!

2.7: `StudentList()`

Provide constructors with the following signatures:

- `StudentList()`
- `StudentList(StudentList)`

The second constructor should be a copy constructor (see e.g., https://www.javatpoint.com/java-copy-constructor-example).

2.8: `StudentList` methods

Implement these methods:

- `boolean add(Student student)`: adds `student` to the list, if the list does not yet contain a student with the same studentId. Returns `true` on success and `false` otherwise.
- `boolean remove(Student student)`: removes the student with the studentId of `student` from the list. Returns `true` if an entry was deleted and `false` otherwise.
- `Student remove(int pos)`: removes the student at position `pos` from the list. Returns the corresponding student if an entry was deleted and `null` in case `pos` is too small / big.
- `Student get(int pos)`: Returns the student at position `pos` or `null` in case `pos` is too small / big.
- `boolean ArrayList<Integer> findLastname(String lastname)`: Returns a (possibly empty) list of positions, corresponding to the students in the list whose lastname matches `lastname`.
- `ArrayList<Integer> findFirstname(String firstname)`: Returns a (possibly empty) list of positions, corresponding to the students in the list whose firstname matches `firstname`.
- `int findStudentsByAge(int age)`: Should the list contain students with the age `age`, return all positions of the students in the list, else return an empty list.
- `int size()`: Return the number of students inside the list.
- `private boolean containsId(final long studentId)`: Returns `true` if the list contains the student with studentId `studentId`. Otherwise return `false`.

Ensure that each studentId can only appear once in the list!

Provide useful JavaDoc for each method!

2.9: `StudentList.toString()`

Override the `toString()` method in `Student` and return all five fields as a single `String`, separated by spaces.

2.10: Tests

Test your implementation either with tests, the debugger or ”by hand.” Be prepared to explain how you tested your implementation and why you believe you covered all test cases.

Create a `StudentList` containing at least ten students inside your `main`. Use the copy constructor to create a second `StudentList`. Then change some students inside the second list. Test whether you really wrote a copy constructor. Explain your test in a short comment inside your source code.

2.11: `StudentList.sort()`

Write a method `sort(Student.SortKey key)` in `StudentList` that sorts the list by the corresponding element. Implement the sorting on your own!

Test your implementation. Provide useful comments on how you did the sorting.

2.12: Restrictions

The only allowed imports for this task are:

- `import java.util.Date;`
- `import java.util.ArrayList;`
- `import java.util.List;`

You are not allowed to use any methods provided by `List` / `ArrayList`, except:

- `add(E e)`
- `get(int index)`
- `set(int index, E e)`
- `remove(int index)`
- `size()`
- `iterator()`

Deliverables

Upload a zip file containing the following files in Moodle by the deadline:

Task 2:

- Main.java
- Student.java
- StudentList.java

Notes:

- Each class contains a comment header with your student IDs and group number (AGxxxx).
- Properly format and comment your code.