Step 10 – Set, Optional, equals and hashCode

We have been through the biggest topics in the course, so now we can learn various smaller topics that you should definitely master when learning Java.

Goals for this step:

- I master the use of a Set, and I know when it may be appropriate.
- I understand how I can use Optional.
- I know when it is wise to implement equals and hashCode for objects.
- I understand the need to validate input in methods.

Relevant chapters of the book:

- Chapter 17: Comparing objects
- Chapter 22: Set

Here is the questionnaire for step 10 where you can tell how it went.

We will get to know a new type of collection, namely Set. We will receive training in adding and retrieving objects in a Set. Furthermore, we will look at the use of Optional, how we can validate input to a method and why equals / hashCode may be wise to implement.

Task 1 – A new start

Create a new project. Create a class using a main method. Create a Class Program. In the main method, create an object of the Program class.

Create a Class Person. The class should have two private fields: age (int) and name (String). Make getters and setters and a toString method. Create a constructor that accepts both age and name (and sets the values in the fields).

If you are struggling with this task, then you can choose to use the code starting point located in Canvas. In that case, run the program to see if the import of the code went well. Expected Result: Person {age = 20, name = 'Anton Antonsen'}

Task 2

Change the Program class so that it has a HashSet that can hold Person objects.

Create a method in Program. The method must:

- Create two objects of class Person. Both persons must have the same values for name and age (for example Atle Antonsen, 20 years).
- Put both objects in the hash set.
- Print all the values in the hash set (after the objects were entered).

Call the method from main. What is the result? Shouldn't this only hold unique values? So now what😊

Task 3

Create equals and hashCode methods in the Person class. You can autogenerate these in IntelliJ (shown in the intro video). Run the program again. What happened? Hopefully you have seen that the content of the hash set changed compared to the previous run. Think about why the change happened.

Task 4

Create a method getSamplePerson (String name) in the Program class. The method should retrieve a person in the hash set (if there is a person there who matches by name). If there are several people in the hash set with the name, then it does not matter which of these is returned. If the person is not in the hash set, then the method should **not** return null (check the intro video if you are unsure what is meant by it).

Test that the method works as intended by trying to retrieve a person who exists and who is not in the hash set.

Task 5

Create an addPerson method that puts a person in the hash set. You must validate input to the method. A person should not be able to be null and the age of a person can not be negative. The person's name cannot be null.

Check if the validation works by trying to use addPerson to add valid and non-valid people.

Task 6

Using the addPerson method, add ten people to the hash set.

Then create a method that accepts an age (int) and returns a Set of people (from the hash set) that are older than the age parameter in the method inputs.

Task 7

Get to know HashSet better by trying out different methods that the class offers. Use the Java documentation for Java 15 to find out what methods the class offers.

Extra Task

In this course, we did not go into functional interfaces. But you can have a look here in any case. This is not exam-relevant, but useful to know when programming in Java.

Have some of the functionality from previous tasks done as lambdas and using functional interfaces.