Assignment Student

Object Orientation

Spring 2025

1 Learning Objectives

After having completed this assignment, you should be able to

- Write, compile and execute a Java program (in Visual Studio Code);
- Define classes with fields and methods;
- Distinguish static and dynamic methods;
- Define objects as instances of such a class;
- Use Java arrays;
- Use rudimentary Java input/output;
- Define and use a `toString()` method for user-defined classes;
- Concentrate all input/output in a single class.

2 Programming Environment

Visual Studio Code can be downloaded from `code.visualstudio.com`. The standard edition Java SE or OpenJDK is sufficient for this course. Java versions below 8 or above 21 are currently not supported. We recommend installing Java 21.

After installing both Visual Studio Code and Java, open Visual Studio Code and install the 'Extension Pack for Java' extension. This can be done by going to: File → Preferences → Extensions → Search for the required extension and install.

To start the assignment, go to Brightspace and download the `assignment-student-start.zip` file. Unzip the file somewhere where you can easily find it, e.g., `programming/java/`. Next, start Visual Studio Code, go to: File → Open Folder → Select `programming/java/assignment-student-start`.

Running from the command-line It should now be possible to run the code inside Visual Studio Code. If this does not work, or you decided to use something different from Visual Studio Code, it is also possible to run the code from the command-line. The command `./gradlew run --console=plain -q` runs the code in `Main.java`, and `./gradlew check` runs the tests. On Windows, replace `./gradlew` with `gradlew.bat` in the commands above.

3 Assignment

In this assignment you have to design and implement a Java program that allows the user to create a group of students and offers the possibility to change the name of these students.

For this assignment we assume that every student has a given name and a family name, both consisting of exactly one word. Each student also has a student number, stored in an integer. These values are specified upon the construction of the student object. The student number is fixed, but the name can be changed by an appropriate setter method. The student with name Lucy Liddels and number 42 should be printed as `Lucy Liddels, s42`.

The behavior of the program to create and manipulate a group of students is summarized as follows.

1. Your program should ask the user how big the group of students is supposed to be and create a new group of the given size.
2. For each group member, your program should request a name and a student number and add this student to the group. You can choose to let the user input everything in one line and make your program parse the input, or to let the user input everything in a separate line.

   ```
   Please enter the group size:
   2
   Please enter a student:
   42 Luucy Liddels
   ```

3. After the group has been created, the program should print all the students from the group.

   ```
   The group now contains:
   Luucy Liddels, s42
   Bob Biggens, s101
   ```

4. In a loop, the program should then ask the user to input the student number, only the numerical part without the leading `s`, alongside with a new name. Every time a name is changed, the entire group should be printed again

   ```
   Student number and new given/family name?
   42 Lucy Liddels
   The group now contains:
   Lucy Liddels, s42
   Bob Biggens, s101
   ```

5. The program terminates as soon as the user enters a negative student number.

   ```
   Student number and new given/family name?
   -1
   Bye!
   ```

3.1 Input and Output

In this exercise all I/O should happen in the Main class. The other classes should only be used to organize and manage students and groups. No I/O should happen in the other classes.

3.2 The Main Class

Java allows to put a `main` function in every class, which means one project can have many `main` functions. When running a program, you have to tell Java which class to use as the main class. This can become confusing when other people read your code. You should have one class in your project whose sole responsibility it is to have the `main` function. This class should have a clear name, like `Main`, or `Assignment01Main`. This class should only have the `main` function, and maybe a couple of helper functions if needed. No other class should have a `main` function.

4 Test Cases

You can download a project template with some test cases for this assignment from Brightspace. These tests exist for your convenience, to provide some test data and expected outputs of the program. Please use them for self study to check your own work. We will not check or grade the test cases.

For each assignment we will provide test cases whenever possible. Alongside the test cases there will sometimes be a class called AssignmentXXTester, where XX is the assignment number. The Tester will be provided as a class with empty methods. You have to implement these methods. The Tester should not contain any program logic, but should only call the corresponding functions from your project. The Tester exists because we do not want to give away too much about the assignment solution in the test cases.

We hope that you find the little additional effort of filling in the Tester worth the added confidence that your implementation is correct.

For example, in this assignment you will be given the class `Assignment01Tester` that looks as follows.

```
public class Assignment01Tester {
  public void createGroup(int i) {
  }
  public void addStudent(int sNumber, String firstName, String
      lastName) {
  }
  public String printStudents() {
    return "";
  }
}
```

You should fill in this class as follows.

```
// Import your package, so that you can use your classes
import assignment.*;
public class Assignment01Tester {
  private Group group;
  public void createGroup(int i) {
    group = new Group(i);
  }
  public void addStudent(int sNumber, String firstName, String
      lastName) {
```

```
    group.addToGroup(new Student(sNumber, firstName, lastName))
        ;
  }
  public String printStudents() {
    return group.toString();
  }
}
```

5 Hand In

These instructions are important. Read them carefully.

5.1 Enrol In a Group

Please work in teams of two, but teams of one are also okay. Before you can hand in your project, you and your teammate have to enrol in a group in Brightspace. **Find an empty group and make sure you and your teammate are enrolled in the same group.** If you find other people accidentally enrolled in your group, leave the group and pick another one.

Groups will be made fixed on the day of the assignment deadline. You can only switch teams after that by sending an email to Maris Galesloot. If you decide to switch teams, tell Maris as soon as possible, because grades given to a submitted assignment automatically apply to all members of the group.

5.2 Submit Your Project

To submit your project, follow these steps.

1. Find the folder that contains your assignment. In Visual Studio Code, you can find this by going to: File → Open Folder. Add the correct folder to a zip file. At the root level, your zip file should contain only one folder, e.g. `assignment-student-start`, which contain the entire project, i.e. the `app` folder and the Gradle files. *Do not submit only the `.java` files or the `src` folder!*
2. **Submit this zip file on Brightspace**. Do not submit individual Java files. Only one person in your group has to submit it. Submit your project before the deadline, which can be found on Brightspace.
3. **Do not submit any other format.** Do not submit .rar or .tar.gz or .7z files. Only zip files are allowed.