Lab 1: Java Basics, Merge Sort, Maven and IDE

Objectives

* Creating a Java project; creating and editing a Java program

* Adding a library to the build path

* Running project inside and outside the IDE

* Full mark: 40 points

Source files

* `MergeSort.java`

* `pom.xml`

1 Introduction

Welcome to the first lab. These labs are generally intended to let you explore concepts introduced in the course lectures. During each lab, you need to follow the instructions from problem setup to deliverables. This lab contains 2 deliverables, which you should submit in the eclass.

1.1 IDE

There are multiple integrated development environment (IDE) that can be used to write programs (Eclipse, IntelliJ, VSCode, etc.). Most of the IDEs contain functionality to allow developers create, share and edit generic projects and files in the platform, while participating within a multiple team development environment repository. In general, IDEs allow you to develop applications in different programming languages, such as: C, C++, Ruby, R, Java, JavaScript, PHP, Python, etc.

In this lab we will focus on writing, editing and running Java programs.

To develop your labs, you can use your IDE of preference. Please, use the one you know the most and you are comfortable using it. The labs will not have a tutorial of how to use the IDEs.

1.2 IDE Resources

- Eclipse Homepage : http://www.eclipse.org/

- IntelliJ IDE : https://www.jetbrains.com/idea/

- Visual Studio Code : https://code.visualstudio.com/

Merge Sort

Sort a list of integers using the merge sort algorithm:

* https://youtu.be/lLTHALzhbs0 (Super short, no words, pictures only)

* https://youtu.be/Kg4bqzAqRBM?t=1488 (The traditional version from M.I.T.)

* Or find your own

Deliverable 1 -- Create and Run a Simple Java Program (Merge Sort)

1. Using your preferred editor implement a program to sort an array of integers using the merge sort algorithm. Use as base of your program the file `MergeSort.java` that is part of this lab

2. There are different ways to execute the byte-compiled file:

    * Using your IDE to run it (This depends of the IDE that you are using)

    * Using the command line

    * Using a building tools (e.g., Maven)

3. Using your IDE to run or

    Using the command line, open a terminal from your operating system (Terminal/cmd/PowerShell) and compile your program:

    ```

    > javac MergeSort.java

    ```

    After compiling the file, the file MergeSort.class is created. To run it, you need to execute:

    ```

    > java MergeSort

    ```

The output should be a list of numbers to sort and in the second line the same numbers but sorted after using the implemented merge sort algorithm. For example:

```

107 86 156 0 126 191 43 63 36 140

0 36 43 63 86 107 126 140 156 191

```

**This deliverable will be evaluated for correctness, code quality and documentation (30 points).

Apache Maven

A defined build process is one of the most necessary tools in the software development process, ensuring that the software you produce is built to the required specifications. You should establish, document, and automate the exact series of steps required to correctly build your product.

A defined build process helps close the gap between the development, integration, test and production environments. A build process alone will speed the migration of software from one environment to another. It also removes many issues related to compilation, library paths, or properties that cost many projects considerable time and money.

Apache Maven is a Java-based build tool with special support for Java programming language. It helps programmers to build complex applications and place the files in the desired location with less effort. Maven executes different tasks using Java classes and XML-based configuration files.

A Maven build file comes in the form of an XML document. All you need is a simple text editor to edit a Maven build file (`pom.xml`).

A brief introduction of using Maven can be found in https://maven.apache.org/guides/getting-started/maven-in-five-minutes.html

Maven uses a well-defined file structure to organize the source code and other resources. In general, inside the project directory exist a `pom.xml` with the project specification and dependencies. The source code are stored into the `src` directory. Usually there are two directories inside `src`; a directory for the main code (`main`) and a directory for the testing code (`test`).

Deliverable 2 -- Maven Build File

In this step, you need to compile and run your code using Maven.

- *Step 1:* Copy and paste your source code (all java files) into `src/main/java` folder in your project directory (`MergeSort` in this lab).

- *Step 2:* Consider the sample Maven build file `pom.xml` that is located inside the project directory (`MergeSort`).

Maven works on stages. A phase is a step in the build lifecycle, which is an ordered sequence of phases. When a phase is given, Maven will execute every phase in the sequence up to and including the one defined. For example, if we execute the compile phase, the phases that actually get executed are:

1. validate
2. generate-sources
3. process-sources
4. generate-resources
5. process-resources
6. compile

- *Step 3.1:* Compile your program you can use your IDE or the command line. Inside of your project directory execute the following command:

```
MergeSort> mvn compile
```

- *Step 3.2:* To generate a JAR file with all the required byte-code and dependencies, you can execute the following command:

```
MergeSort> mvn package
```

- *Step 4.1:* At this point you can run your program using Maven, executing:

```
MergeSort> mvn exec:java -Dexec.mainClass="MergeSort"
```

- *Step 4.2:* During compilation, Maven creates a target directory where the compiled class are stored, as well as the JAR file after execute the package phase. To manually run the programs you can use:

```
MergeSort> java -cp target/MergeSort-1.0-SNAPSHOT.jar MergeSort
```

**This deliverable will be evaluated for correctness (10 points).

Evaluation criteria

- **Deliverable 1** : Create and Run a Simple Java Program (Merge Sort)
    - Correctness : 15 pts
        - Does not work: (0)
        - Recursion is correct but merge fails : (1-6)
        - Work but not in all the cases : (7-11)
        - Works correctly in all the cases : (12-15)
    - Code quality : 10 pts
        - Naming and usage of variables : (3)
        - Design (logic structure)
            - Correct use of indentation : (1)
            - Use of helper functions/procedures : (5)
            - Correct use of statements : (1)
    - Documentation : 5 pts
        - No documentation (0)
        - One line comments but not using proper javadoc : (2)

- Clear documentation about what the function/procedure does using javadoc style (5)

- **Deliverable 2** -- Maven Build File
    - Correctness : 10 pts
        - Code in JAR runs correctly : (5)
        - Compilation and JAR creations works correctly: (5)

Total evaluation: 40 pts