MINISTRY OF EDUCATION AND SCIENTIFIC RESEARCH

TECHNICAL UNIVERSITY
OF CLUJ-NAPOCA

FACULTY OF AUTOMATION AND COMPUTER SCIENCE
COMPUTER SCIENCE DEPARTMENT

FUNDAMENTAL PROGRAMMING TECHNIQUES

Lab Resources

Ioan Salomie
Marcel Antal

Tudor Cioara
Claudia Daniela Pop
Mitrea Dan

Ionut Anghel
Dorin Moldovan

2021

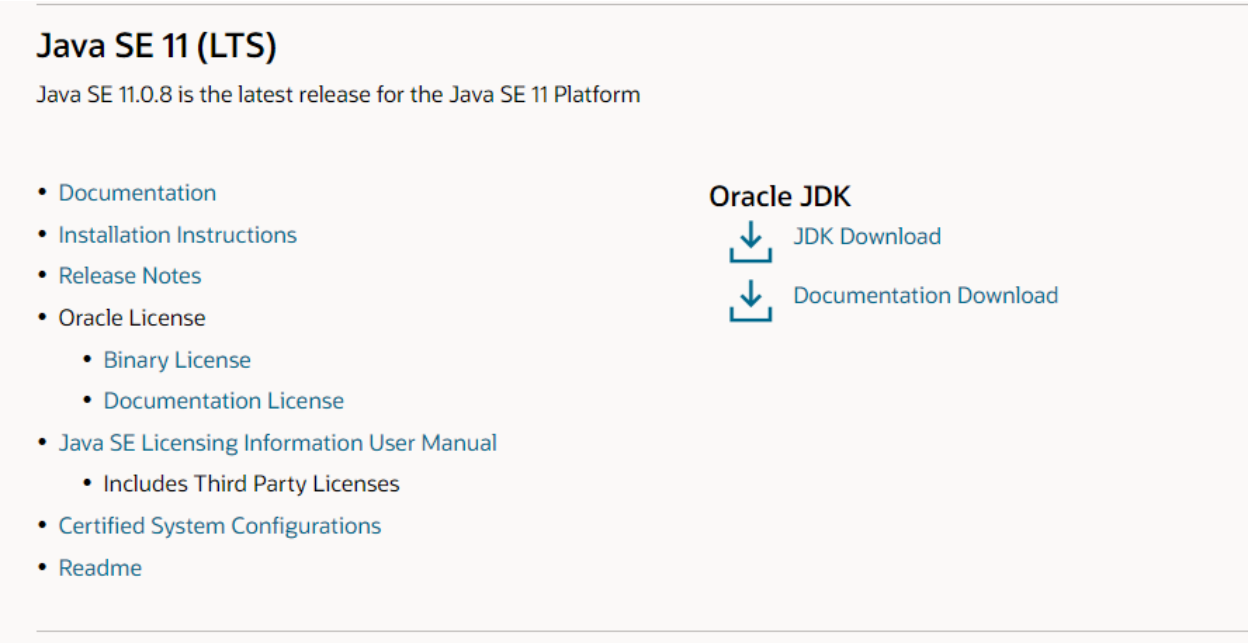
Contents

1 Java

1.1 Java JDK and JRE

1) Access the next link:

   http://www.oracle.com/technetwork/java/javase/downloads/index.html



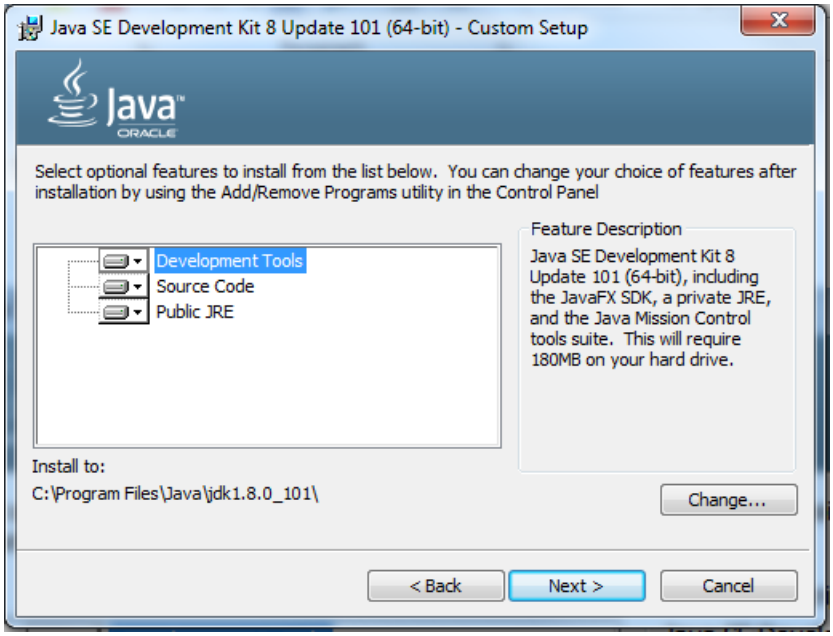
2) Click on the icon which is above Java Platform (JDK). You will be redirected to Java downloads.

Java SE Development Kit 11.0.8

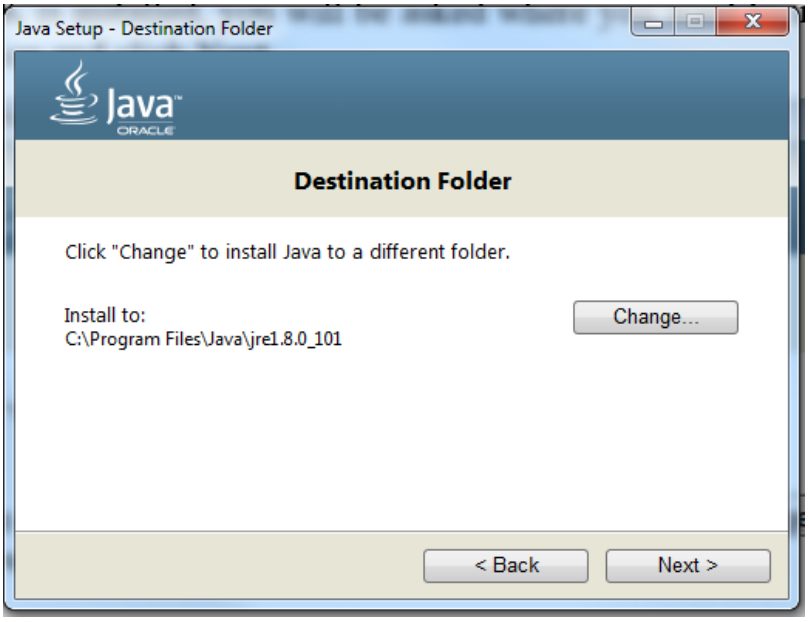
This software is licensed under the Oracle Technology Network License Agreement for Oracle Java SE

Product / File Description	File Size	Download
Linux Debian Package	148.77 MB	jdk-11.0.8_linux-x64_bin.deb
Linux RPM Package	155.45 MB	jdk-11.0.8_linux-x64_bin.rpm
Linux Compressed Archive	172.66 MB	jdk-11.0.8_linux-x64_bin.tar.gz
macOS Installer	166.84 MB	jdk-11.0.8_osx-x64_bin.dmg
macOS Compressed Archive	167.23 MB	jdk-11.0.8_osx-x64_bin.tar.gz
Solaris SPARC Compressed Archive	186.49 MB	jdk-11.0.8_solaris-sparcv9_bin.tar.gz
Windows x64 Installer	151.73 MB	jdk-11.0.8_windows-x64_bin.exe
Windows x64 Compressed Archive	171.16 MB	jdk-11.0.8_windows-x64_bin.zip

3) Click on the link Accept License Agreement.
4) Click on the link which corresponds to your version of the Operating System. In the example the version which is used corresponds to Windows x64 and the file is named jdk-11.0.8_windows-x64_bin.exe.
5) After *java-version*.exe is pressed, a file with the same name will be downloaded.
6) Click on *java-version*.exe.
7) You will be asked the next question: Do you want to allow the following program to make changes to this computer? Click Yes.
8) Click Next.
9) You will be asked where you want to install Java. Use the default location and click Next.



10) After the JDK is installed, you will be asked where you want to install the JRE. Use the default location and click Next.



11) After the installation is completed click Close.

1.2 Set JAVA_HOME and JAVA_JRE variables

1) Click Start.
2) Right-Click on Computer.
3) Select Properties.
4) Click on Advanced System Settings.
5) Click on Environment Variables.
6) Under System Variables click New.
7) In the text field associated with the name of the variable insert JAVA_HOME and in the field associated with the value of the variable insert C:\Program Files\Java*java_version*;.
8) Click OK.
9) Under System Variables click New again.
10) In the text field associated with the name of the variable insert JRE_HOME and in the field associated with the value of the variable insert C:\Program Files\Java*java_version*;.
11) Click OK.

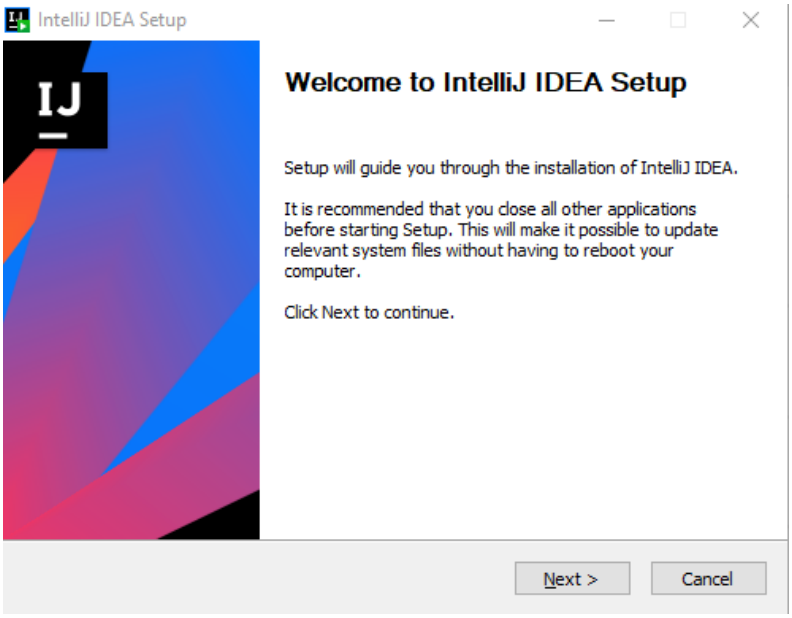
2. IntelliJ IDEA

We recommend using IntelliJ IDEA as the IDE for developing your applications during the laboratories. As a student, you can benefit of a free educational license for IntelliJ IDEA, by applying here https://www.jetbrains.com/community/education/#students -> Click on *Apply now* -> Fill in the requested information and click on *Apply for free products.*

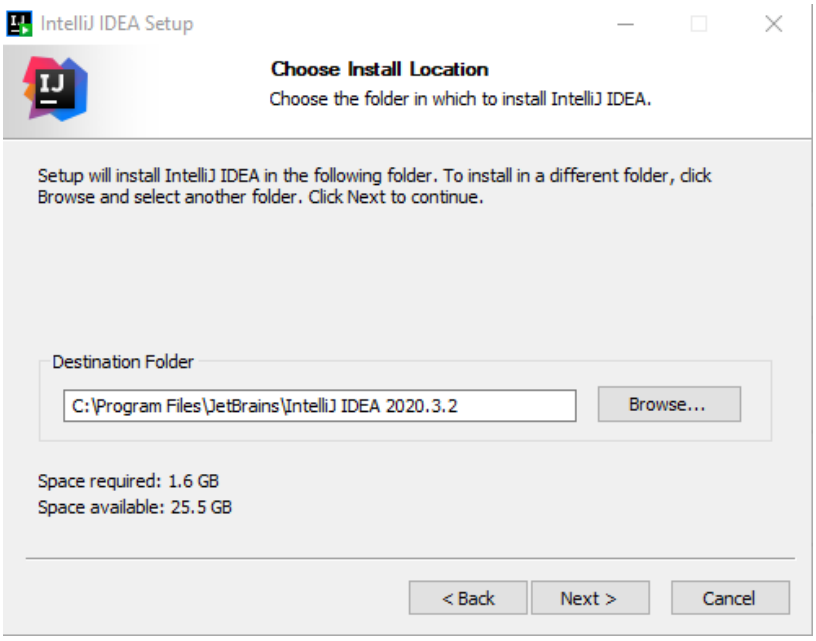
After obtaining the license, download Intellij IDEA Ultimate "*last version*". Click on the downloaded file. You will be asked whether you allow the program to make changes to this computer -> Click Yes.

Then, follow the steps below:

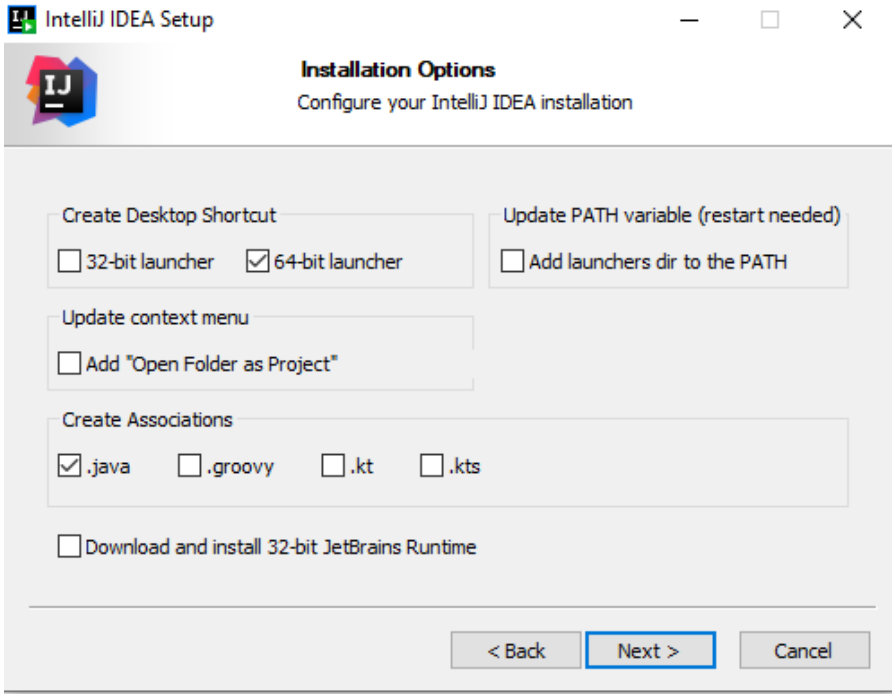
1) Click next



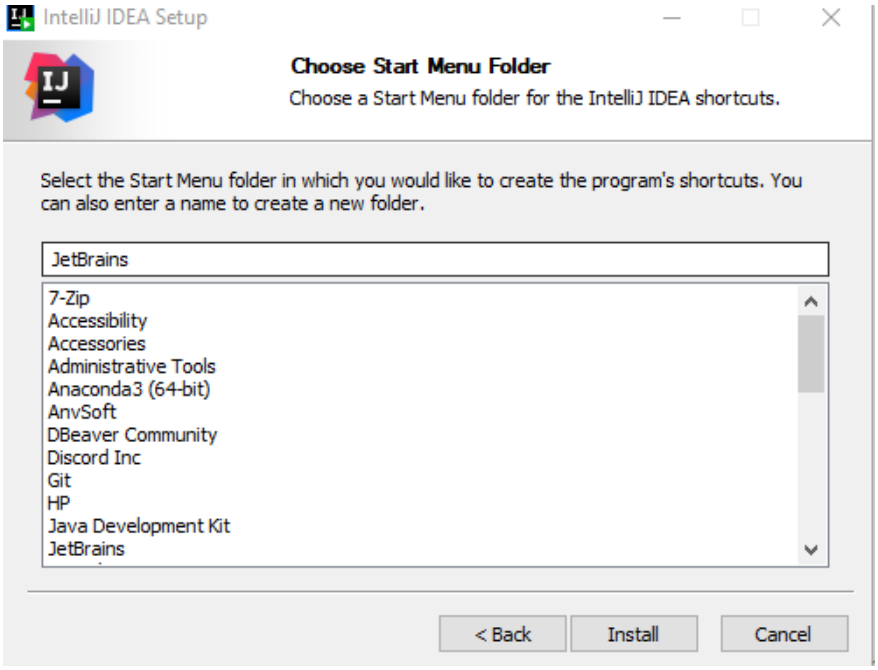
2) You will be asked where you want to install Intellij IDEA. Use the default location and click Next.



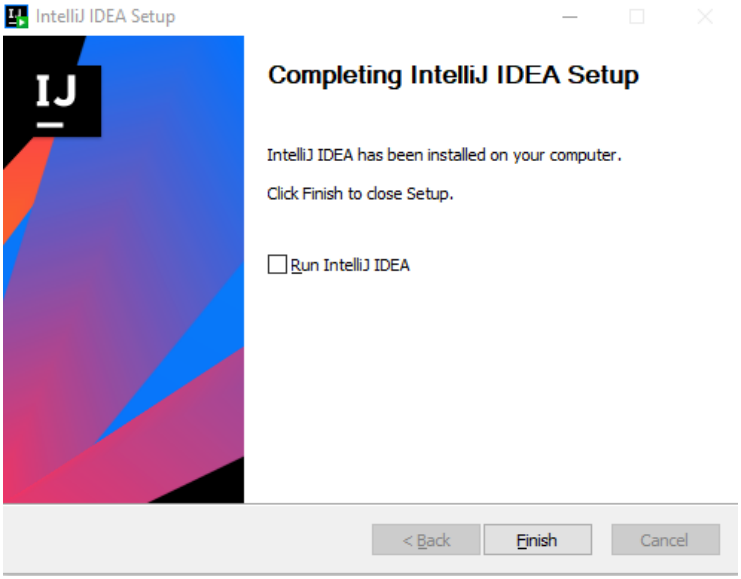
3) Configure the installation options (see an example in the image below) and click Next.



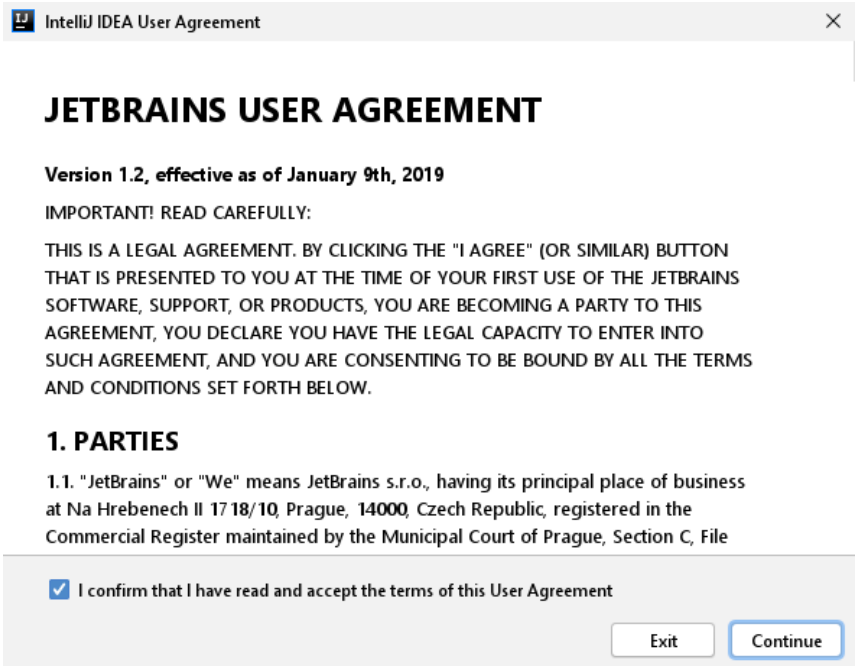
4) Click Install.



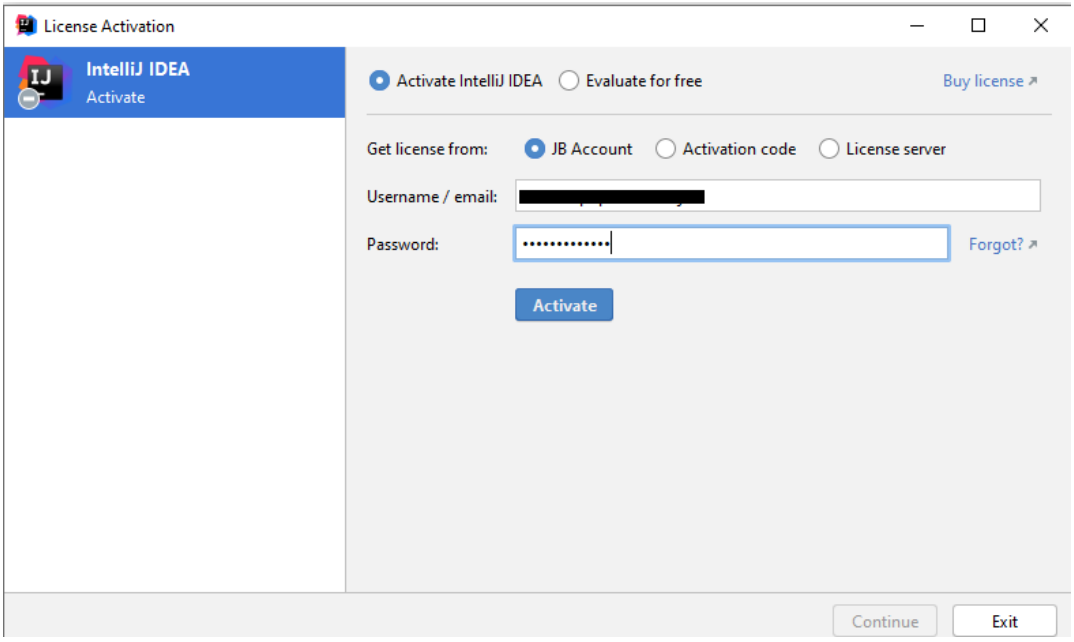
5) After the installation completes click on *Finish.*



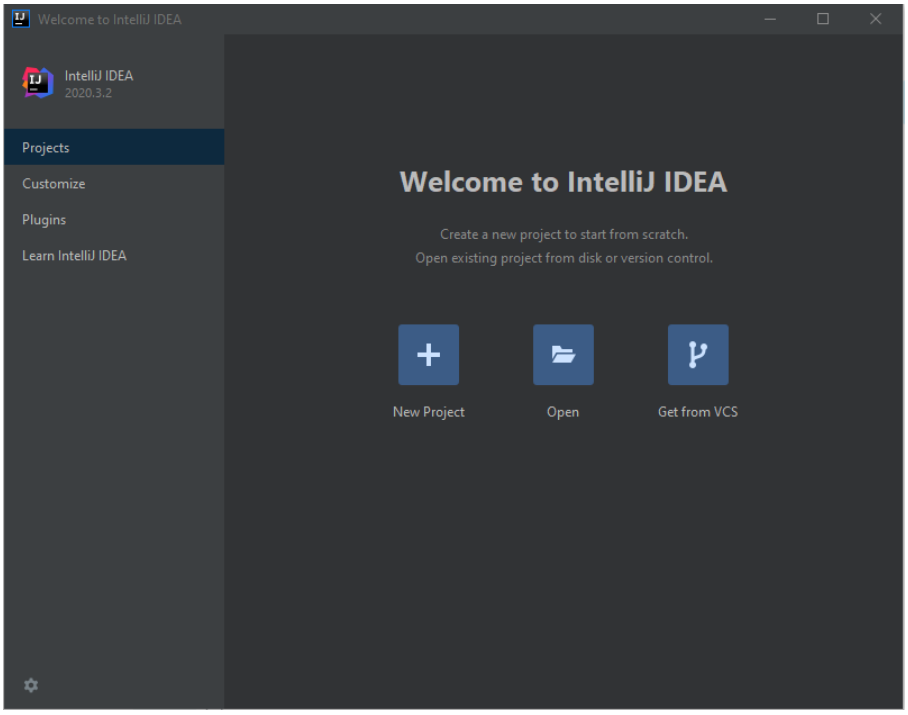
6) Run IntelliJ IDEA. When you first run it after the installation completes you will be prompted to accept the terms of the User Agreement – check the option and click *Continue*.



7) Activate IntelliJ IDEA by inserting the username/email and password you used when creating the account for obtaining the students license. Click *Activate*.



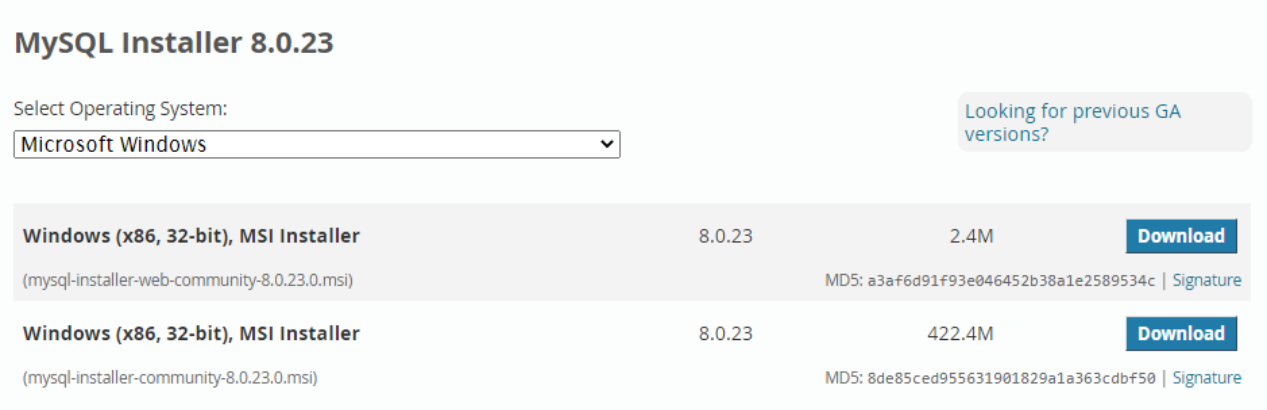
8) After the activation completes, the IDE will open.



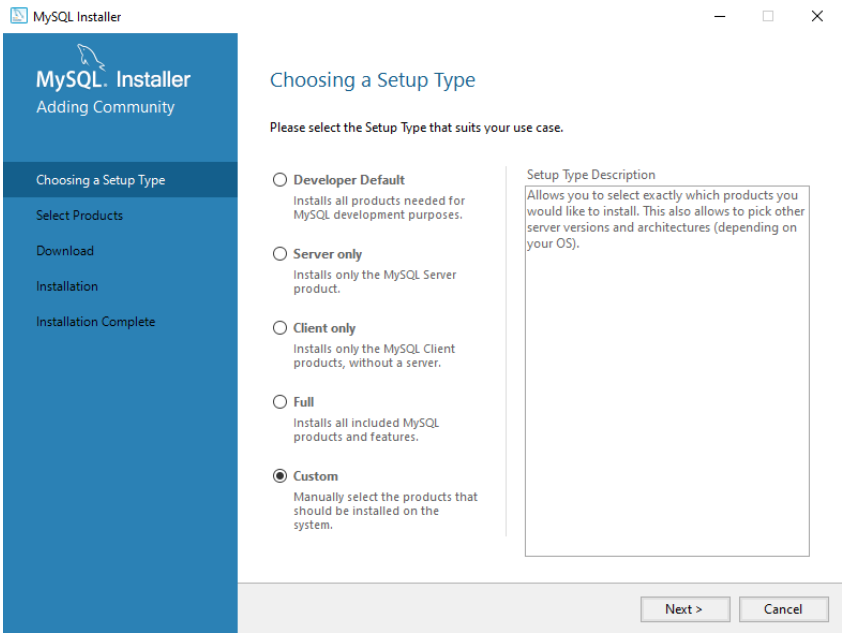
3. Database Server

A database server must be installed on the local machine. A MySQL server can be used to run locally the projects. For installing the MySQL server follow the next steps:

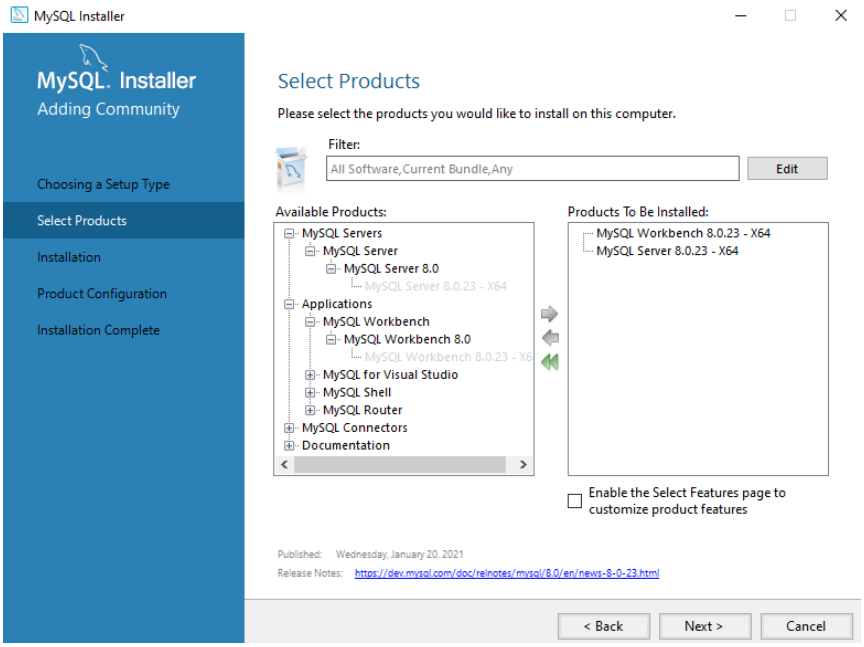
1) Click on the next link: https://dev.mysql.com/downloads/windows/installer/.



2) Click on the second Download button.
3) Click on No thanks, just start my download.
4) Click on the file downloaded file.
5) You will be asked whether you allow the program to make changes to this computer -> Click Yes.
6) You will be asked to select the Setup Type that suits your use case. Select Custom and click Next.



7) You will be redirected to "Select Products and Features". Select "MySQL Server 8.0.23 – X64" and "MySQL Workbench 8.0.23 – X64" and click Next.



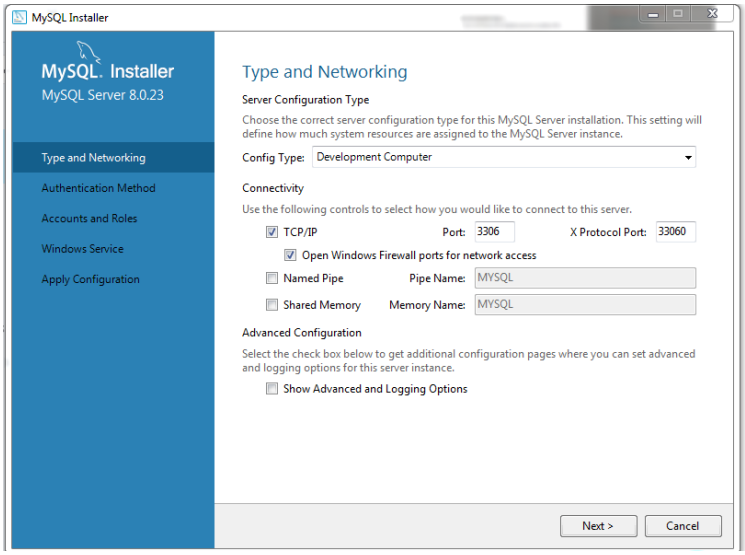
8) Click Next.

9) Click Execute.

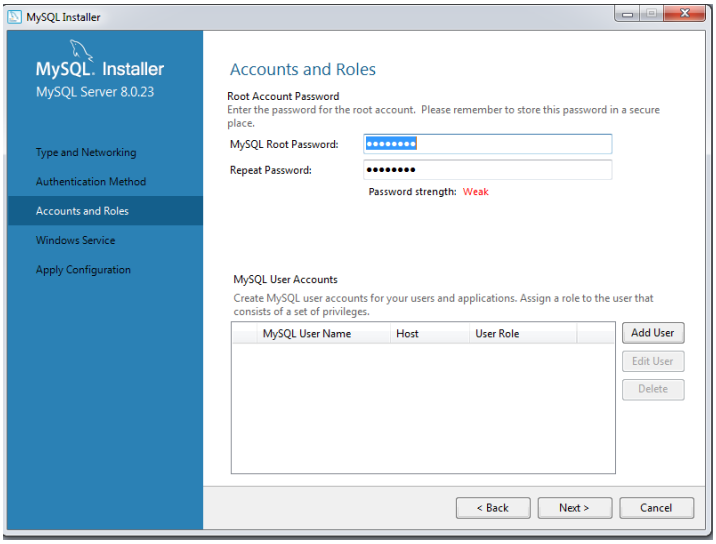
10) Click Next.

11) Click Next and follow the steps for the configuration of the MySQL Server. At this stage pay attention to the following:

- The port on which the MySQL server is running on (i.e. 3306 is the default port)



- The password for the root account – this must be set by you and make sure you remember it as it will be used for further connections to the MySQL server.



12) Click Finish.

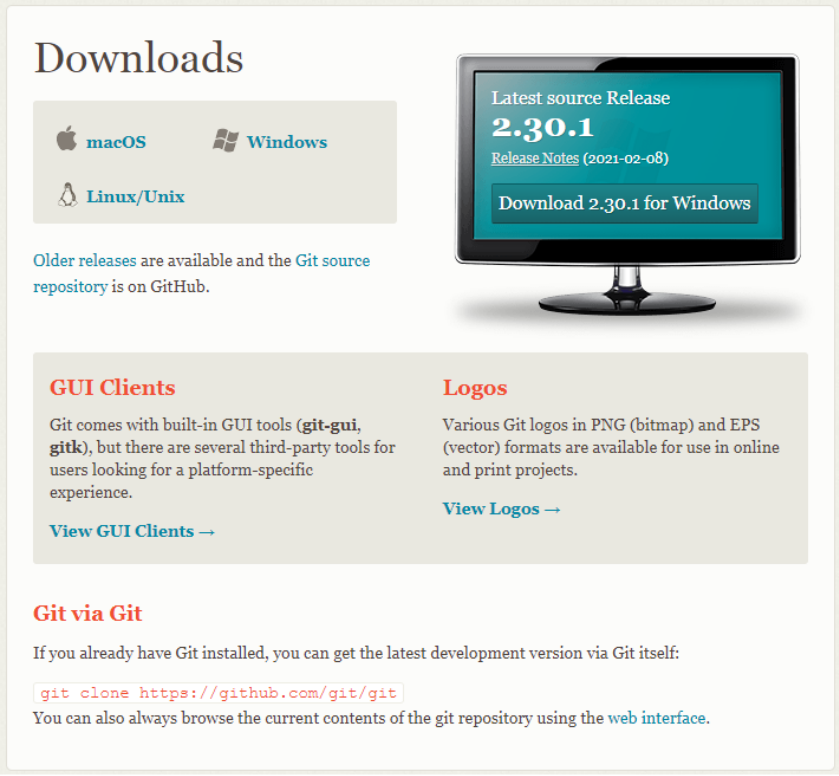
4 Git and Maven Projects

Modern software projects development requires effort from large teams of developers that have to collaborate in order to integrate their work and create the final product. Coordinating and tracking the changes between multiple source files is a difficult task, thus an automatic tool was developed in 2005, initially for the development of the Linux Kernel, and, since then, it has penetrated all levels of software development. The tool is a version control system (VCS), named GIT, which tracks changes of computer files and helps coordinating several people who work on those files.

Furthermore, large applications often encounter problems with the software build settings as well as with the dependency description. In order to address these problems, another automatic tool was created, MAVEN that defines conventions for the build procedure and uses an XML file to describe the software project, dependencies, external modules, components and plug-ins.

4.1 Git installation

1) Click on https://git-scm.com/downloads.
2) Select your operating system.



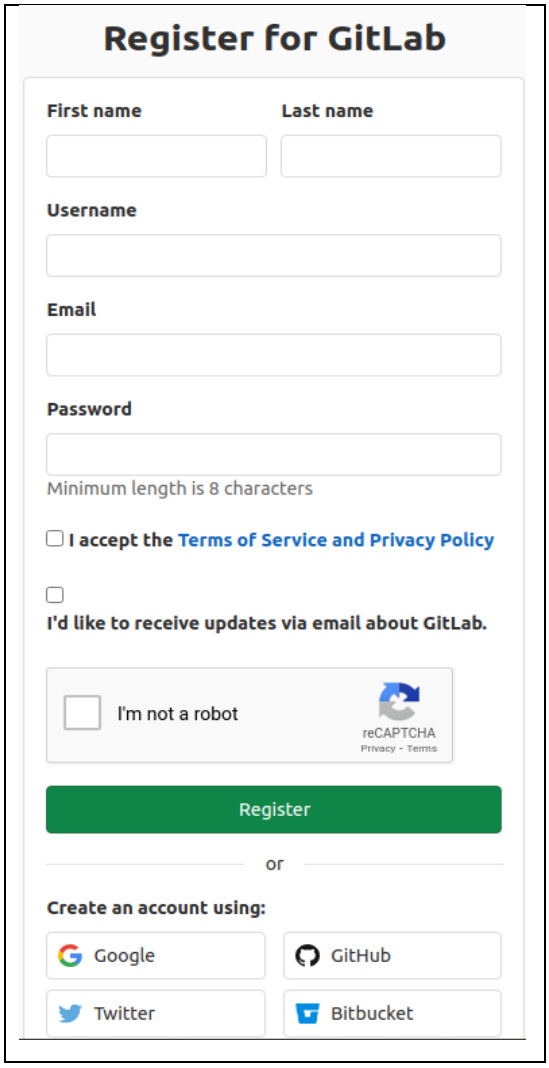
3) If you select Windows, a file called *Git-version-64-bit.exe* should be downloaded. In the case you select another operating system or if your system is on 32 bits then a file with a similar name should be downloaded.

4) Click on this file and follow the default installation guidelines, except for the step where you are asked which terminal emulator you want to use. Select the second option.

4.2 Create Account on GitLab

1) Click on https://about.gitlab.com/.

2) In the right corner, click on Register. You will be asked to introduce your personal information. Or, if you already have an account, just *Sign In.*



3) In the next window, you can choose the role of "Software Developer". In the next checkbox, you can choose the "Just me" option.

4) Following up, you will have to create a group. The name of the PRIVATE group must be of the format: *PT2021_GroupNumber_LastName_FirstName*

(e.g. PT2021_30221_Popescu_Ioan)

5) For now, do not invite any other teammates to have access to the group, as we will do this later.

6) Click *Create group*.

7) For the next section, *Create/import your first project*, just write a random name in the *Project name* text field, like *test project.* We will create a project for each assignment, but in a few minutes.

8) Click *Create project*

9) You can choose any experience level you consider having, because this document will detail all the steps necessary to complete your assignments.

10) After selecting the experience level, you will see the *home screen* of the group you have just created.

    a. If you already had an account and just signed in, you must create the group we have just talked about. Go to: *Groups → Your Groups → New Group* then create a new PRIVATE group with the format: *PT2021_GroupNumber_LastName_FirstName*

    (e.g. PT2021_30221_Popescu_Ioan)

11) Now you must give access to your group, to the PT lab assistants. On your Group page, go to: *Members → Invite Member →* and offer **Maintainer** rights for the user: utcn.dsrl@gmail.com .

12) Inside the group, you can create your own projects for different applications of the PT lab. Remember to keep the same naming conventions for the projects, considering the following format:

PT2021_GroupNumber_LastName_FirstName_Assignment_Number

4.3 Basic Instructions

4.3.1 Create a project from scratch

1) Create the folder *PT2021_GroupNumber_LastName_FirstName_Assignment_Number* on *D:*.

2) Right click on this folder and click *Git Bash Here.*

3) Write the next commands:

    a) git init

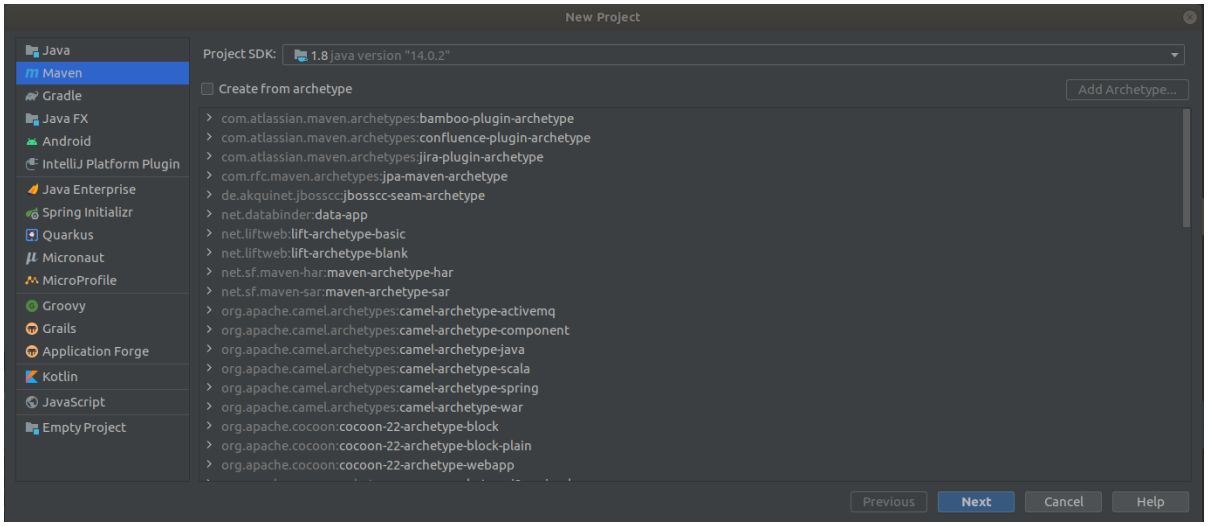
Note: the command will create a new local repository with a default master branch. Notice that after executing this command, a hidden .git folder will be created inside the current folder representing the repository where git stores all necessary files. From now on, it will be possible to track all the changes that will be performed to the files from the original folder. The original folder is considered to be the working directory, while the .git folder is referred as the repository that tracks the made changes (for more details check this link).

b) git remote add origin

https://gitlab.com/group_name/project_name.git

Note: this command will connect the local repository created using the "git init" command with the remote repository's origin (for more details check this link).

4) Open IntelliJ, select *File -> New -> Project.* Then choose *Maven* from the list from the left, choose the appropriate Java SDK, then click on *Next.*



5) Instead of using the default location suggested, use this one:

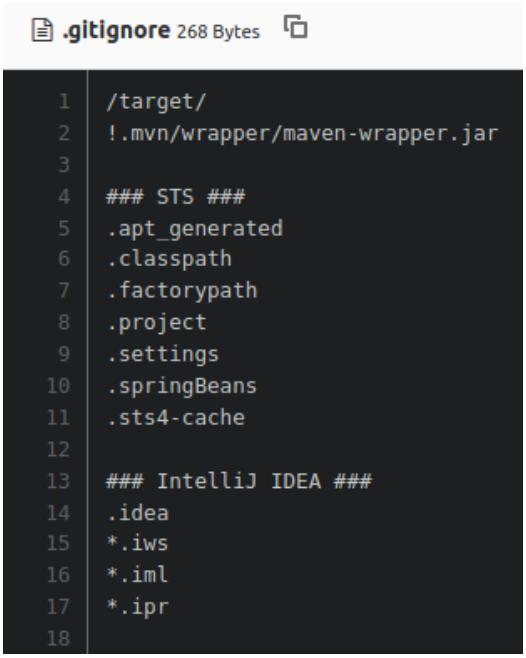
PT2021_GroupNumber_LastName_FirstName_Assignment_Number

Do not forget to also choose the appropriate name for your new Project, most likely using the format we requested (unless it is a test project, not for one of your assignments).

6) Click *Finish.*

7) Now, before pushing anything to the remote project, you must create a .gitignore file in the project, which tells git which files to ignore when committing and pushing to your remote

projects. You don't want unnecessary files, such as IDE configuration files, to be pushed, because they are strictly relevant for your local system. Just create a file named "*.gitignore*" and write the following lines:



8) Right click on the folder

PT2021_GroupNumber_LastName_FirstName_Assignment_Number and click Git Bash Here; then introduce the next commands:

a) git add .

Note: the command will mark any changes that you have made to your project files (e.g. creating/modifying/deleting files) as staged so that they can be included in the next commit (for more details check this link).

b) git commit –a –m "initial commit"

Note: the command commits any files you have added with the git add command and also commits any files you have changed since then – at this step the changes are saved only locally. An explanation message is given in order to document what has been added/changed (for more details check this link).

c) git push –u origin master

Note: the command sends the committed changes to your remote repository (for more details check this link).

4.3.2 Update the project

1) Create a new class named *Main* in the same package as the class *App*.
2) Navigate to folder *PT2021_GroupNumber_LastName_FirstName_Assignment_Number*, *right click* and select *Git Bash Here*
3) Insert the next commands:
    a) **git add .**
    b) **git commit –a –m "add new class"**
    c) **git pull origin master**

Note: the command fetches and merges changes on the remote server to your working directory (for more details check this link).

    d) **git push –u origin master**

4) You can always see the modifications that were not committed yet by using:
    a) **git status**

4.3.3. Create Groups for projects

Although you have probably created your first group and project on GitLab (if you have followed the instructions from section 4.2), we will repeat the basic steps for working with groups on GitLab.

1) You will have a group corresponding to all your PT projects for this semester. To create a new group, go to:

    Groups → Your Groups → New Group then create a new PRIVATE group with the format:

    PT2021_GroupNumber_LastName_FirstName

    (e.g. PT2021_30221_Popescu_Ioan)

2) Now you must give access to your group, to the PT lab assistants. On your Group page, go to: *Members → Invite Member →* and offer **Maintainer** rights for the user: utcn.dsrl@gmail.com .
3) Inside the group, you can create your own projects for different applications of the PT lab. Remember to keep the same naming conventions for the projects, considering the following format:

    PT2021_GroupNumber_LastName_FirstName_Assignment_Number

4.3.4. Create and work with a new branch

The real value of working with git, is the power of *branches*. They allow multiple developers to work simultaneously on the same project, on different features, and then to merge all the new changes in the master branch.

1) **Create a branch production**

The first step towards working with branches, is to create a new branch. When you create a new branch, it will automatically be initialized with the currently existent code. Then, while inside that branch, all changes will be added only on that branch.

In order to create a new branch:

- pull all the changes from the remote project, in order to be up to date:
  - *git pull*
- create the branch on your local machine and switch directly to that branch:
  - *git checkout -b <branch-name>*
- push the newly created branch to the remote repository:
  - *git push origin <branch_name>*

An example to create the branch production is the following:

```
#git pull
#git checkout -b production
#git push origin production
```

2) **switch between branches**

When working with multiple branches, it is important to keep track of all the available branches, and to always know on which branch you currently are.

- To bring locally meta-data information about existing branches:
  - *git fetch --all*
- In order to see all existent branches:
  - *git branch -a*
- In order to see on which branch you currently are:

- *git status*

- *in order to switch from a branch to another, use the above-mentioned command:*
  - *git checkout -b <branch-name>*
    - *if the branch with that name is already existent, it will just switch to that one, instead of creating a new one*

You can try to create a new branch, make some small changes, then switch back to the master branch, and see that that change is not present in the master branch.

3) **commit changes to new branch**

When making changes on a new branch, you must always commit and push them to the remote branch, just like working on master.

- First, make sure you are on the right branch:
  - *git status*
- Repeat the same process as if you were working with master. However, pay attention to the names:
  - *git add .*
  - *git commit -a -m "commit message"*
  - *git push -u origin <branch-name>*
- And now, your remote branch <branch-name> contains all the changes you have pushed.

4) **merge branch with master**

An important step when working with branches, is to always keep the master branch up to date with the latest **working and functional** code from your other branches. Merging two branches, as the name suggests, is the process of merging the code from two branches. If the branches contain changes on different parts of the code, the merge process will work instantly. If both branches contain changes on the same parts of code, git will require you to solve the conflicts: from the two modifications, you must choose the one which you want to remain in the final version.

DO NOT FORGET: do not merge code which is not working properly, or which is not tested, into master. Master must always contain the latest functional version of your project.

In order to merge two branches:

- git merge <branch-with-new-changes> <branch-to-be-updated>

For a more in-depth explanation of branches and how they can be manipulated to serve your needs, we suggest checking the following tutorial:

https://git-scm.com/book/en/v2/Git-Branching-Basic-Branching-and-Merging

4.3.5. Getting git to work with a proxy server

1) In the UTCN laboratories you need to set the proxy server in order to use GIT bash
2) Open Git Bash
3) Insert the following commands:
    a) git config --global http.proxy http://proxy.utcluj.ro:3128
    b) git config --global --get http.proxy
4) In order to unset the proxy, use the following command:
    a) git config --global --unset http.proxy

4.3.6. Getting MAVEN to work with a proxy server

1) In the UTCN laboratories you need to set the proxy server in order to use MAVEN projects
2) Go to Windows Explorer-> Drive C-> Users -> *Your User* -> .m2
3) Create the folder **conf**
4) Go to conf folder and create the file **settings.xml** with the following content:

```
<settings xmlns="http://maven.apache.org/SETTINGS/1.0.0"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:schemaLocation="http://maven.apache.org/SETTINGS/1.0.0
                    http://maven.apache.org/xsd/settings-1.0.0.xsd">
  <localRepository/>
  <interactiveMode/>
  <usePluginRegistry/>
  <offline/>
  <pluginGroups/>
  <servers/>
  <mirrors/>
  <proxies>
    <proxy>
      <id>myproxy</id>
      <active>true</active>
```

```
    <protocol>http</protocol>
    <host>proxy.utcluj.ro</host>
    <port>3128</port>
    <username></username>
    <password></password>
    <nonProxyHosts>localhost,127.0.0.1</nonProxyHosts>
   </proxy>
  </proxies>
  <profiles/>
  <activeProfiles/>
</settings>
```

5) Go back to folder **.m2**
6) Delete the folder **repository**
7) Open **Eclipse**
8) Go to **Window->|Preferences->|Maven->|User Settings**
9) At the **User Settings** tab browse for the **settings.xml** file created at step 4
10) Click **Apply** and **OK**
11) Go on the project, right click and go to **Maven->Update Project**