Table of Contents

[Initial cloning 1](#_Toc34035725)

[Making changes and pushing them to GitHub 1](#_Toc34035726)

[Setting up a project in Eclipse 3](#_Toc34035727)

[Adding a 32-bit Java compiler to Eclipse 3](#_Toc34035728)

[Creating a new project 5](#_Toc34035729)

[Adding libraries to the build path 7](#_Toc34035730)

[Creating a run configuration 9](#_Toc34035731)

[Resolving “JRE not compatible with workspace .class file” error 17](#_Toc34035732)

# Initial cloning

Note: this only needs to be done once.

1. open the Windows Command Prompt (Windows key+R and then type cmd) and navigate to the folder where you want the code to be located
2. run the following command to clone the ePTM repository that’s hosted on GitHub, entering your GitHub username and password if prompted: git clone https://github.com/cvclcm/ePTM\_v2.git

# Making changes and pushing them to GitHub

1. within the Windows Command Prompt, navigate to the “ptm” folder that was created when you cloned the repository
2. pull from the repository to ensure that you have the latest version of the code: git pull origin master
3. if you get a merge conflict, you must first resolve it and then commit the code before the pull can be completed. This procedure for this is:
   1. for each file that Git reports as having a conflict, search for “>>>>”, which marks the conflicting code that Git couldn’t figure out how to automatically merge. The file will include both the GitHub version and the local version of the conflicting code; you have to resolve the conflict manually by editing the code to keep just the version you want.
   2. commit the code with the conflicts resolved: git commit -am “*Your commit message here*"
4. make your code changes (in Eclipse or a text editor)
5. commit your changes to your local repository: git commit -am “*Your commit message here*”
6. push up your changes to GitHub: git push your\_repo master

# Setting up a project in Eclipse

## Adding a 32-bit Java compiler to Eclipse

Note: this only needs to be done once.

First, a 32-bit Java 8 JDK must be installed on your machine. If you do not already have it, you can download it from Oracle’s website, e.g., jdk1.8.0\_201.

Second, the compiler must be added to the list of installed compilers in Eclipse. From the main toolbar in Eclipse, select Window -> Preferences and navigate to the Installed JREs tab:

A screenshot of a social media post

Description automatically generated

Next, click the “Add” button and select “Standard VM”:

A screenshot of a cell phone

Description automatically generated

Click the “Directory” button, then navigate to the home directory of the 32-bit JDK (e.g., C:\Program Files (x86)\Java\jdk1.8.0\_201), click the “Finish” button, and then click “Apply and Close” in the subsequent dialog:

A screenshot of a cell phone

Description automatically generated

## Creating a new project

From the File menu, select “Java Project”:

A screenshot of a cell phone

Description automatically generated

Uncheck “Use default location” and specify the top-level ptm folder of the code that you cloned from GitHub. If it is not already selected, select “Use a project specific JRE”, choose the 32-bit JDK you installed previously, and click Next:

A screenshot of a social media post

Description automatically generated

Use the default options in the subsequent dialog box and then click Finish:

A screenshot of a social media post

Description automatically generated

## Adding libraries to the build path

Right-click on the project in the project navigator pane and select Build Path -> Configure Build Path:

A screenshot of a cell phone

Description automatically generated

Select the Libraries tab, select Classpath, and then click “Add JARs”:

A screenshot of a cell phone

Description automatically generated

In the Jar Selection dialog box, select your project, drill down into the lib folder containing the \*.jar files, and Shift-select all of them:

A screenshot of a social media post

Description automatically generated

## Creating a run configuration

Select Run -> Run Configurations from the menu:

A screenshot of a cell phone

Description automatically generated

Click on the “New launch configuration” icon, which is the blank page with a plus:

A screenshot of a cell phone

Description automatically generated

Fill out each tab as shown below. Note: the project name may be different from what is shown below depending on what you named your project when you were creating it, and any path names will need to be modified to match your machine. The launch configuration name can be anything:

A screenshot of a cell phone

Description automatically generated

A screenshot of a cell phone

Description automatically generated

A screenshot of a cell phone

Description automatically generated

A screenshot of a cell phone

Description automatically generated

A screenshot of a social media post

Description automatically generated

A screenshot of a social media post

Description automatically generated

A screenshot of a social media post

Description automatically generated

## Resolving “JRE not compatible with workspace .class file” error

Assuming you chose the “Use a project specific JRE” and specified the 32-bit JRE, you shouldn’t run into this problem. But in the event you do, you may need to configure the project to use the correct 32-bit JRE. Right-click the project name in the navigator and choose Properties:

A screenshot of text

Description automatically generated

Select “Java Compiler” in the pane, click “Enable project specific settings” and change the “Compiler compliance level” and other settings so they match the screenshot below:

A screenshot of a cell phone

Description automatically generated