Introduction to Eclipse

We will be using Eclipse in this course to create and run Java programs. Eclipse is a platform that can be used for Java application development as well as other things. A separate handout describes how to obtain and install Java and Eclipse. This handout will take you through the process of creating, compiling and running a Java program using Eclipse.

Eclipse is installed in both CIS labs and you're welcome to work there. It is your responsibility to remove your files from the lab computer when you're leaving the lab.

Specifying the Eclipse workspace

When you start Eclipse, it asks you to specify the directory in which to store your projects. By default, it uses a directory called workspace within the directory in which Eclipse is installed. To switch the workspace to another location, go to the File->Switch Workspace and select the desired folder.

When you're working on a lab computer, don't forget to remove your files from the lab computer when you're done; otherwise someone else could get access to them.

Eclipse Perspectives and Views

Eclipse workbench provides several views and perspectives to access information regarding a project. Each perspective is a collection of views designed for a certain kind of project. We are going to be using the Java perspective, which is usually the default one.

A view is a window that displays specific kinds of information regarding the tasks. We will need the following views within the Java perspective (see Figure 1 for illustration):

- **Navigator** - provides access to project files. It is not displayed by default, so you will need to **activate it by choosing Window-> Show View->Navigator.**
- **Problems** - displays syntax and run time errors;
- **Console** - is the program interaction window; it displays the textual input/output from a running Java program.

Select the Java perspective by choosing Window->Open Perspective->Java. Then, select the Navigator view by choosing Window-> Show View->Navigator to see and access the contents of each project directory. The Navigator view is displayed on the left side of the Eclipse window. Select the Console view in the same manner, and notice that it usually appears on the bottom.

Creating a new project

To create and run a Java program in Eclipse you must form a project for it.

1. Create a new Java Project by selecting File->New->Java Project.
2. In the dialog window that shows up:
   - Enter a project name into the Project name field;
   - Select the first option (Use project folder as root for source and class files) from the Project Layout box.
   - Only when creating your very first project, follow *Configure default* link on the right inside the Project Layout box. In the dialog box that follows, select the **Project** option and click OK
3. Click on Finish.

A project folder is now added to your workspace and you can see it in the the Navigator view on the

left side of your Eclipse window.

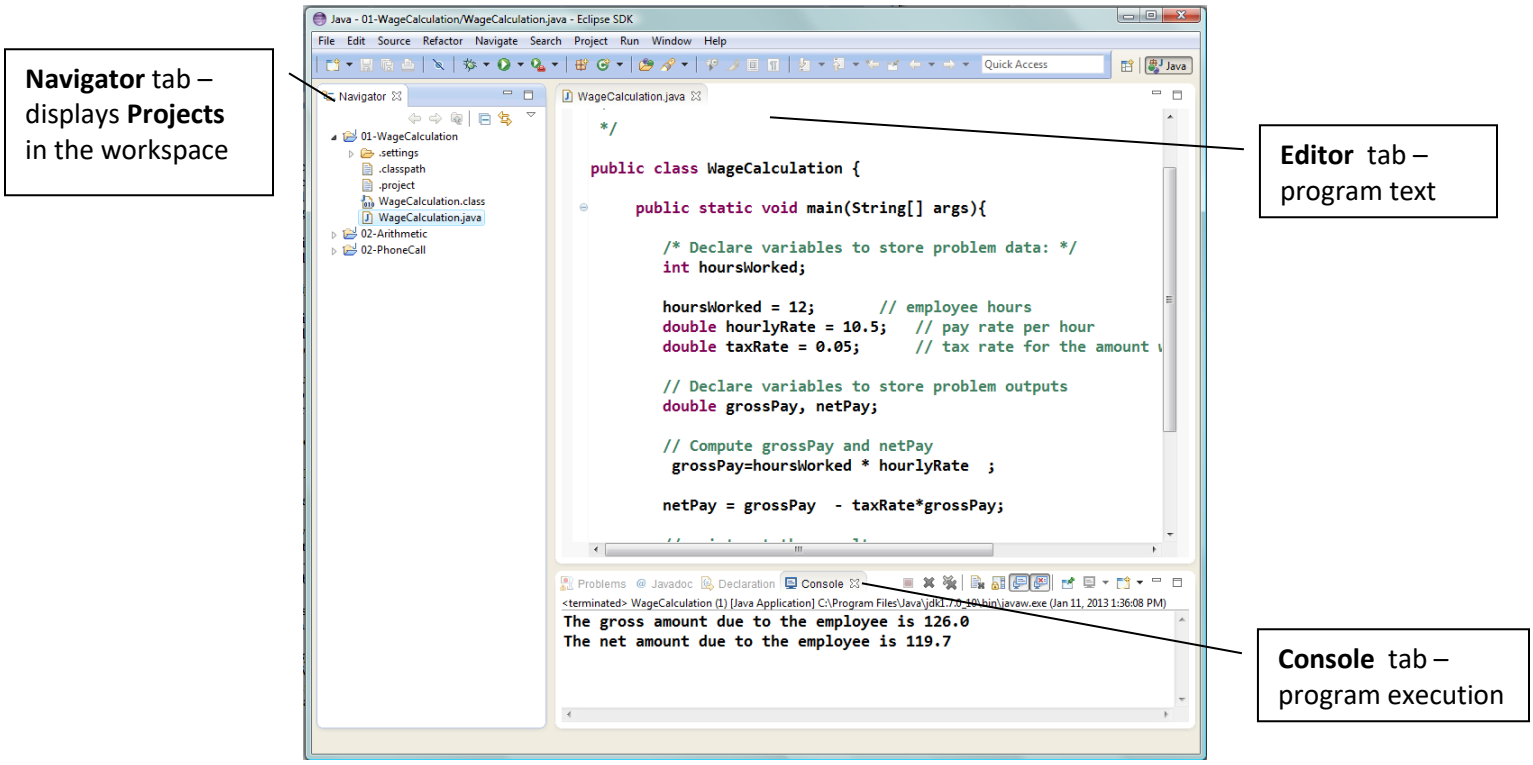

Figure 1 - Navigator, Console and Text Editor views.

Adding Java code to the project.

A Java program is a collection of classes. Each class is defined in a separate `.java` source file and later compiled into a `.class` file. A `.java` file must have the same name as the class it defines.

We'll demonstrate how to add the class definition presented in Figure 2 to the project. First, make sure the project is selected in the Navigator view, then select File->New->Class from the menu. Enter "Hello" into the Name field. Verify that there is a check box indicating that you would like Eclipse to include a `public static void main(String[] args)` method. Click on Finish.

A Java editor for Hello.java will open. In the main method enter code as shown in Figure 2.

```
/* A very simple Java program. Author T. Babaian.

*/
public class Hello {
      public static void main (String args[]) {
            System.out.println("Hello, world!");
      }
}
```

Figure 2 – The text for the Hello program

Save using Ctrl-s. This automatically compiles Hello.java.

The Eclipse editor includes a lot of very useful features like syntax highlighting, parenthesis matching and

much, much more. One of the most useful ones is the auto-formatting feature, which modifies the Java code in the editor window so that it is properly indented. As we will learn, consistent indentation is one of the hallmarks of professionally-looking code, as it makes the code much easier to understand. Ctrl-Shift-F, or Source->Format will format the selected piece of code for you.

Compilation and Syntax errors

Eclipse periodically saves your Java file and compiles it each time it saves it. To save a project press Ctrl-s or click on the Save button***. Note, that if you have src and bin folders inside the project folder, your Java file must be in placed in the src folder, or Eclipse will not compile it.***

If the compiler detects syntax errors in the program, the error messages will appear in the Problems view pane on the bottom of the window. Each line containing an error will also be marked in the editor pane by a red circle with an X inside. To experience syntax error detection, erase the semicolon in the end of the `System.out.print` statement and save the file. Click on the displayed error mark and read the error description that appears.

Sometimes, a red error mark on the border of the editor window would also have a yellow bulb sign displayed next to it (a Quick Fix sign). This designates that Eclipse has suggestions about how the error could be fixed. To see the suggested fixes, click on the Quick Fix sign. To make Eclipse carry out its suggestion, simply select it by a mouse click. To see how the Quick Fix feature works, rename the class from Hello to Mello, for example. The Quick Fix list should contain at least 2 options: renaming the compilation unit (i.e. the file) or renaming the type (i.e. the class), so that the file name matches the name of the class.

Running the program

Once the program has compiled successfully, it is ready to be run. Most of the time, all you need to do is to right-click on the `.java` file (which must have a main() method to be runnable) in the Navigator pane, and select **Run As->Java Application.**

Notice a window pane appearing on the bottom. This Console view shows the output from the running program and allows the user to enter input data. Right now the Console should display the string:

Hello, world!

Creating a project for an existing java file

Here's how to create a project for an existing java file stored somewhere in your file system. In Eclipse, select File->New->Project->Java Project. In the next window, specify the name of the project and click on the Finish button.

Now, you must add the existing file to your project. You can do it by dragging-and-dropping the java file from the Windows file explorer, from desktop or from another Eclipse project folder into your created project folder in the Navigator view tab.

Note again, that if you have **`src`** and **`bin`** folders inside the project folder, your Java file must be in placed in the **`src`** folder, or Eclipse will not compile it.