# COMPSCI 230 Tutorial 1
# Eclipse IDE

This tutorial provides basic information on how you can use Eclipse to assist you while programming and debugging in Java.

## 1. What is Eclipse?

It is an Integrated Development Environment (IDE) where you can create, edit, compile, run, and debug your projects. It supports not only Java but also C, C++ etc. It is an open source supported by IBM and can be run on any platform with a Java Virtual Machine (JVM).

## 2. Why use Eclipse?

It makes the programmer's job easier. The features are provided in an integrated fashion which makes programming tasks easier in comparison to command utilities.

Some important features include -

- Code generation.
- Syntax error checking.
- Formatting and indentation.
- Spelling mistakes.
- Ease of debugging.
- Suggestions for errors fixing, etc.

## 3. How to start Eclipse?

Firstly we need to boot into the linux environment called Ubuntu. In order to boot Ubuntu you must tell the boot loader **when starting the computer** which environment to operate in.

***Note 1:** *To do this from a windows environment within the computer labs requires rebooting the computer.*
***Note 2:** *Once booted into Ubuntu, you need to manually enter your university credentials into 'Netlogin' to allow for access to the internet.*

In CS labs, Eclipse is available in Ubuntu by selecting -

**Applications -> Programming -> Eclipse**

Eclipse is also available in OSX or windows. You can start eclipse from windows by going to –

**Start -> Program Files -> Development -> Development Environment -> Eclipse**

## 4. How to create a Project?

To create a project: File>New>Java Project OR File>New>Project and choose 'Java Project' from the list. You can then either choose to: 'Create a new project in workspace'. Enter a name for your project and click 'Finish' OR 'Create project from existing source'. Enter a name for your project, browse to the folder where the project exists, select the project, and click 'Finish'.
*Task 1: Create a new project for the purpose of this tutorial. Once you have created the project, your project will appear in the 'Package Explorer'.*

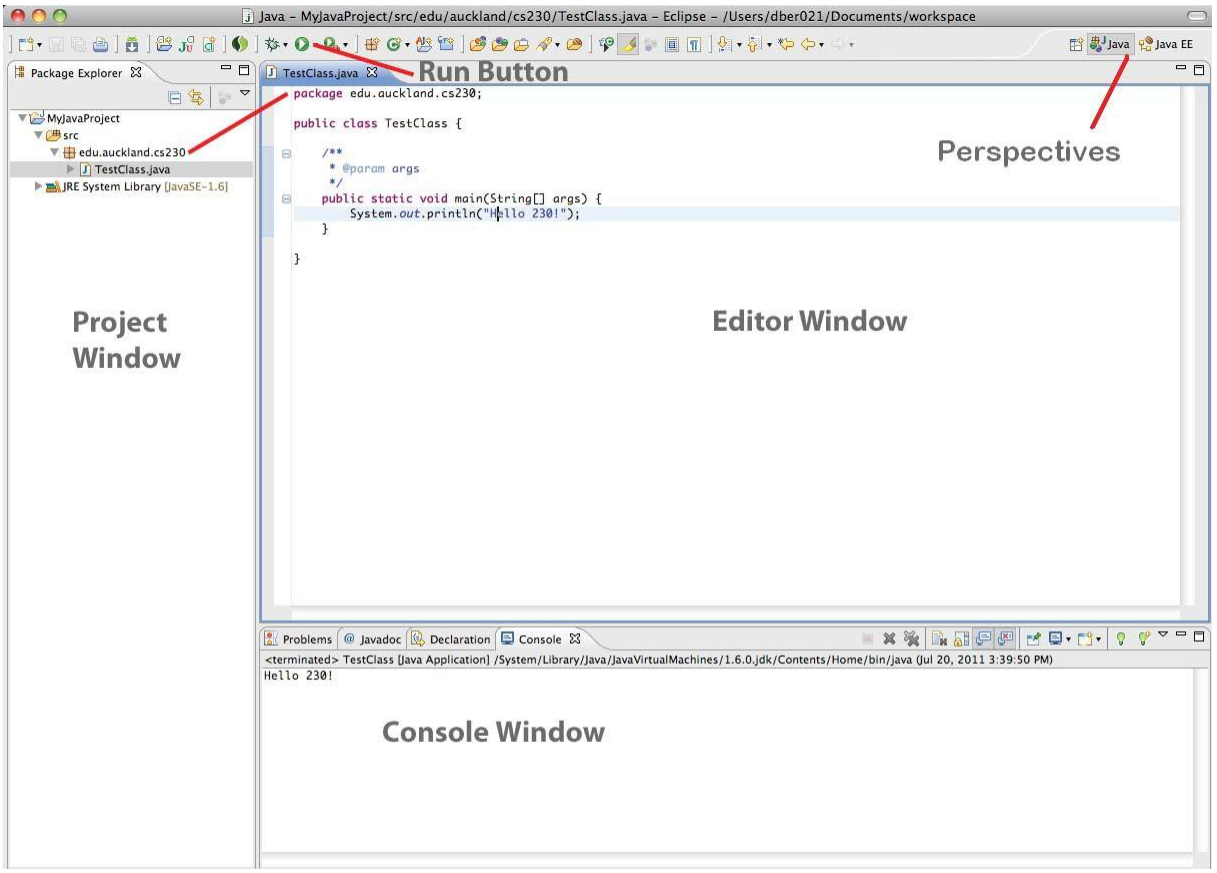


**Figure 1: Eclipse Interface**

## 5. How to add a new class?

Right click the appropriate package, choose New>New Class. Choose appropriate options for your class i.e., you could have your class inherit from java.lang.Object , you could have your new class inherit from an existing class in your package in which case you could browse the superclass as well as inherit constructors from superclass by ticking 'Constructors from superclass' option, include a main method in your class by clicking 'public static void main (String[] args)', add interfaces that your class needs to implement, etc. Once you have selected your options, click Finish.


*Task 2. Create a new class in your newly created project. Give it a meaningful name, choose to inherit from java.lang.Object, and include main method in the class. Include a line of code in your class that prints your name. Note how when you are adding the code, Eclipse looks up appropriate methods and presents you the list to choose from. Also include a line of erroneous code and notice how Eclipse will find and display the errors. When you double click a displayed error, Eclipse opens*

*the file containing the error and highlights the line of code that needs to be corrected.*

### 6. How to run your project?

You can run your project by clicking the Run button located the top left side of the Eclipse IDE toolbar. You may want to choose Run As 'Java Application'.

To **stop** a running project, click the red stop button located above the output pane. This button is enabled only when a project is running.

*Task 3. Run your project to print your name to the output console.*

### 7. What are packages?

Packages are used in Java programming to organize the code. The files are contained within the packages. Note how the class you created in 5 is contained within the **'default package'**. You can also create your own packages by clicking **New>Package** in Eclipse and then create your code files within the package. The files contained within the package have the line **'package <package_name>'** on their top. You could also import other packages and files within other packages in your Java classes.

### 7. Perspectives

Perspectives are available to show different environments within eclipse. Some examples of these perspectives or environments are 'Java' (for development of java code) or 'Debugging'. By toggling between perspectives we can see that our workbench has different tools available designed so that a task can be carried out easily. Eclipse is also dockable which allows for movement of tools and windows through the perspective. Occasionally this can go wrong and it is simple to just reset the perspective back to a familiar state by choosing from the menu bar -

**Window -> Reset Perspective**

### 8. Debugging

Eclipse has powerful debugging tools that allow you to view the contents of the application state in memory. It allows for developers to step through code one line at a time and stop at almost any point. To allow for debugging a user must place breakpoints in their code to tell the debugger where to run to and wait for the developer's next instruction. This point of execution may hold details of state that are pertinent to fixing or understanding a problem. Once a breakpoint has been entered and exists within the execution of code, the debugger can be started by clicking the Bug button that sits beside Run. Typically Eclipse will ask the user to change to the debugging perspective which is a perspective that provides the developer with tools that help find bugs and understand code. Just like the running of a normal application within Eclipse, we must either allow the execution to complete or choose to stop the application and free resources. Once either of these terminal states occurs, switch back to Java perspective to carry on developing.