Just Enough Eclipse

What is Eclipse(TM)?

Eclipse is a kind of universal tool platform that provides a feature-rich development environment. It is particularly useful for providing the developer with an Integrated Development Environment for the Java platform. Its open and extensible nature has made it one of the most popular IDE tool.

Why is it important?

It has already been recognized that an IDE is an indispensable tool for use in the development lifecycle. An IDE can definitely cut down development time and reduce development effort as it provides an integrated environment for writing codes, compiling programs and debugging. Of course the modern IDE like Eclipse provides much more than that. Features like Refactoring, integration with Application Servers, generation of documentation and Unit Testing are also very important in expediting the development lifecycle. It is thus important before the start of a project to carefully consider the use of an IDE tool.

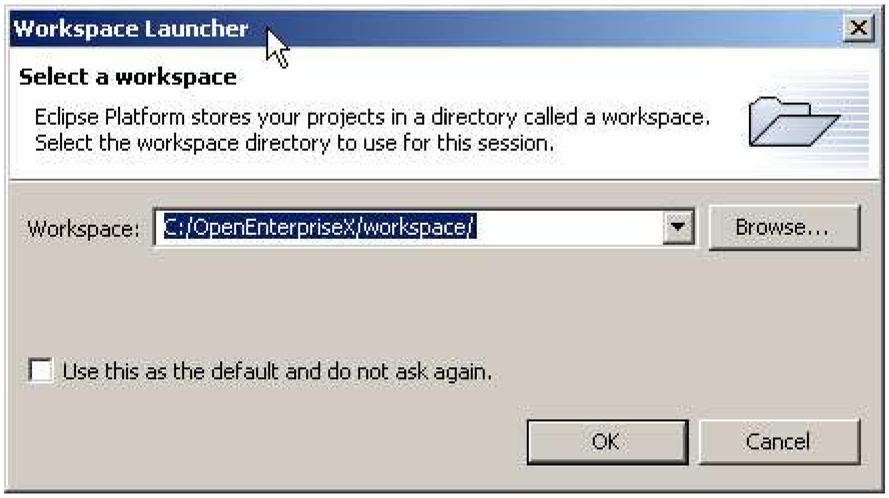
What is this tutorial about?

The main aim of this tutorial is to ensure that you have "JUST ENOUGH" knowledge of Eclipse. It does not intend to give you all the nuts and bolts of all about Eclipse. The documentation/help in Eclipse provides excellent materials in that aspect. This tutorial aims to provide enough material for you to have a basic understanding about Eclipse and be able to move on to the other tutorials with regards to OpenEnterpriseX. In the subsequent tutorials, when development with components like EJB(Enterprise Java Beans) or Servlets, Eclipse will be assumed to be the integrated development environment to be used.

This tutorial will go through the creation of a Project, a Folder and a Java Class. Some other aspects like content assist will also be introduced.

Step 1

Goto the Start Menu->OpenEnterpriseX->Eclipse
If this is the first time you launch Eclipse, you will see the following screen. The screen allows you to set the Workspace for Eclipse. A Workspace is a default directory used by Eclipse to store project files. Use the default directory provided and Click OK. If you do not like to set the Workspace everytime you start Eclipse, you can check on the 'Use this as default and do not ask again' checkbox.


Workspace Launcher

Select a workspace

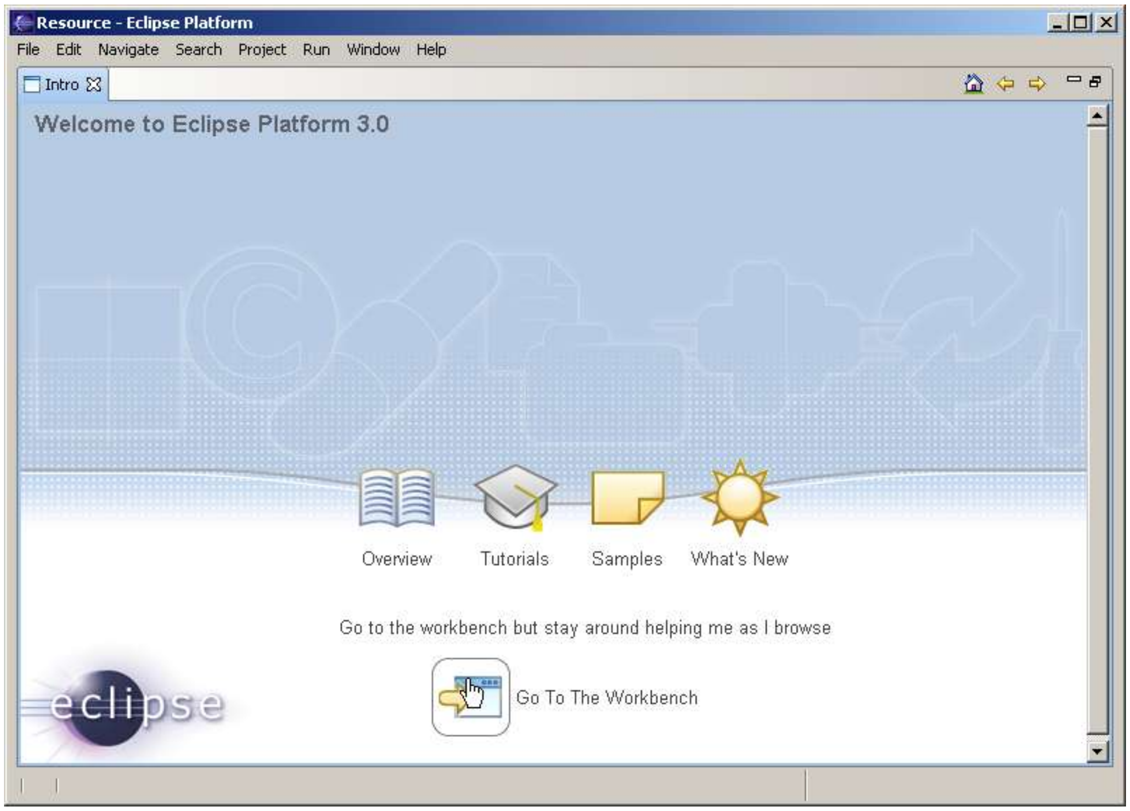
Eclipse Platform stores your projects in a directory called a workspace.
Select the workspace directory to use for this session.

Workspace: C:/OpenEnterpriseX/workspace/

Use this as the default and do not ask again.

Step 2

This is the first screen that you will see after Eclipse is launched. For those who are interested in exploring Eclipse in detail, simply explore the different sections like the Overview and Tutorials. The “Java Development User Guide” in the Help menu is especially useful. However in this tutorial we will focus on the Just Enough aspects of Eclipse.

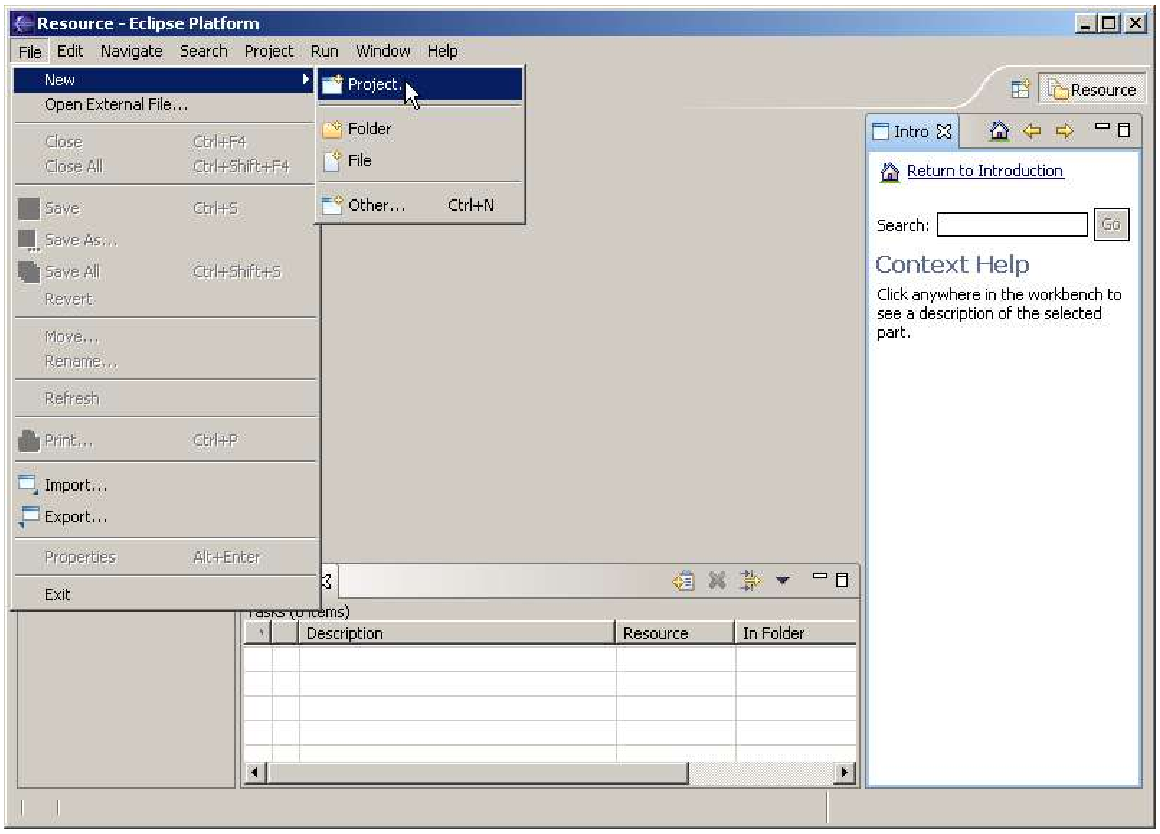

Resource - Eclipse Platform
File Edit Navigate Search Project Run Window Help
Intro
Welcome to Eclipse Platform 3.0
Overview
Tutorials
Samples
What's New
Go to the workbench but stay around helping me as I browse
Go To The Workbench
eclipse

Step 3

In this and the remaining sections, we will cover how to create a Java Class and run it.

First of all, you need to create a Project. A Project is like a logical grouping of related files with the necessary environement settings like the libraries to use. It is the basis of working with the source files.

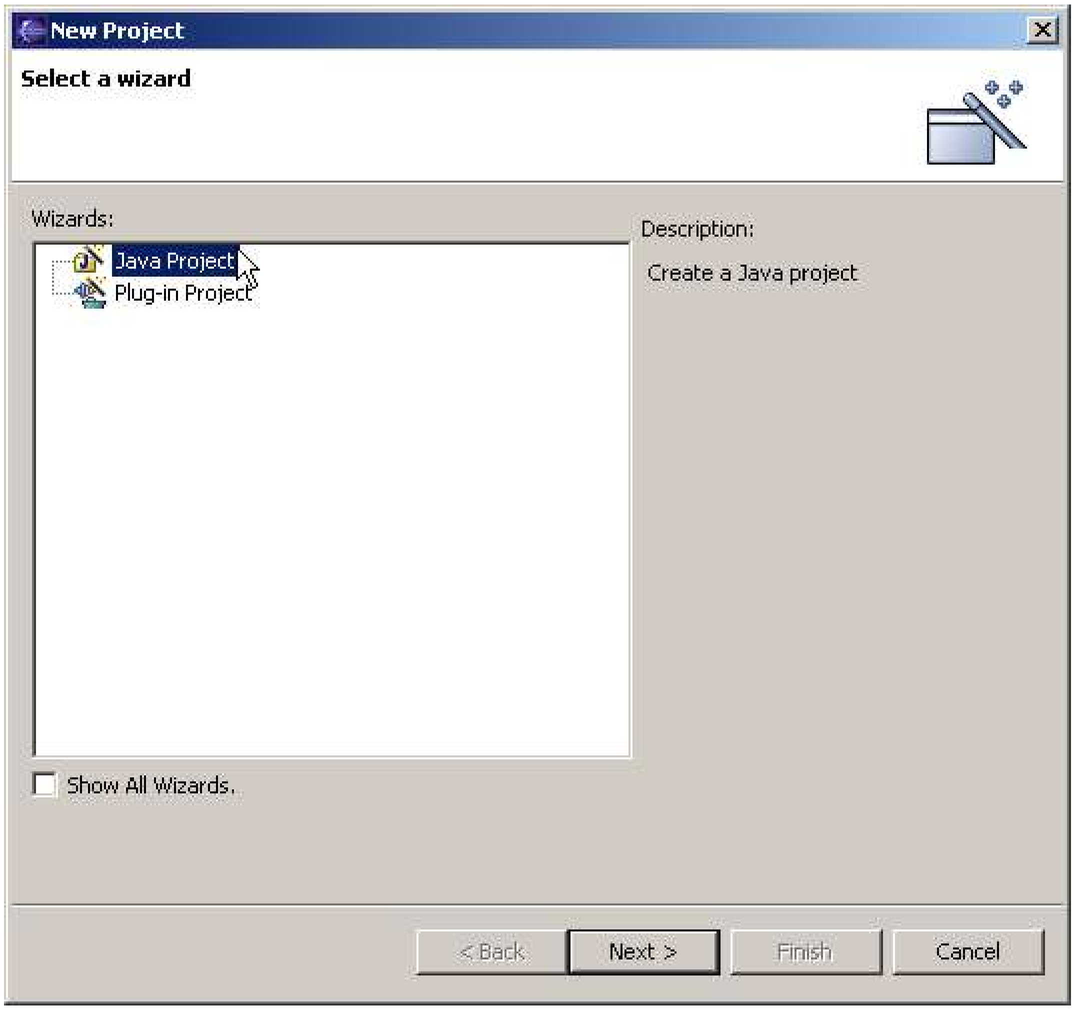
Create a new Project by going to New->Project as shown in the diagram below.



Step 4

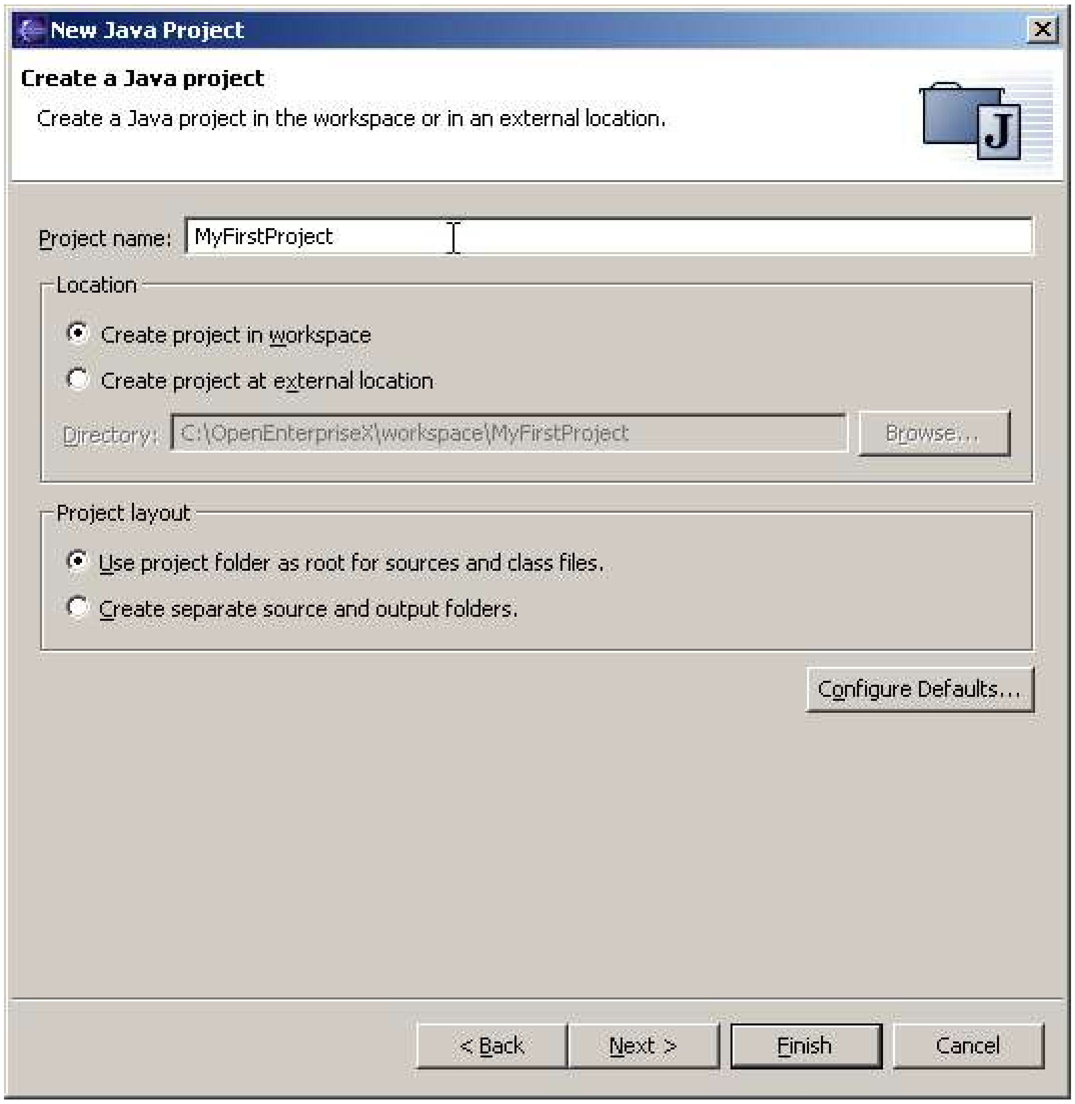
There are two types of Project, Java Project and Plug-in Project.
A Plug-in Project is only used for developing plugins for Eclipse. As mentioned previously, Eclipse is an extensible platform for the development of plugins. However, we are only interested in creating a Java Project for now.

Click on Java Project and Click Next.



Step 5

We will name our project MyFirstProject and place it in the default workspace. We will also use the project folder as root for sources and class files.


New Java Project

Create a Java project

Create a Java project in the workspace or in an external location.

Project name: MyFirstProject

Location

(●) Create project in workspace

( ) Create project at external location

Directory: C:\OpenEnterpriseX\workspace\MyFirstProject

Project layout

(●) Use project folder as root for sources and class files.

( ) Create separate source and output folders.

Step 6

The New Java Project Wizard appears. Below is a brief description of the different options and tabs in this page of the Wizard.

Source – For specificying the folder for storing source codes (.java files). Each source folder can define an exclusion filter to specify which resources inside the folder should not be visible to the compiler.

Projects – Required projects on the build path. Adding a required project will result in all its classpath entries marked as 'exported' to be added to this project.

Libraries – For specification of libraries that will be used in this project. For example, if the project is going to interact with a database then the JDBC drivers can be specified here.

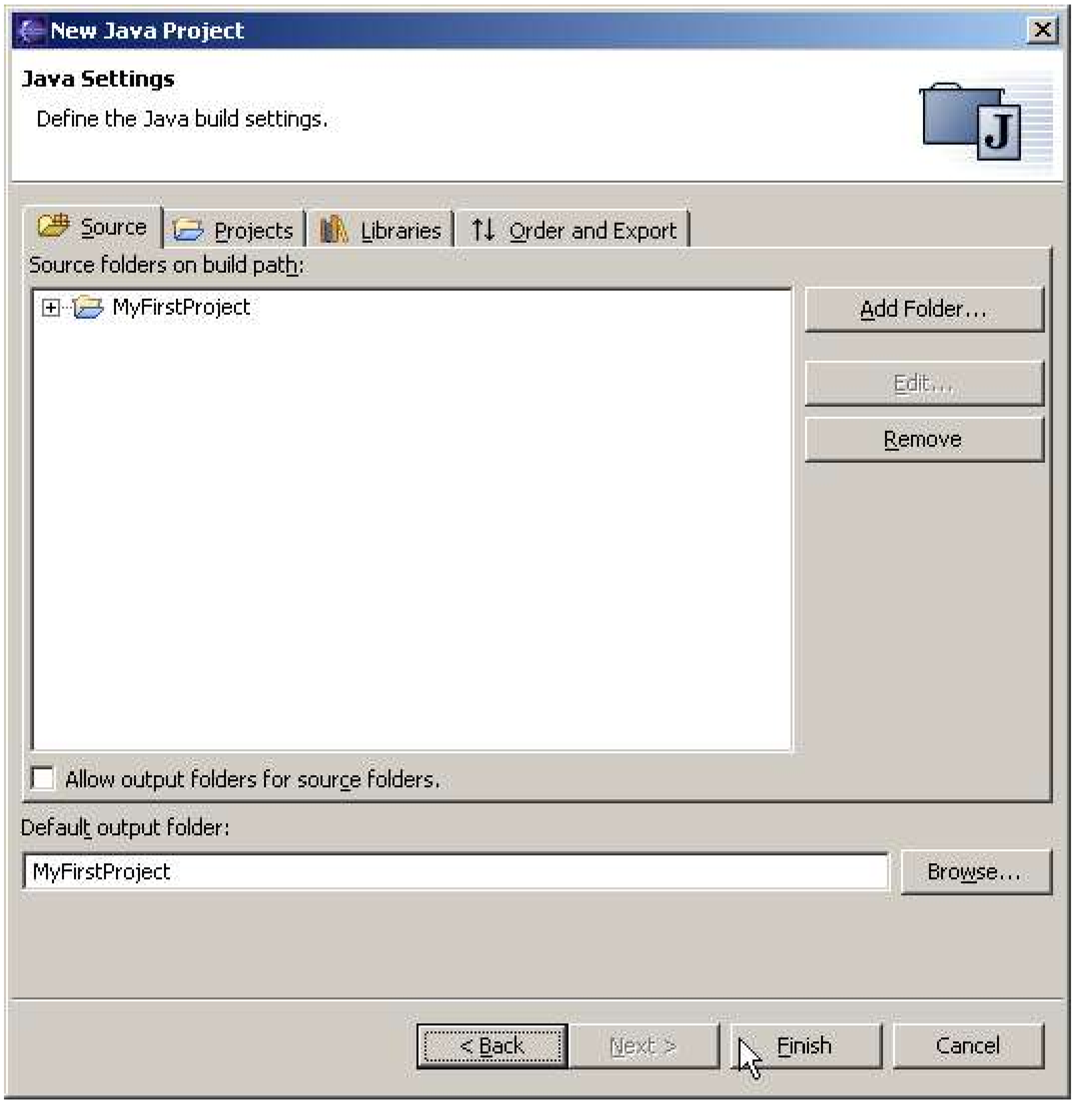
Order and Export – Build Class Path Order and Exported Entries
During compilation, the build class path is used to search(in the particular order) for classes referenced by the source code of this project. Each project will have it's own build classpath.

Exported Entries are class path that are visible to projects that require this project.

Default Output Folder - The compilation output (eg classes) will reside in this folder. Java files in the source folders will be compiled to .class files and written to the output folder. The output folder is defined per project except if a source folder specifies an own output folder.

The information above can be modified even after a project is created. Simply go to Project->Project Properties->Java Build Path to make the changes.

Leave everything as default for now.



Java Settings

Define the Java build settings.

Source folders on build path:

MyFirstProject

☐ Allow output folders for source folders.

Default output folder:

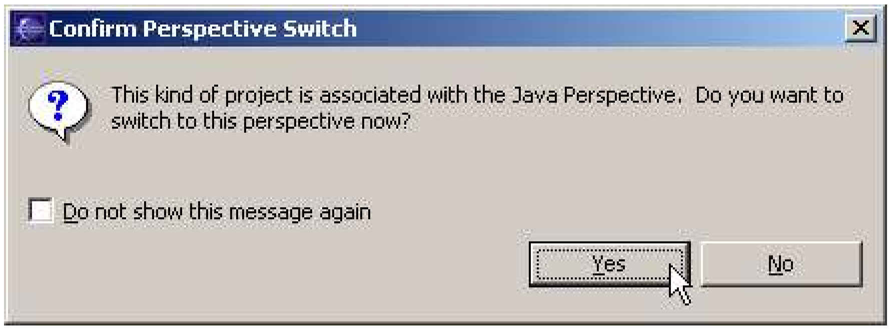
MyFirstProject

Step 7

The dialog will prompt you whether to switch to the Java perspective. As we are going to do Java development, click on the Yes button.

A perspective defines a set of views to provide a set of capabilities for the developers. For example, a Java perspective is geared towards allowing developers to edit Java source files while the Debug perspective contain views and capabilities for debugging Java programs.

A developer can also choose to define his own perspective.

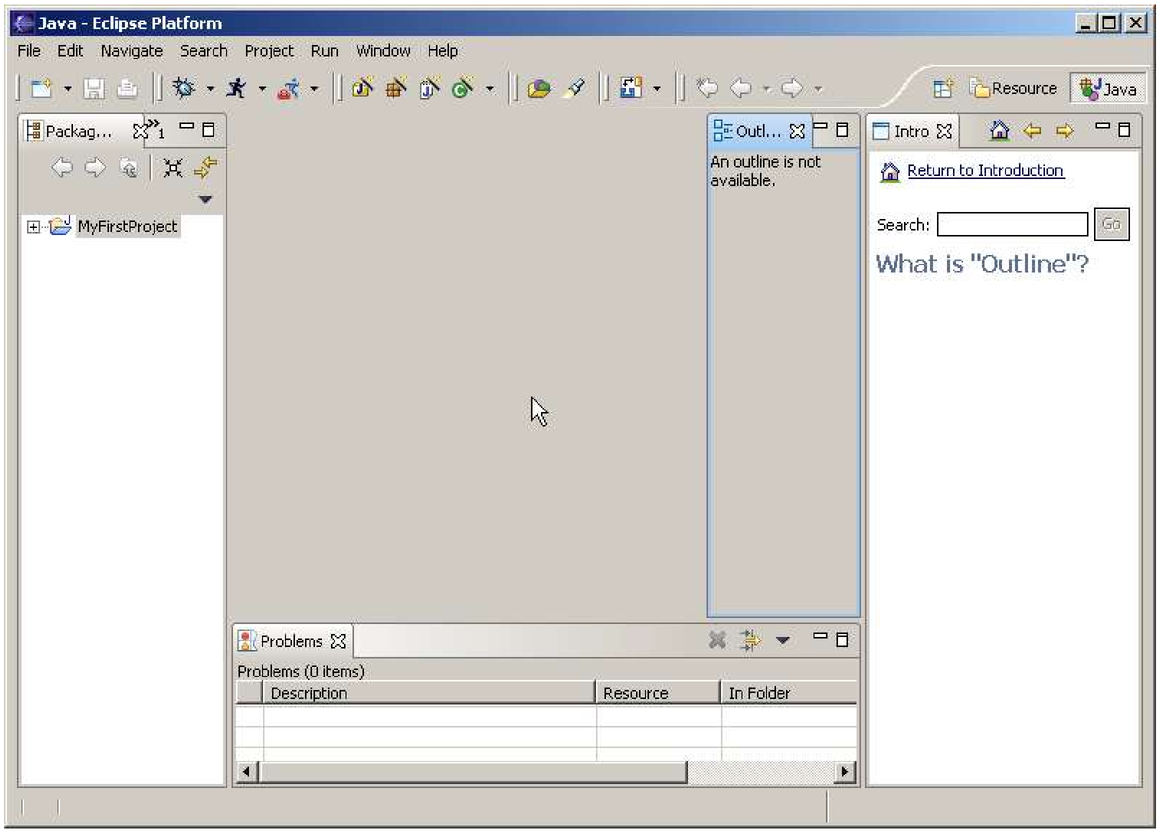


Confirm Perspective Switch

This kind of project is associated with the Java Perspective. Do you want to switch to this perspective now?

Step 8

This is what you will see after the creation of a project. On the left hand side is the Package Explorer. It shows the source folders, the referenced libraries and Java files hierarchy of the Java project. We will explain the different views in more details later.

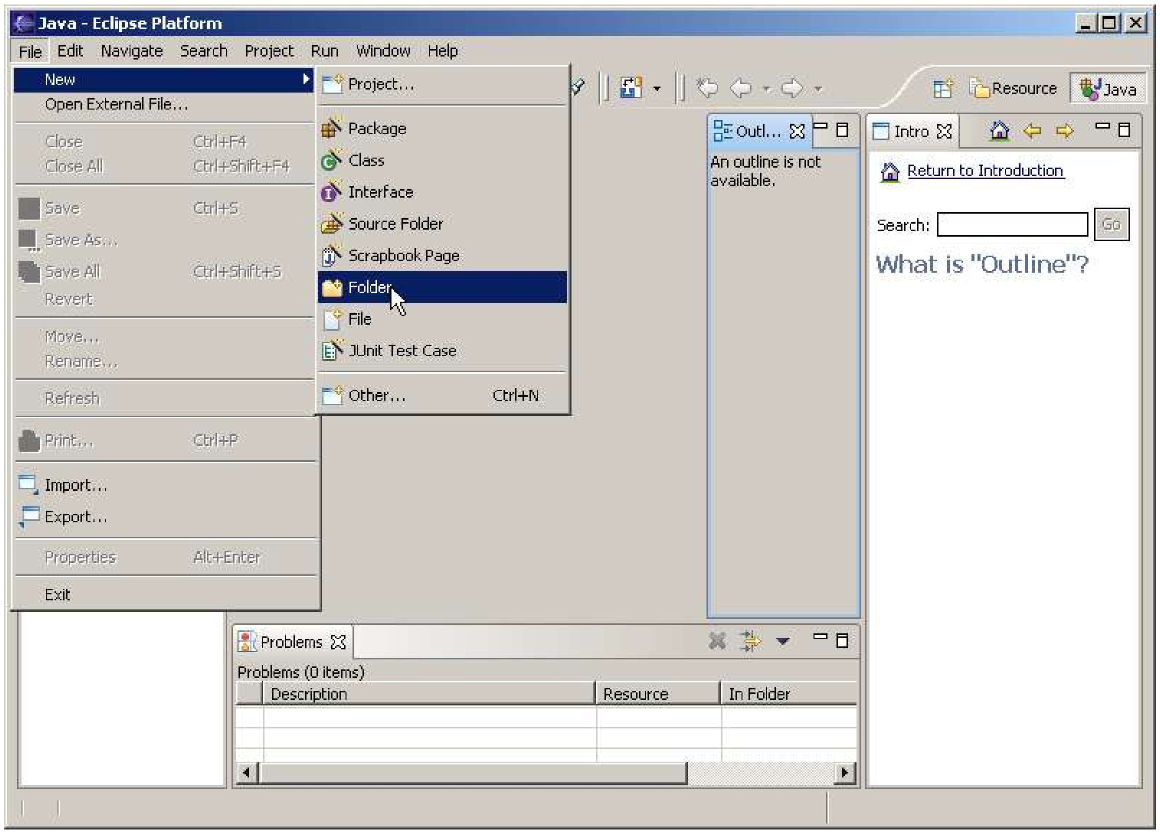


Step 9

Next, create a new Folder by going to File->New->Folder

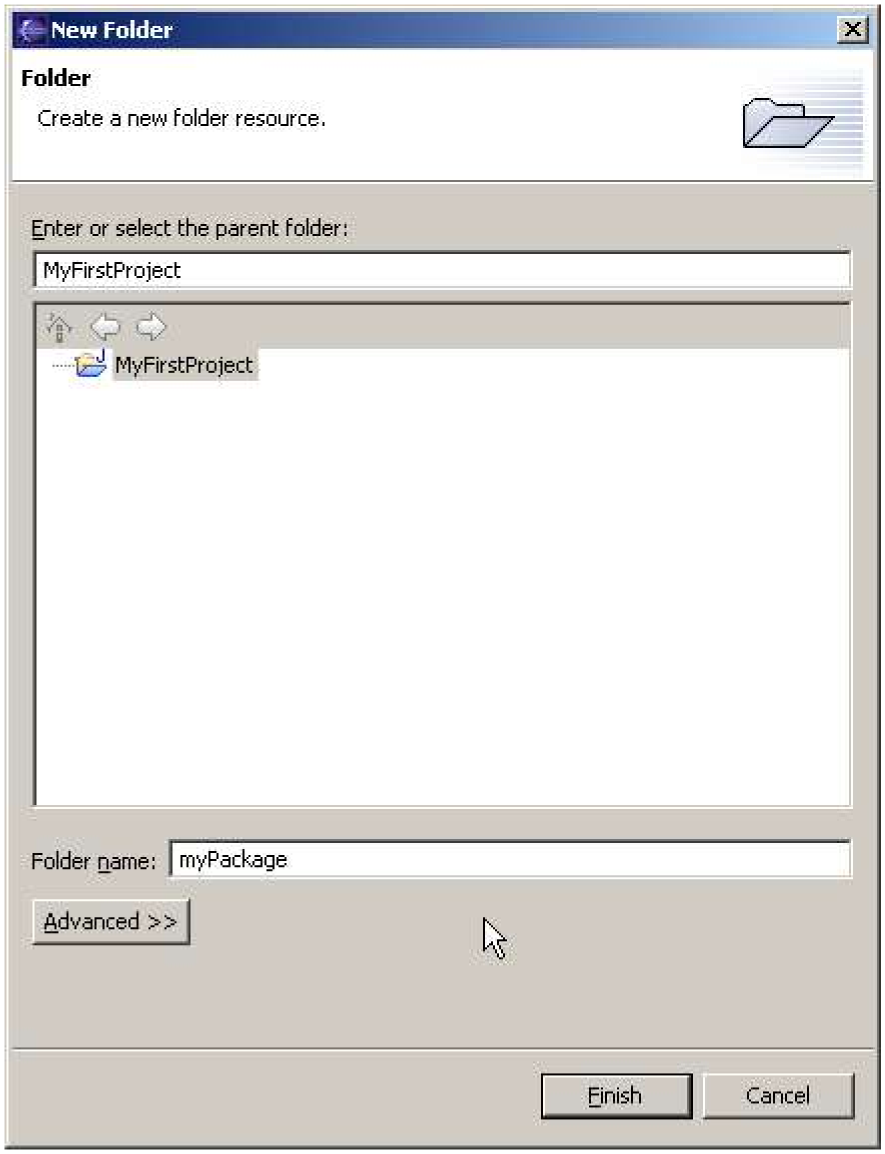
Typically you will arrange your project in a manner like com.companyname.projectname kind of folder structure. This means there will be a com Folder and inside a subfolder call companyname which includes yet another subfolder projectname.

An example is org.openEnterpriseX.openEnterpriseProject.



Step 10

First select the parent folder by clicking on 'MyFirstProject'. Add the subfolder name as myPackage.



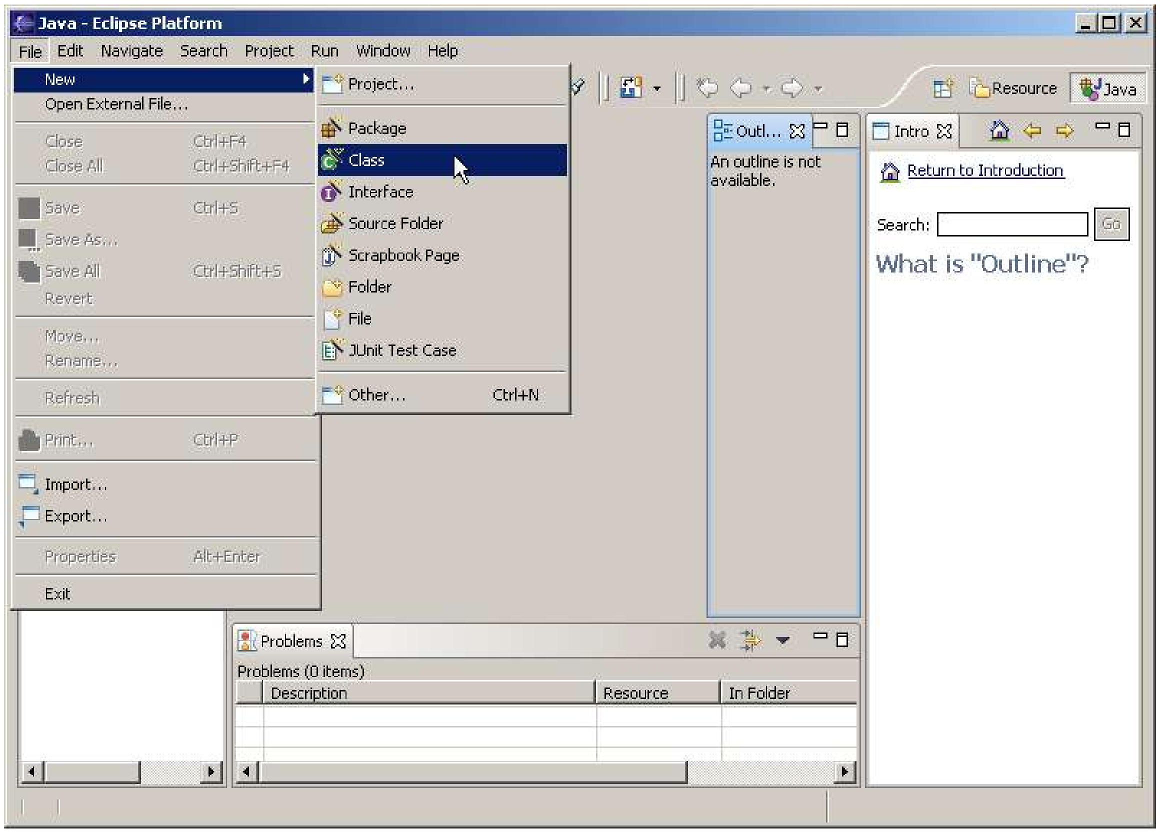
Enter or select the parent folder: MyFirstProject

MyFirstProject

Folder name: myPackage

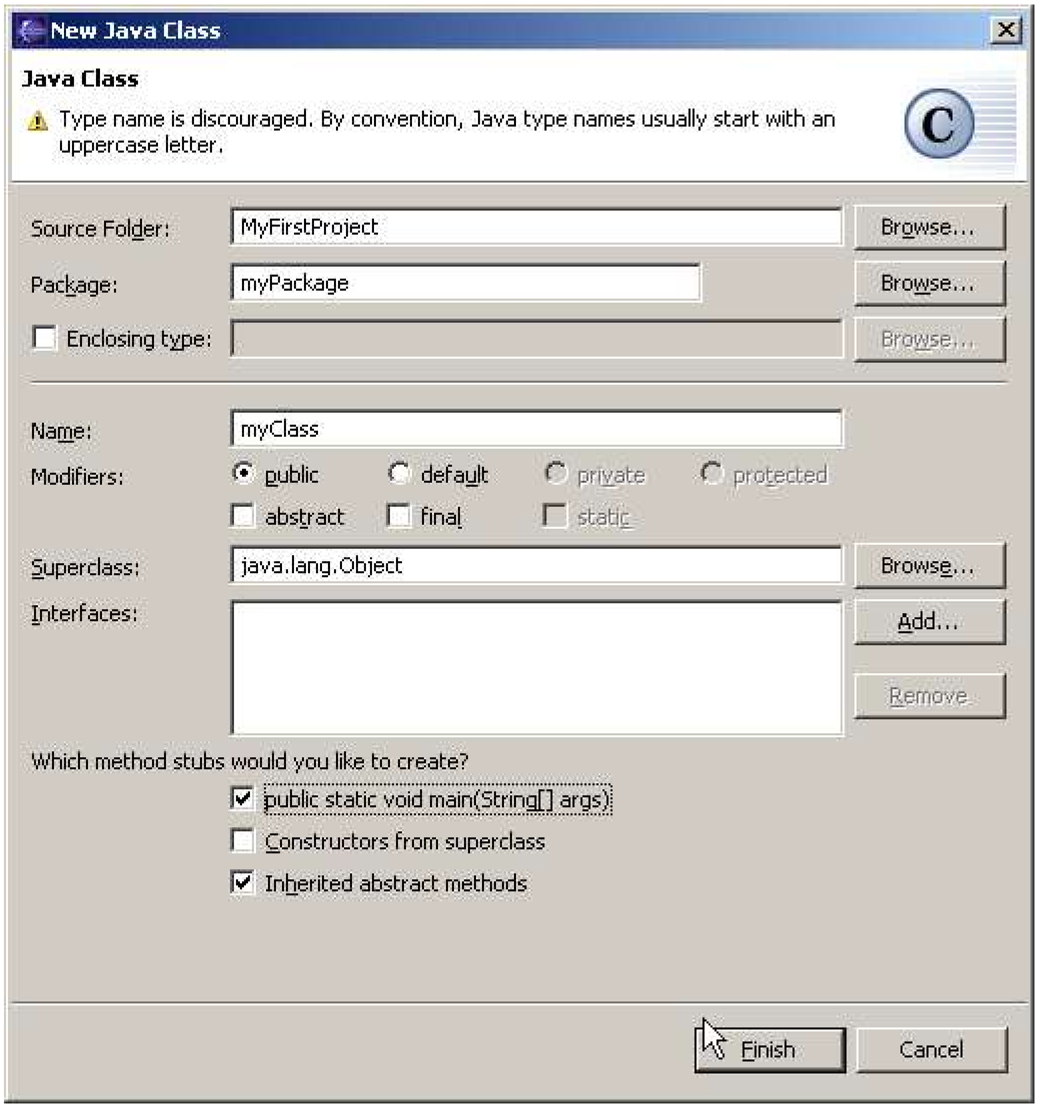
Step 11

```
The last part to create is the Java Class.
Goto File->New->Class
```



Step 12

Enter the Source Folder and Package. The Java file will be stored in MyFirstProject->myPackage. Specify the class name as myClass. Leave the modifiers of the class as public. Eclipse allows the specification of a superclass or interfaces implemented by this class in this wizard. It also allows the specification of the stubs to create. Make sure to check on 'public static void main(String argv[])'. Leave the other stubs options as they are. Codes will be generated based on what is specified here later.


New Java Class

Java Class

Type name is discouraged. By convention, Java type names usually start with an uppercase letter.

Source Folder: MyFirstProject

Package: myPackage

☐ Enclosing type:

Name: myClass

Modifiers: ● public ○ default ○ private ○ protected

☐ abstract ☐ final ☐ static

Superclass: java.lang.Object

Interfaces:

Which method stubs would you like to create?

☑ public static void main(String[] args)

☐ Constructors from superclass

☑ Inherited abstract methods

Step 13

Key in the source code in the Java editor as follows

```
public class myClass {

      public void printHelloWorld()
      {
            for (int x=0;x<5;x++)
                  System.out.println("Hello World:"+x);
      }

      public static void main(String[] args) {
            myClass mc = new myClass();
            mc.printHelloWorld();
      }
}
```

While you are keying the source code, take note of Content Assist feature. For example when you are keying in 'System.out.println', upon keying in 'System.' the editor will prompt you to choose a method name from a list of methods. Upon choosing the method, the editor will auto fill in the rest of the line of code for you. To bring up Content Assist manually, press the 'Ctrl Space' key. This is a very useful shortcut that can help you reduce development time.

Step 14

Upon creation of the Java class, the Java Perspective appears. Below is a description of the different components.

Package Explorer
The Package Explorer shows for each project, the source folders and referenced libraries.

Outline
On the right hand side is the Outline View. This view list structural elements of Java source files for example variables, method names, package declaration and import declarations.

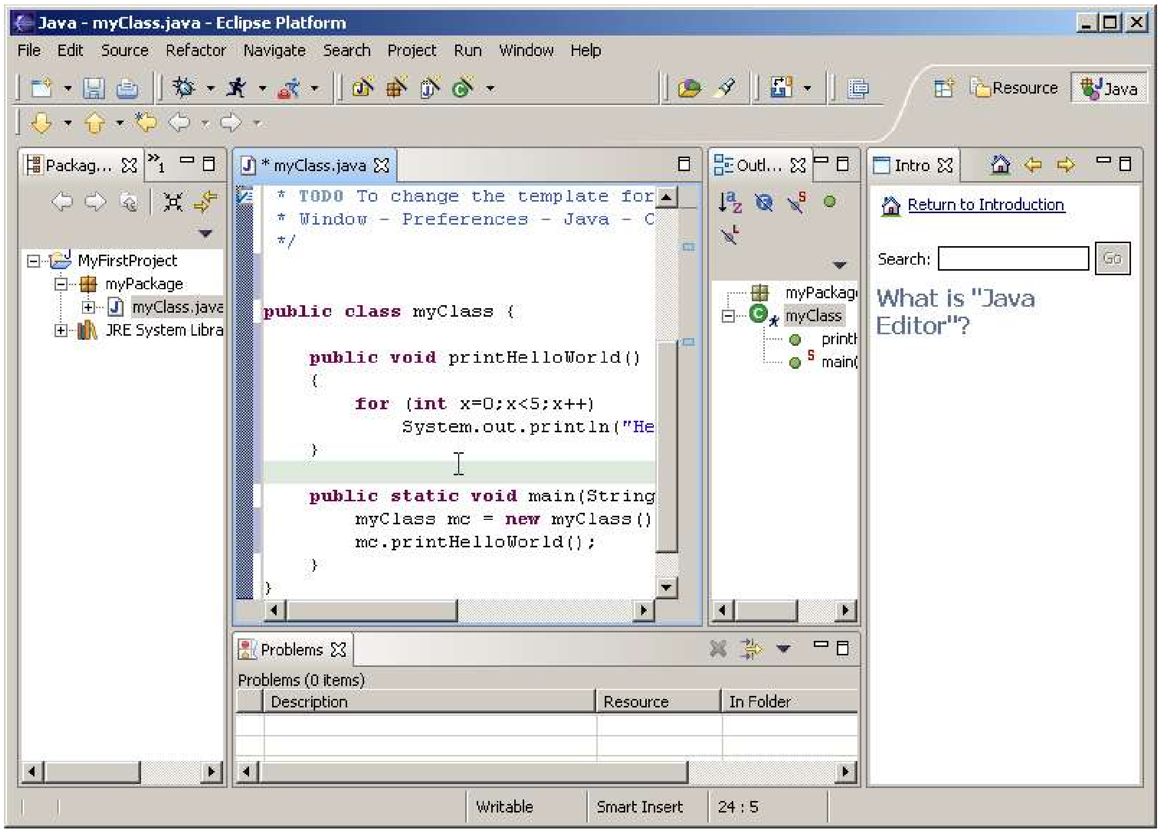
Java Editor
In the middle is the Java Editor which allows the editing of Java source files. Different elements in the Java source file are rendered in different colors.

Help
On the right hand side is also the Help View. This view lets you search and view help documentation.

Problems
The problems view shows all errors made in the source code.

On the top right hand corner, there are two buttons, one for Resource Perspective and one for Java Perspective. Eclipse has been designed to be a platform for different plugins. Each plugins might come with certain views to display specific information. A perspective allows you to customize the views associated with it. Thus when a perspective is selected, the necessary views are brought up for display. It is possible to see a list of other perspectives by going to Window->Open Perspective. To customize the perspective, goto Window->Customize Perspective.

Click on the Resource perspective and see the changes to the views.



```
 * TODO To change the template for
 * Window - Preferences - Java - C
 */


public class myClass {

    public void printHelloWorld()
    {
        for (int x=0;x<5;x++)
            System.out.println("He
    }

    public static void main(String
        myClass mc = new myClass()
        mc.printHelloWorld();
    }
}
```

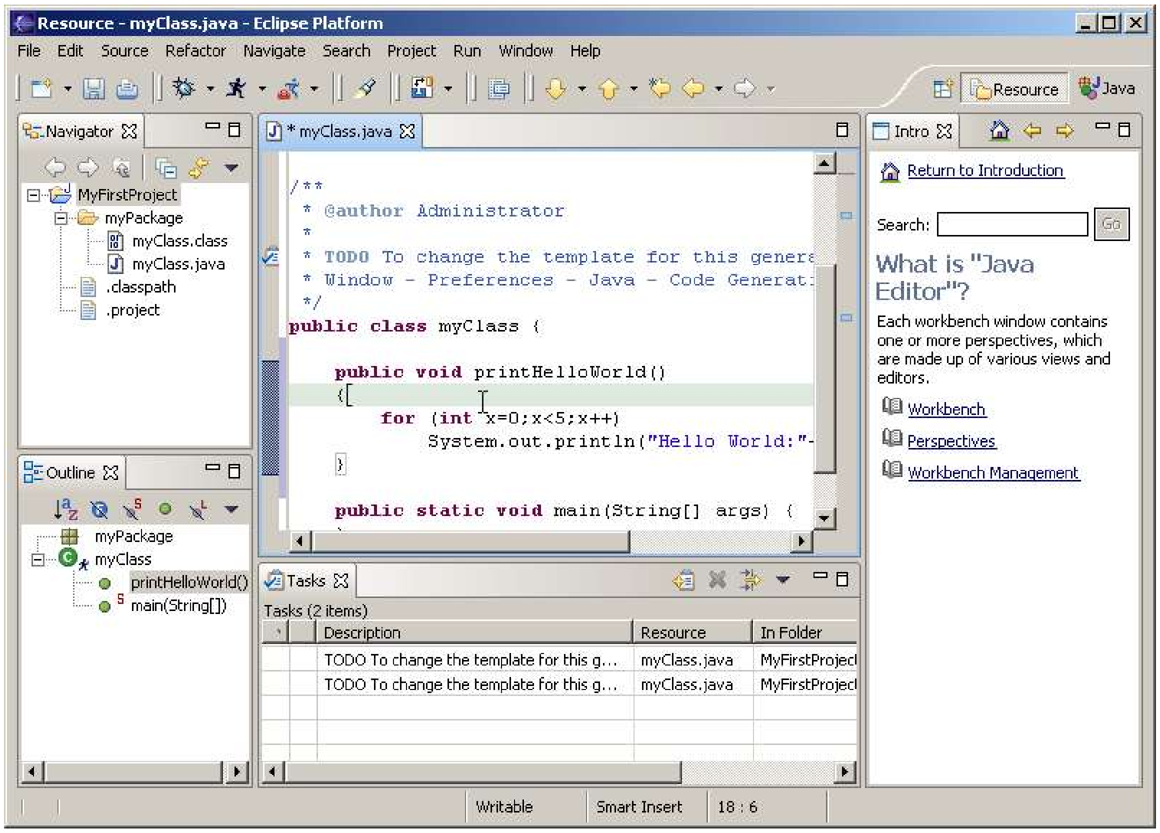
Problems (0 items)

	Description	Resource	In Folder

Writable   Smart Insert   24 : 5

Step 15

Explore the Resource Perspective and switch back to the Java Perspective before continuing.



```
/**
 * @author Administrator
 *
 * TODO To change the template for this genera
 * Window - Preferences - Java - Code Generati
 */
public class myClass {

    public void printHelloWorld()
    {
        for (int x=0;x<5;x++)
            System.out.println("Hello World:"-
    }

    public static void main(String[] args) {
```

What is "Java Editor"?

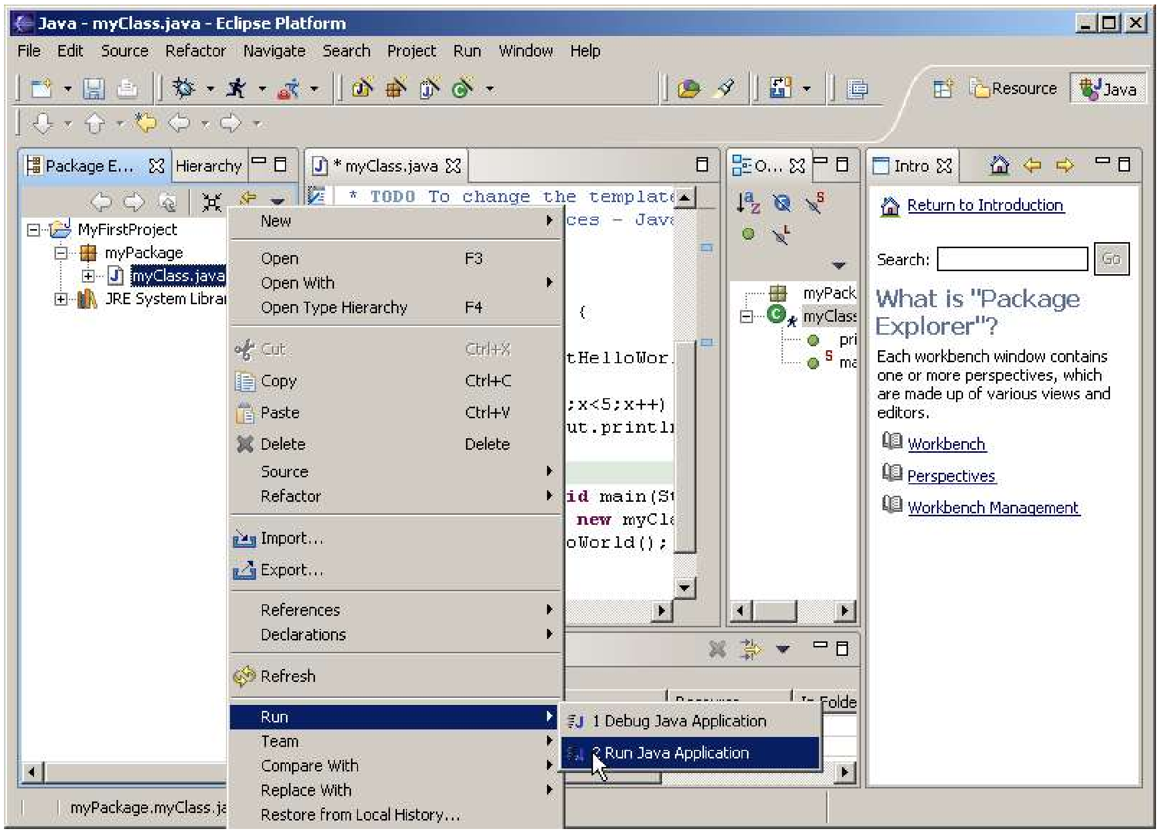
Each workbench window contains one or more perspectives, which are made up of various views and editors.

Tasks (2 items)

	Description	Resource	In Folder
	TODO To change the template for this g…	myClass.java	MyFirstProject
	TODO To change the template for this g…	myClass.java	MyFirstProject

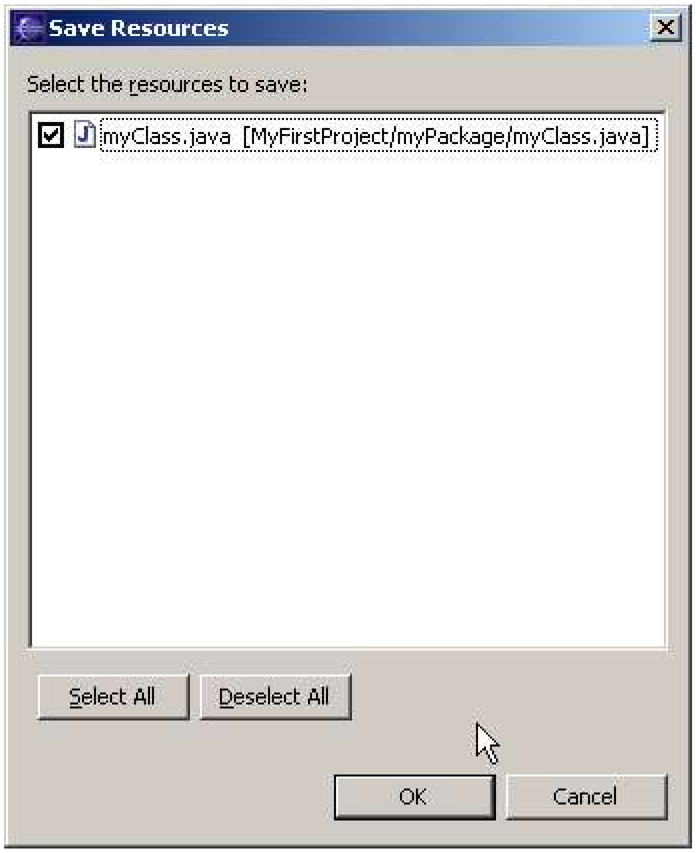
Step 16

After keying in the source code, make sure there are no errors in the Prolems View. If everything is correct, Right click on myClass.java->Run->Run Java Application (in the Package Explorer) to run the Java program we have written. If there are problems in the Problems View, click on the problem and check the syntax of the source code entered. If there are any errors, correct the errors before proceeding with the above.



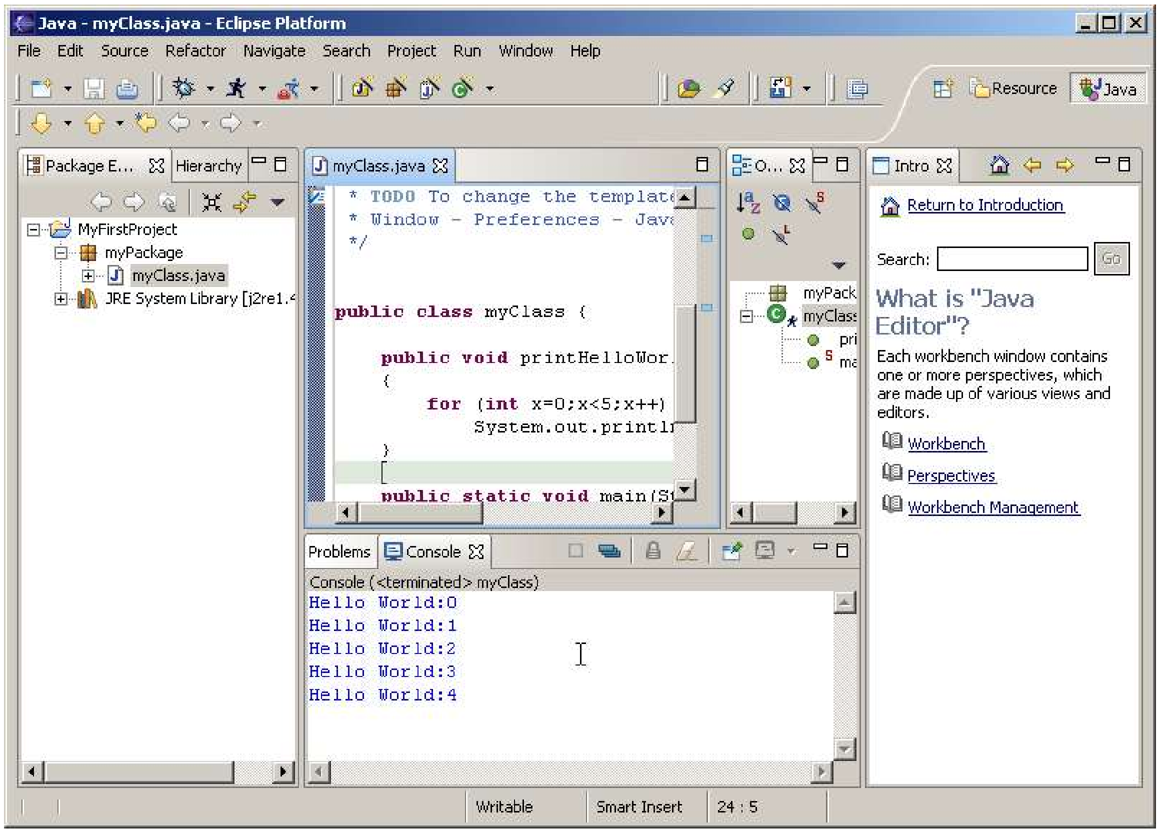
Step 17

If you have not saved your files, Eclipse will prompt you to save them. You will notice that you did not perform any manual compilation. The reason is Eclipse depends on an incremental compiler. At the time when you are keying your code, it already starts compiling the code. Any errors will be flagged out immediately in the Problems View.


Save Resources
Select the resources to save:
☑ myClass.java [MyFirstProject/myPackage/myClass.java]
Select All
Deselect All
OK
Cancel

Step 18

The results are shown in the console.



```
 * TODO To change the template
 * Window - Preferences - Java
 */


public class myClass {

    public void printHelloWor
    {
        for (int x=0;x<5;x++)
            System.out.printl
    }
    [
    public static void main(St
```

Each workbench window contains one or more perspectives, which are made up of various views and editors.

Console (<terminated> myClass)

```
Hello World:0
Hello World:1
Hello World:2
Hello World:3
Hello World:4
```

You have completed the first tutorial in creating a runnable Java Class. At the same time you have equipped yourself with enough knowledge about Eclipse to move on to the subsequent tutorials.