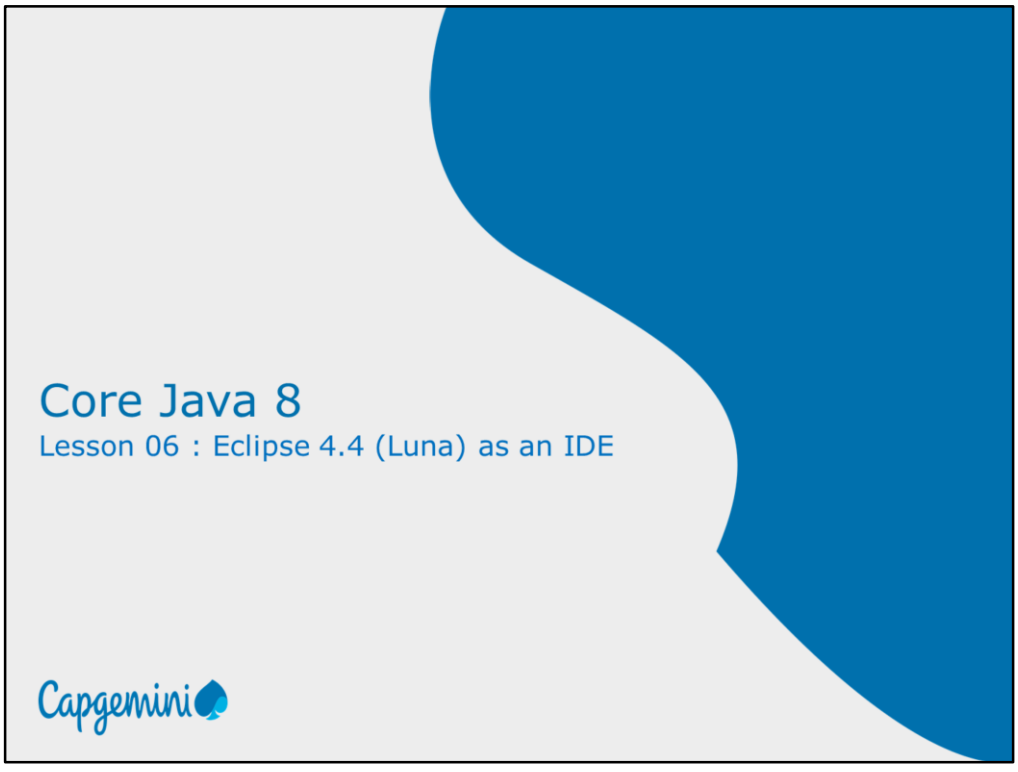

Core Java 8
Lesson 06 : Eclipse 4.4 (Luna) as an IDE
Capgemini

Lesson Objectives

After completing this lesson, participants will be able to:

- Understand fundamentals of working with Eclipse
- Creating and Managing Java Projects through Eclipse IDE
- Use different features of Eclipse to develop rapid applications

This lesson demonstrate the use of IDE to create Java applications with ease.

Lesson Outline:

Lesson 2: Eclipse4.4 as an IDE

- 2.1: Installation and setting up Eclipse
- 2.2: Introduction to Eclipse IDE
- 2.3: To create and manage Java projects
  - 2.3.1: Debugging your Java Program
- 2.4: Miscellaneous options
  - 2.4.1: Creating Jar files
  - 2.4.2: Verifying JRE installation
  - 2.4.3: Creating a Jar file
  - 2.4.4: Setting Classpath
  - 2.4.5: Passing Command line arguments
  - 2.4.6: Import and Export Options
  - 2.4.7: Automatic Build/Manual Build options
  - 2.4.8: Using Javadocs
  - 2.4.9: Tips and Tricks

6.1: Installation and Setting up Eclipse

Installing Eclipse 4.4 (Luna)

You need to follow the given steps to install Eclipse 4.4:

- Download Eclipse-SDK zip file from https://eclipse.org/downloads/
- Unpack the Eclipse SDK into the target directory
  - For example: c:\eclipse4.4
- To start Eclipse, go to the eclipse subdirectory of the folder in which you extracted the zip file
  (for example: c:\eclipse4.4\eclipse) and run eclipse.exe

6.2 : Introduction to Eclipse IDE

Integrated Development Environment

IDE is an application or set of tools that allows a programmer to write, compile, edit, and in some cases test and debug within an integrated, interactive environment

IDE combines:

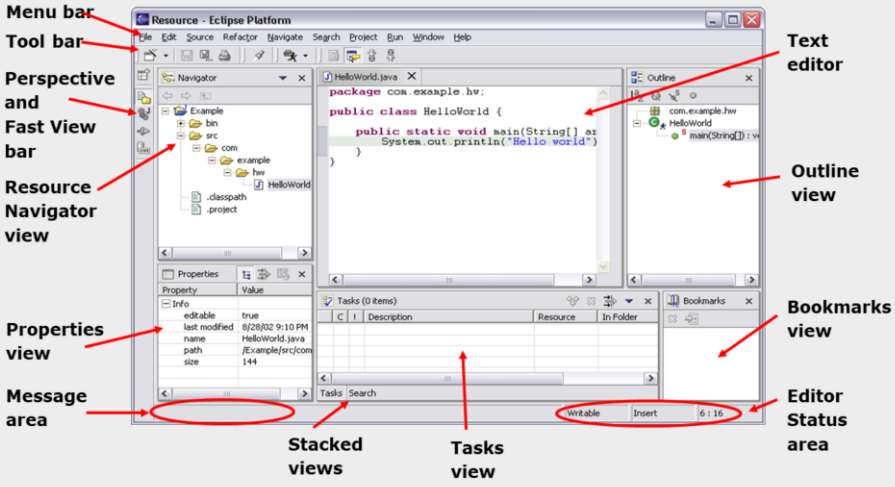
- Editor
- Compiler
- Runtime environment
- debugger

What is an IDE?

- The **Eclipse Project** is an open source software development project dedicated to providing a robust, full-featured, commercial-quality, industry platform for the development of highly integrated tools and rich client applications. Eclipse runs on Windows, Linux, Mac OSX, Solaris, AIX and HP-UX. Eclipse is actually a generic application platform with a sophisticated plug in architecture - the Java IDE is just one set of plugins. There is an active community of third party Eclipse plugin developers, both open source and commercial. Our objective is to code Java programs faster with Eclipse 4.4 as an IDE.
- Eclipse4.4 features include the following:
  - Creation and maintenance of the Java project
  - Developing Packages
  - Debugging a java program with variety of tools available
  - Running a Java program
- Developing the Java program will be easier as Eclipse editor provides the following:
  - Syntax highlighting
  - Content/code assist
  - Code formatting
  - Import assistance
  - Quick fix

6.2 : Introduction to Eclipse IDE

Workbench Terminology

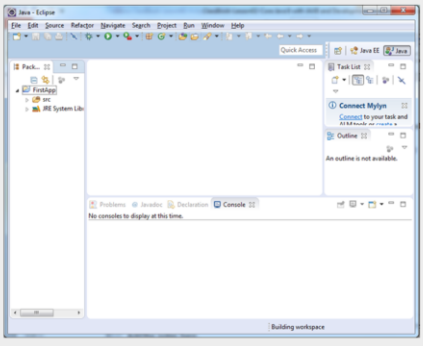


6.2 : Introduction to Eclipse IDE

The Workbench

The term "Workbench" refers to the desktop development environment
It allows you to select the Workspace
A Workbench consists of the following:

- perspectives
- views
- editors

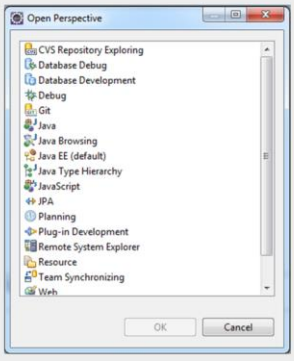


6.6 : Introduction to Eclipse IDE

The Workbench

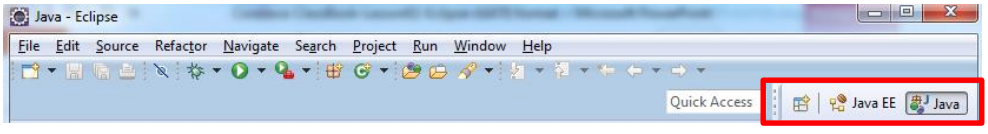
Perspective:

- A perspective defines the initial set and layout of views in the Workbench window
  - Workbench offers one or more Perspectives
  - A perspective contains editors and views, such as the Navigator
  - By default the **Java perspective** is selected
  - The title bar indicates which perspective is open

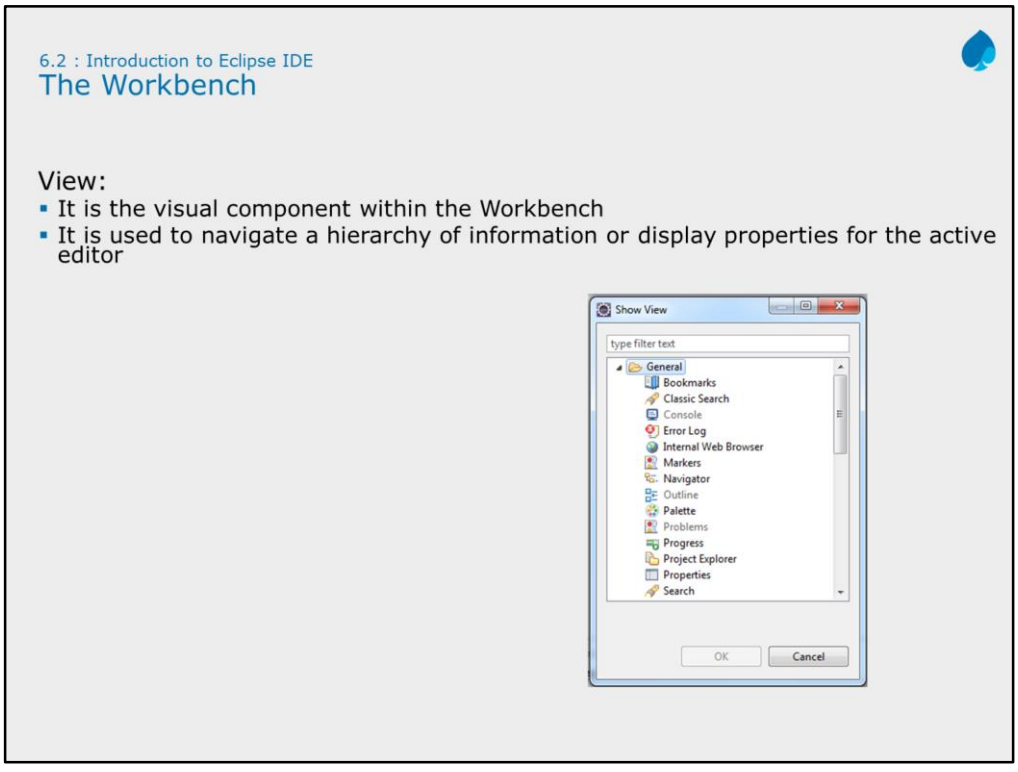


The Workbench:
The term **Workbench** refers to the desktop development environment.

Perspectives:
Each Workbench window contains one or more perspectives. Perspectives contain **views** and **editors** and control what appears in certain **menus** and **tool bars**. They define visible **action sets**, which you can change to customize a perspective. You can save a perspective that you build in this manner, making your own custom perspective that you can open again later. By default the Java perspective is selected.



Each perspective provides a set of functionality aimed at accomplishing a specific type of task or works with specific types of resources.
For example: The **Java perspective** combines views that you would commonly use while editing Java source files, while the **Debug perspective** contains the views that you would use while debugging Java programs. As you work in the Workbench, you will probably switch perspectives frequently.



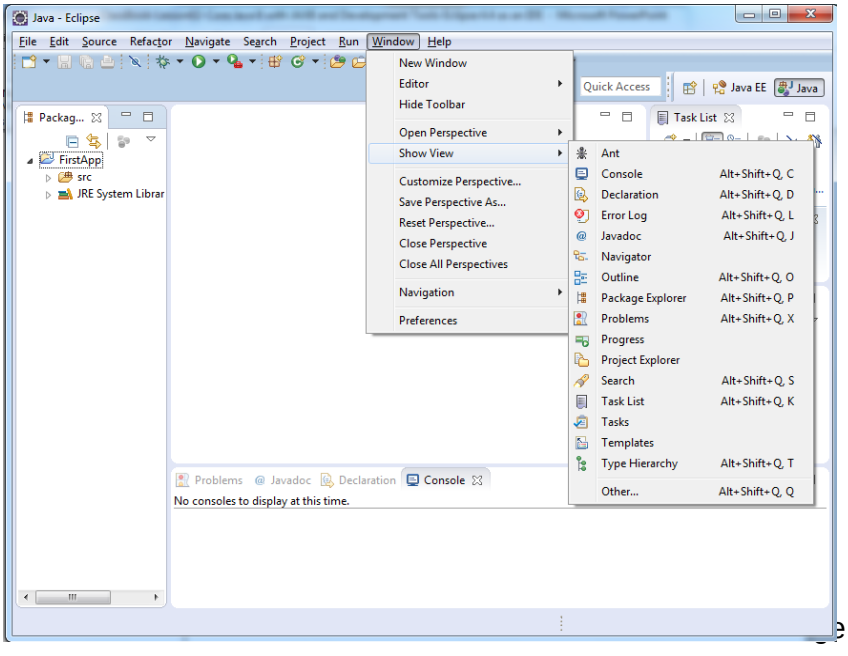
The Workbench:

View:

Views support editors and provide alternative presentations as well as ways to navigate the information in your Workbench.

For example: The Project Explorer and other navigation views display projects and other resources that you are working with.

Perspectives offer pre-defined combinations of views and editors. To open a view that is not included in the current perspective, select **Window → Show View** from the main menu bar.

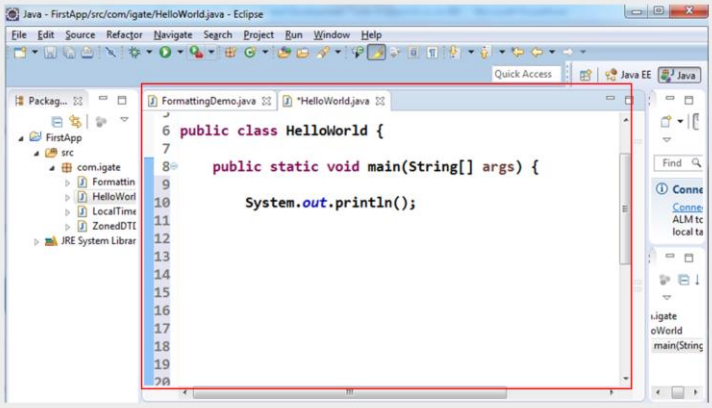


6.2 : Introduction to Eclipse IDE

The Workbench

Editor:

- It is the visual component within the Workbench
- It is used to edit or browse a resource



The Workbench:

Editor: Most perspectives in the Workbench comprise an **editor area** and one or more **views**. You can associate different editors with different types of files.
For example: When you open a file for editing by double-clicking it in one of the navigation views, the associated editor opens in the Workbench.

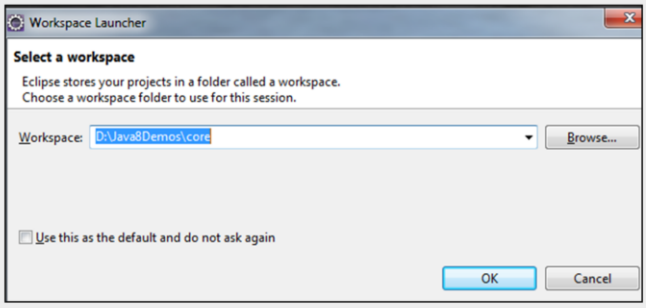
If there is no associated editor for a resource, then the Workbench attempts to launch an external editor outside the Workbench.
For example: Suppose you have a .doc file in the Workbench, and Microsoft Word is registered as the editor for .doc files in your operating system. Then opening the file will launch Word as an OLE document within the Workbench editor area.

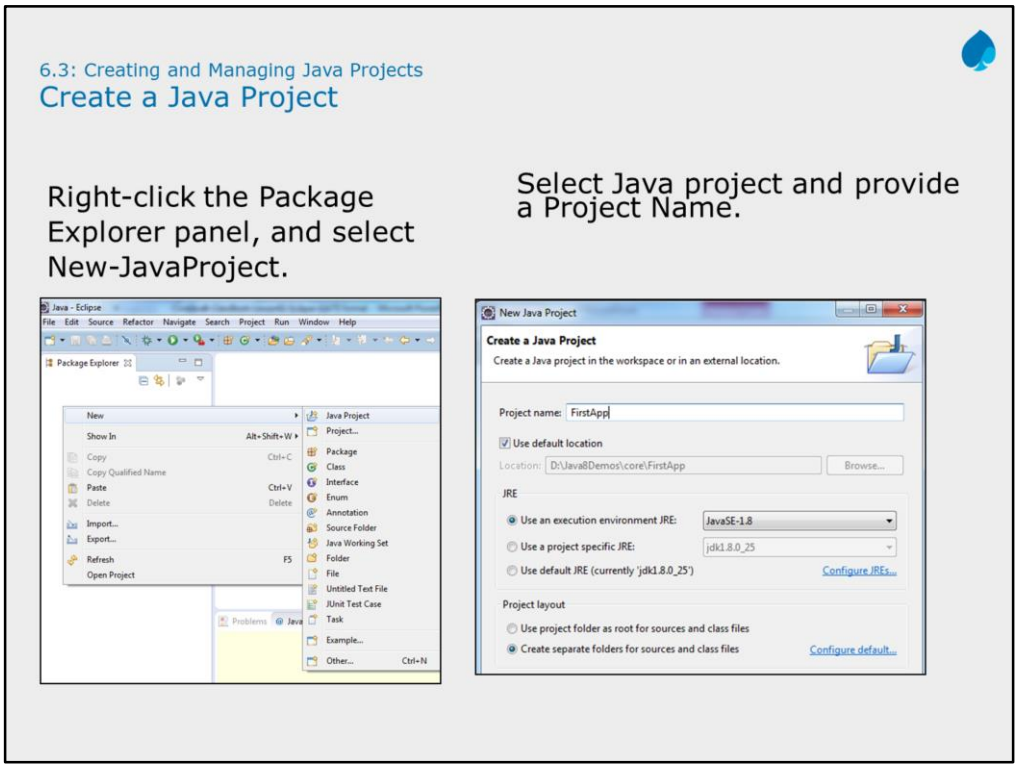
6.3: Creating and Managing Java Projects

Create Workspace

You need to follow the given steps to create a workspace:

- Start up Eclipse
- Supply a path to a new folder which will serve as your workspace
- The workspace is a folder which Eclipse uses to store your source code





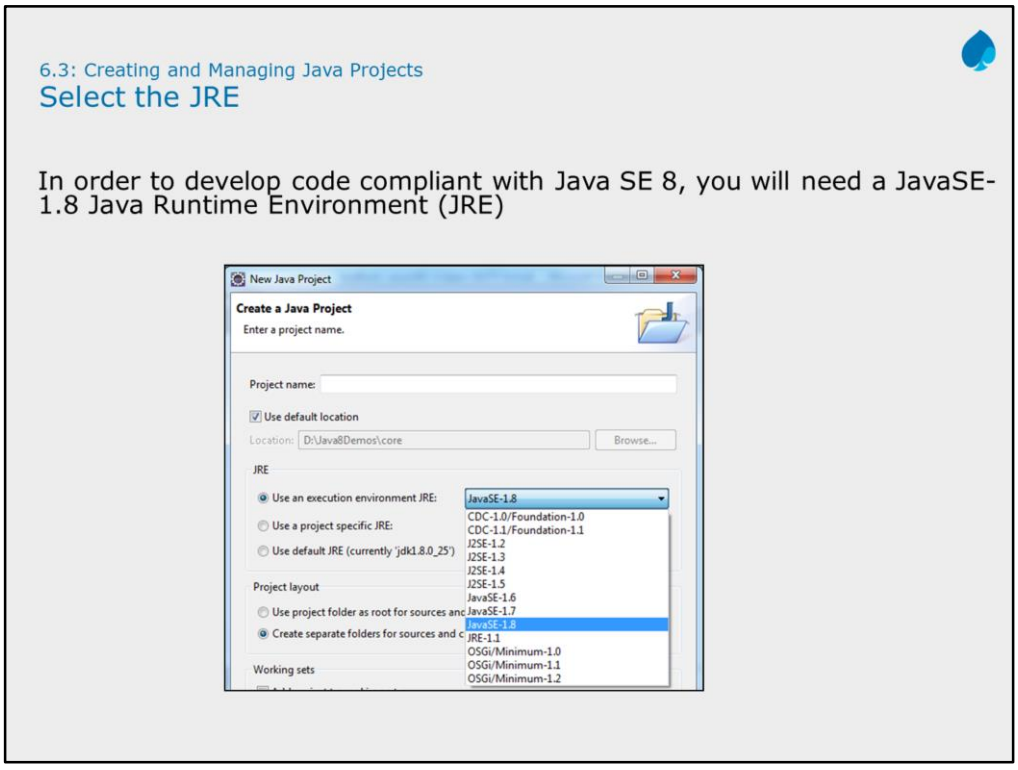
<u>Creating and Managing Java Projects</u>:

Create a Java Project:

- When Eclipse starts, you will see the Welcome page. Close the Welcome page.
- Right-click in the Package Explorer panel, and select **New → JavaProject** and provide a Project name.

Note:

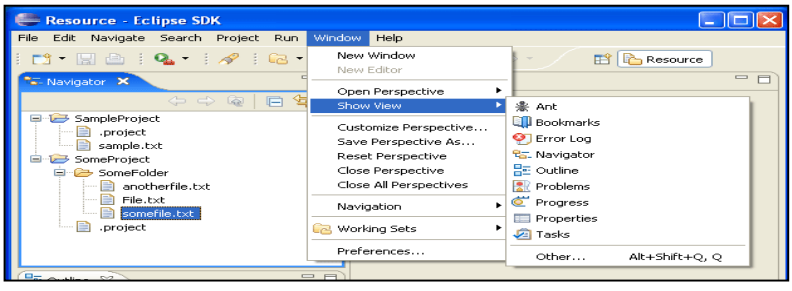
- The name of the project is FirstApp. It is created in the workspace which you have selected in the beginning.
- The Project uses jre1.8.0_25. The same folder will be used for storing both the source files and the class files.
- The Java project can now include the following:
  - Class
  - Package
  - Interface
  - Source folder
  - Folder
  - File
  - Junit Test Case
  - Other - which may include other resources as text files



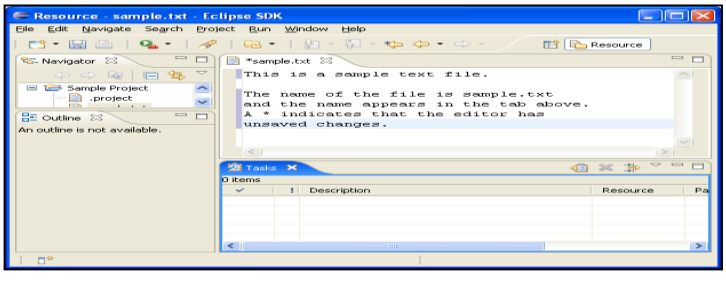
Creating and Managing Java Projects:

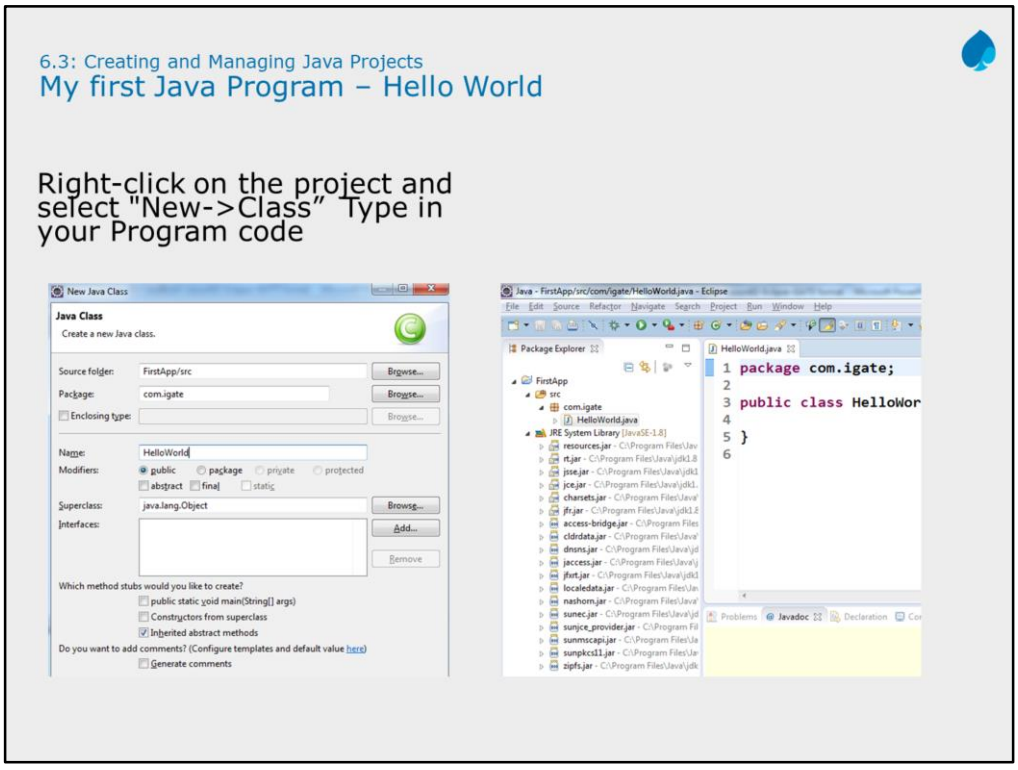
Select the JRE:

- To use the new **Java SE 1.8** features, you must be working on a project that has a **1.8 compliance level** enabled and has a **1.8 JRE**. New projects will automatically get 1.8-compliance while choosing a 1.8 JRE on the first page of the New Java Project wizard.



- Depending on the type of file that is being edited, the appropriate editor is displayed in the editor area.
  For example: If a .TXT file is being edited, a text editor is displayed in the editor area.





<u>Creating and Managing Java Projects</u>:

My first Java Program – Hello World:

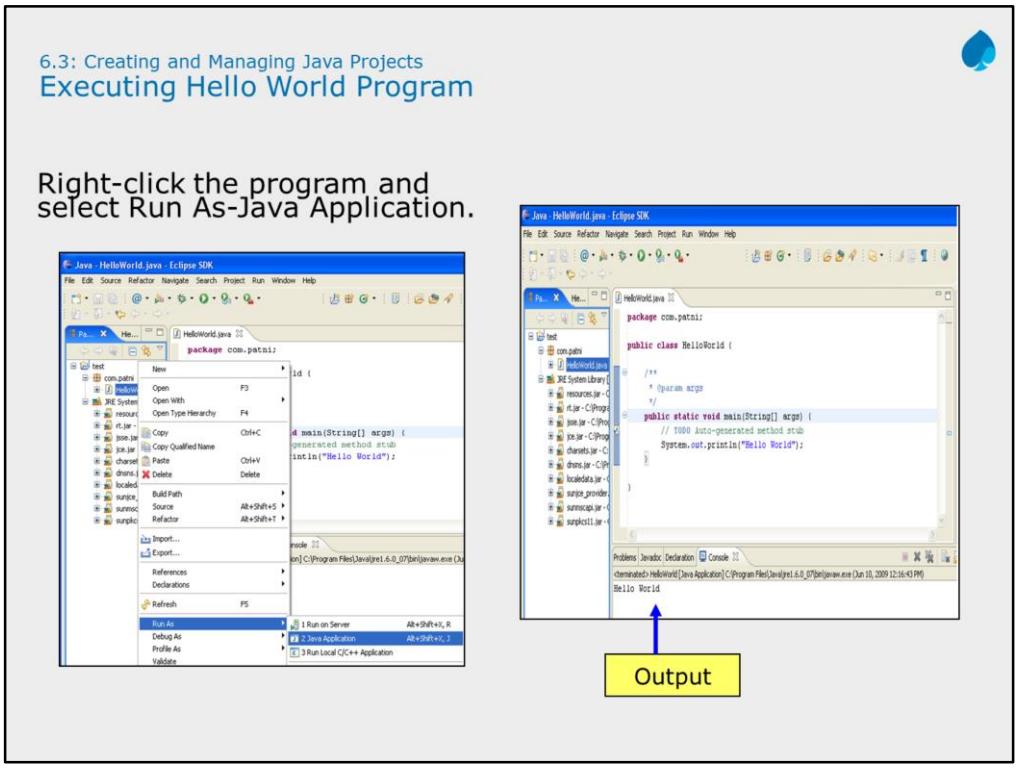
- If you want to create some new Java code, right-click the project and select "**New → Class**".
- In the dialog box created, give a name for the Java program in the **Name** textbox.
- Also notice that a package name is also given, as the usage of default package is discouraged. A package in Java is a group of classes which are often closely or logically related in some way. (The **Package chapter** is discussed later in the course).
- Eclipse will generate skeleton of the class including **default** constructor.

Note:

- The package name is com.igate.
- The Java program's name is HelloWorld.
- The HelloWorld class will have the public access modifier.
- The HelloWorld class inherits from java.lang.Object.

Developing the Java program will be easier as Eclipse editor will provide:

- Syntax highlighting
- Content/code assist
- Code formatting
- Import assistance
- Quick fix



<u>Creating and Managing Java Projects</u>:
Executing Hello World Program:

- Right-click the **HelloWorld.java** in the Package Explorer, and select **Run As → Java Application**.
- The program after execution produces the output in the **Console view**.
- Running class from the Package Explorer as a Java Application uses the default settings for launching the selected class, and does not allow you to specify any arguments.

6.3: Creating and Managing Java Projects

Demo

HelloWorld Program using Eclipse IDE

6.3: Creating and Managing Java Projects

Debugging your Java Program using Eclipse

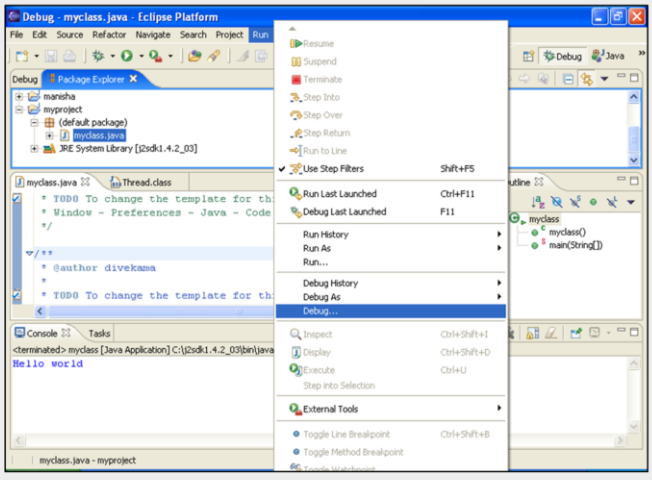
The Java Development Toolkit (JDT) includes a debugger that enables you to detect and diagnose errors in your programs running either locally or remotely

The debugger allows you to control the execution of your program by employing the following:

- setting breakpoints, suspending launched programs, stepping through your code, and examining the contents of variables

6.3: Creating and Managing Java Projects

Debugging your Java Program using Eclipse



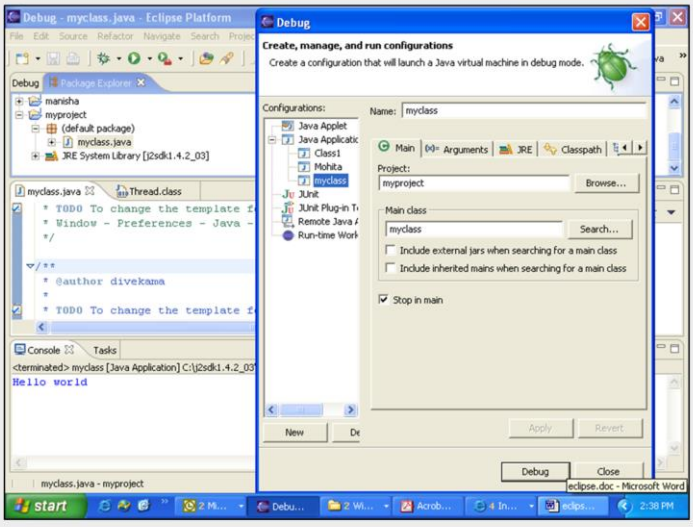
<u>Creating and Managing Java Projects</u>:

Debugging a Java Program:

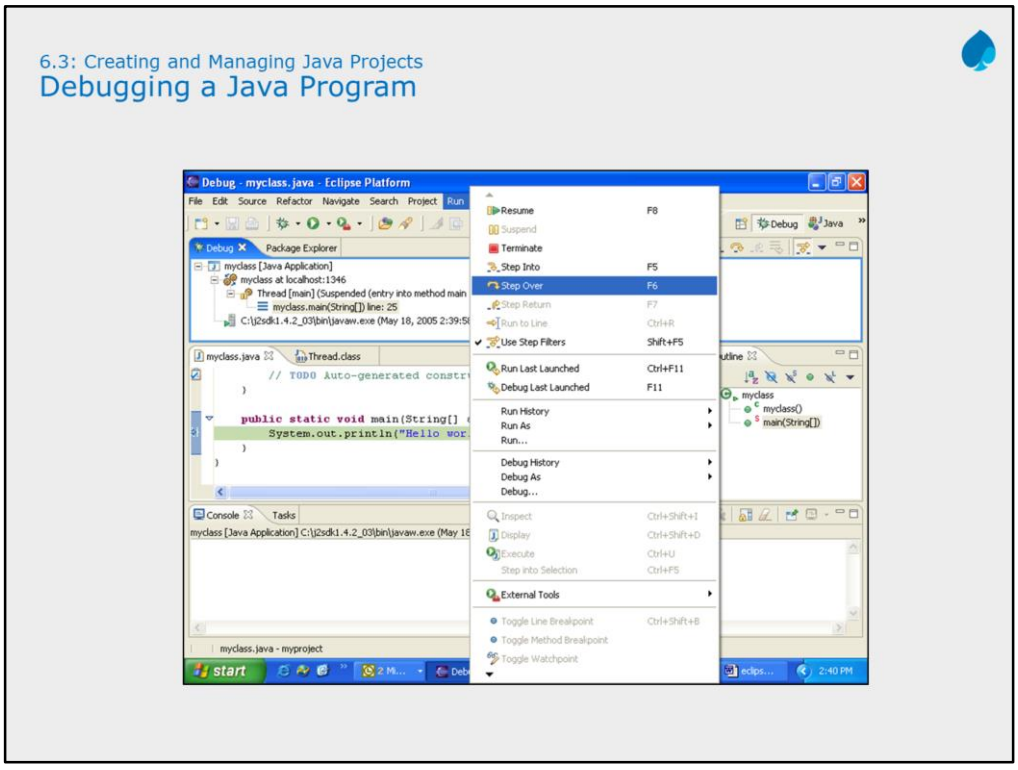
- Eclipse gives you **auto-build** facility, where recompilation of the necessary Java classes is done automatically.
- To debug your Java program, select the Java source file which needs to be debugged, select **Run → Debug**. This will ask you to select an option to halt in public static void main() method, from where you may select to step into each and every function you come across or step over every function and only capture output of each function.

6.3: Creating and Managing Java Projects

Specifying Debugging options



Note: Click **Debug** to start debugging process.



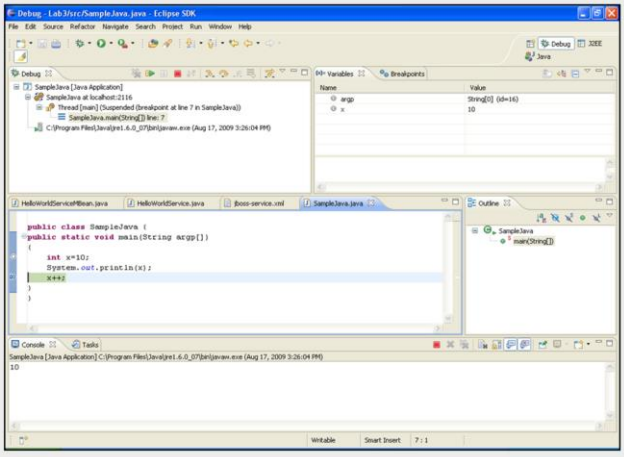
<u>Creating and Managing Java Projects</u>:

Debugging a Java Program:

- Debugging can be attained by stepping-into or stepping-over the statements.
  - **Step-into** will traverse through each and every statement in a function.
  - **Step-over** will generate output after the function call is over.
- Tracing and watching the variable values is available as different debug views.

6.3: Creating and Managing Java Projects

Debugging a Java Program



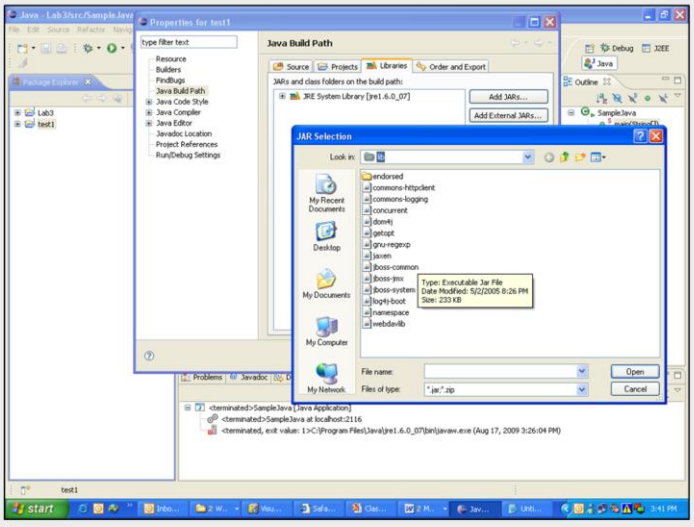
Creating and Managing Java Projects:

Debugging a Java Program:

- You may launch your Java programs from the workbench. The programs may be launched in either **run** or **debug** mode.
  - In **run** mode, the program executes. However, the execution may not be suspended or examined.
  - In **debug** mode, execution may be suspended and resumed, variables may be inspected, and expressions may be evaluated.
- Variables view gives contents of the program variables at different statements in execution.
- Breakpoints can be set for debugging, by opening the **marker-bar** pop-menu and selecting **Toggle Breakpoint**. While the Breakpoint is enabled, the thread execution suspends before the execution of the line happens. **Breakpointing** is a technique to set-up the starting point for program debugging.

6.4: Miscellaneous Options

Adding external jar file



Miscellaneous Options:

Adding an external jar file:

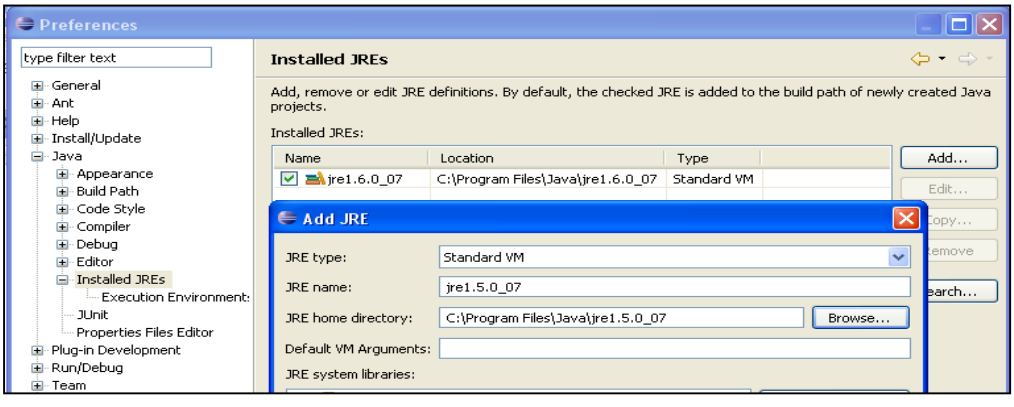
- When you are developing advanced Java programs, you might have to include some external jar files.
  - Click **Project Properties**.
  - Select **Java build Path**.
  - Click the **Libraries** tab, and click the **Add External Jar Files** button.
  - Locate the folder which contains the jar files, and click **Open**.

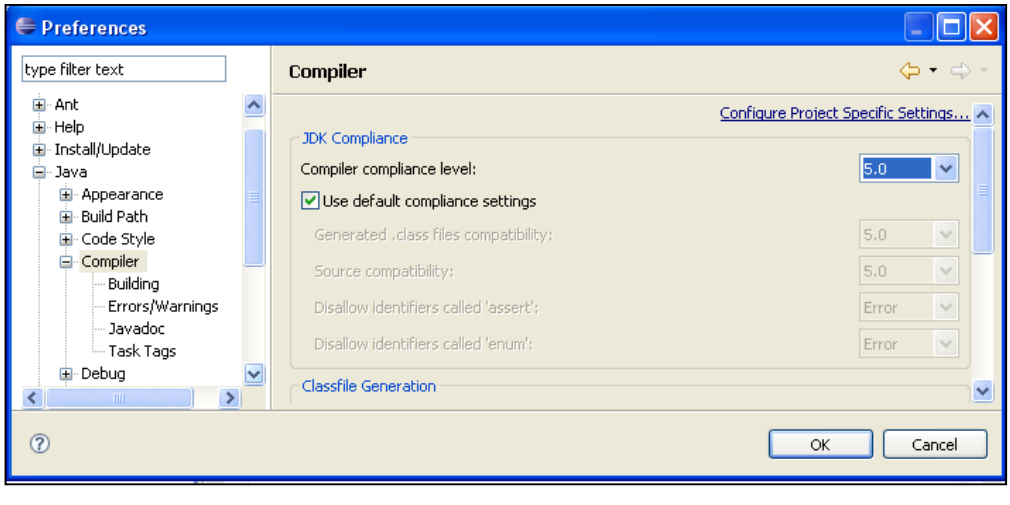
6.4: Miscellaneous Options

Verifying / Changing JRE Installation

Changing the JRE is a common need while working with Eclipse which can be achieved as follows:

- Select the menu item Window → Preferences to open the workbench preferences
- Select Java → Installed JREs in tree pane on the left, to display the Installed Java Runtime Environments preference page
  - To add a new JRE, click the Add button, and select the new JRE home directory
- Change the appropriate compiler.
  - Select Java → Compiler and select the appropriate compiler



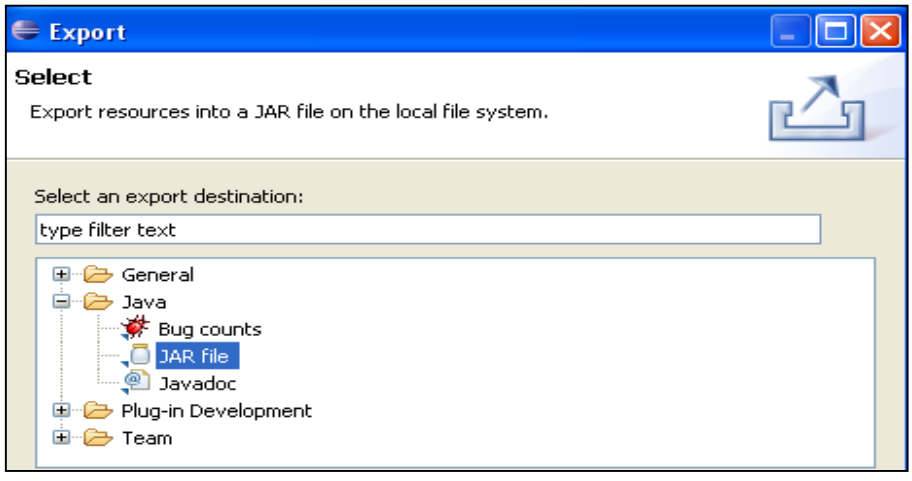


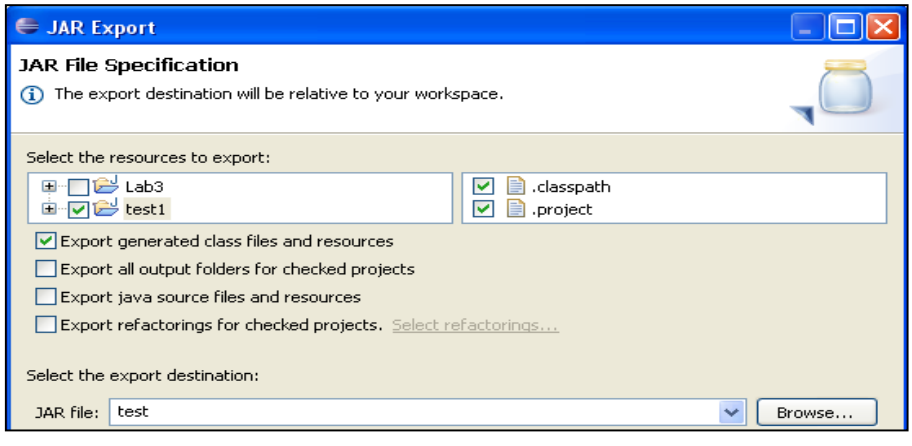
6.4: Miscellaneous Options

Jar File Creation

In the Package Explorer, you can optionally pre-select one or more Java elements to export

- Select Export from either the Context menu or from the File menu
- Expand the Java node, and select JAR file, and click Next
- On the JAR File Specification page, select the resources that you want to export
- Specify a name to the JAR file
- Click Finish to create the JAR file





6.4: Miscellaneous Options

Class path Setting

Classpath variables allow you to avoid references to the location of a JAR file on your local file system

Classpath variables can be used in a Java Build Path to avoid a reference to the local file system

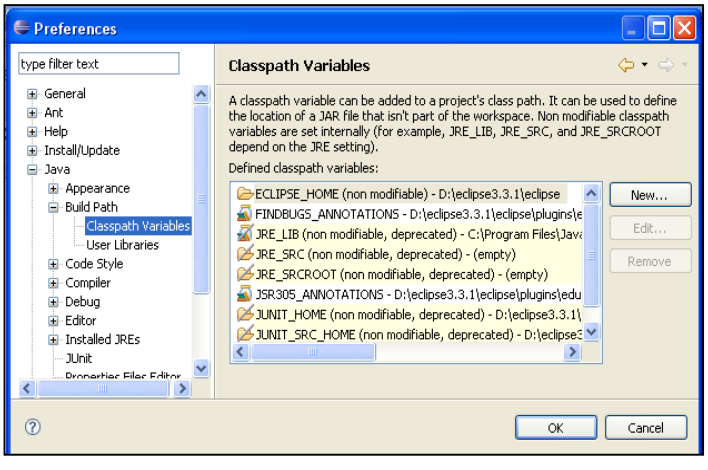
The value of such variables is configured at the following path:

- **Window → Preferences → Java → Build Path → Classpath Variables**

Miscellaneous Options:

Setting Classpath:

Command	Description
1. **New...**	It adds a new variable entry. In the resulting dialog, specify a name and path for the new variable. You can click the File or Folder buttons to browse for a path.
2. **Edit...**	It allows you to edit the selected variable entry. In the resulting dialog, edit the name and/or path for the variable. You can click the File or Folder buttons to browse for a path.
3. **Remove…**	It removes the selected variable entry.

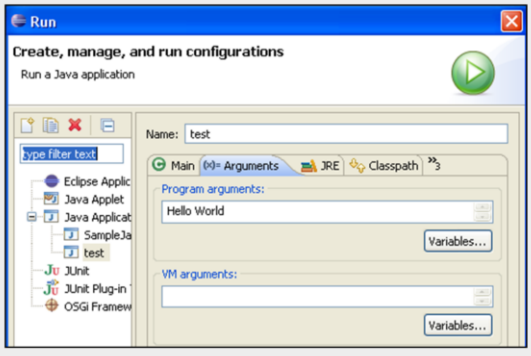


6.4: Miscellaneous Options

Passing Command Line Arguments

Command line arguments can be passed to the program in the following ways:

- Select **Run** → **Open Run** dialog → **Arguments** tab



Note: In the snapshot shown in the above slide, there are two arguments "**Hello**" and "**World**".

6.4: Miscellaneous Options

Import a Project

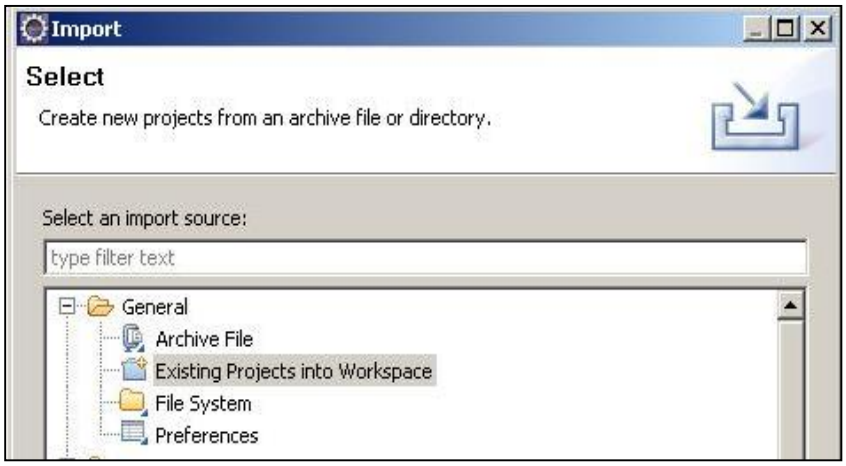
To import an existing project to the workspace:

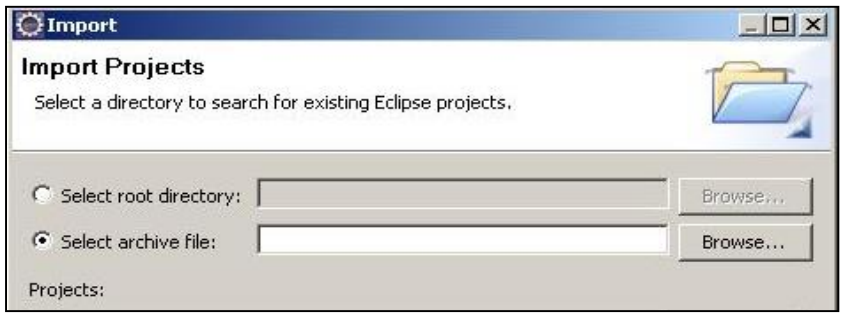
- Go to File ☐ Import
- Select Existing Projects into Workspace option
- Select the radio button next to Select archive file, and click the Browse button
- Find the archive file on your hard disk and click Open to select
- If you have selected an archive file containing an entire Eclipse project, then the project name will appear in the box below, that is already checked
- Click Finish to perform the import

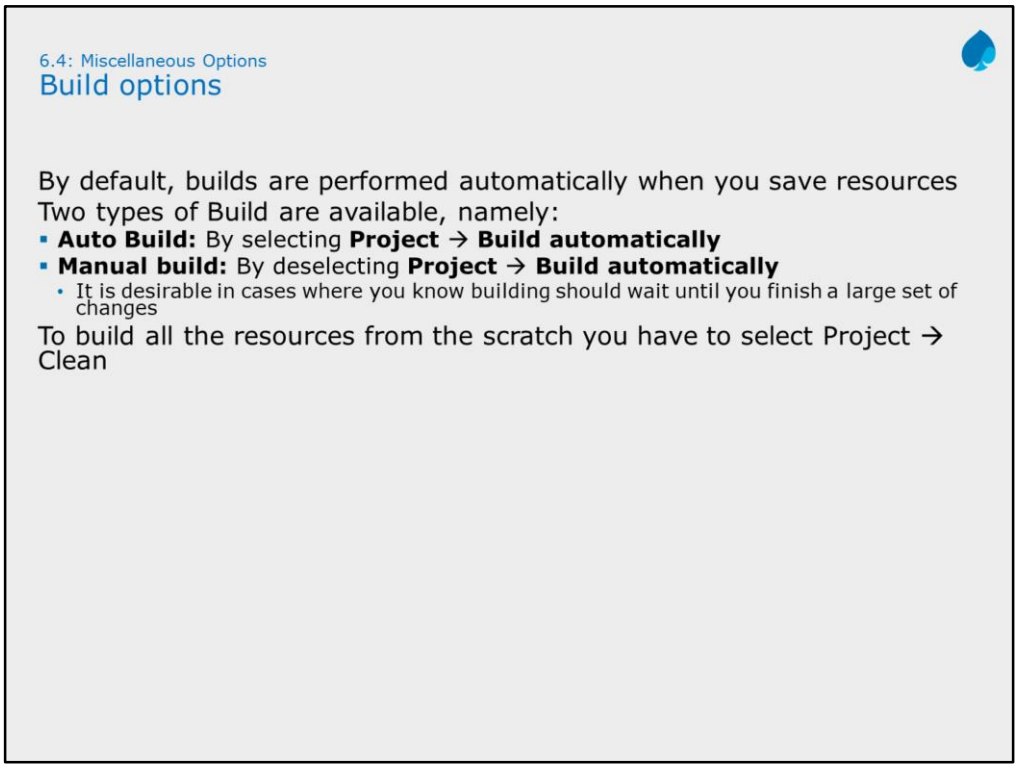
Import and Export Options:

Import a Project:

The snapshots of the above steps are given below:







Builds:

- Builders create or modify workspace resources, usually based on the existence and state of other resources. They are a powerful mechanism for enforcing the constraints of some domain.
  For example: A Java builder converts Java source files (.java files) into executable class files (.class files), a web link builder updates links to files whose name/location have changed, and so on.
- As resources are created and modified, builders are run and the constraints are maintained. This transform need not be one to one.
  For example: A single .java file can produce several .class files.

6.4: Miscellaneous Options

General Tips and Tricks

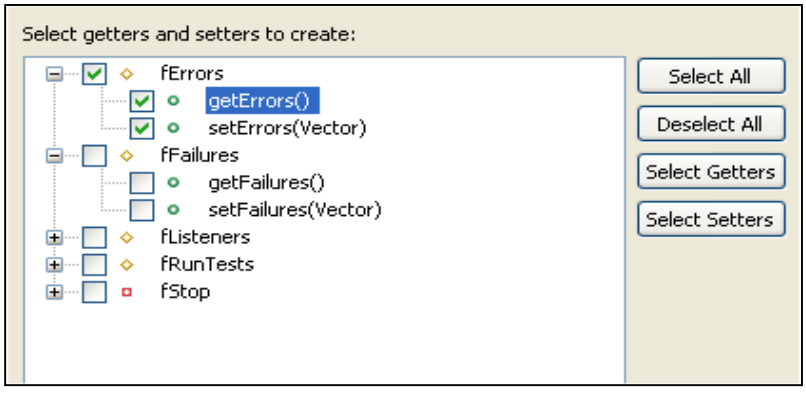
Creating Getters and Setters:

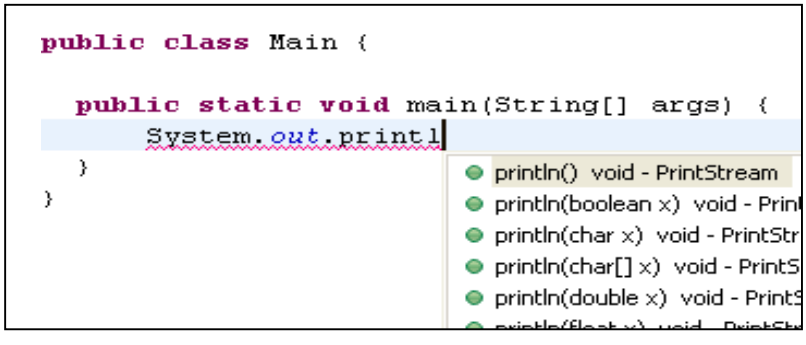
- To create getter and setter methods for a field:
  - Select the field's declaration
  - Invoke Source → Generate Getter and Setter

Content assist:

- Content assist provides you with a list of suggested completions for partially entered strings
- In the Java editor, press CTRL+SPACE or invoke Edit → Content Assist

Miscellaneous Options:
Tips and Tricks:
Creating getters and Setters :
Select the field's declaration, and invoke **Source → Generate Getter and Setter**.





6.4: Miscellaneous Options

General Tips and Tricks

Source menu contains a lot of options which can be used during code generation:

- **Code Comments:** You can quickly add and remove comments in a Java expression
- **Import Statements:** You can use it to clean up unresolved references, add import statements, and remove unneeded ones
- **Method Stubs:** You can create a stub for an existing method by dragging it from one class to another

6.4: Miscellaneous Options

General Tips and Tricks

- **Try / Catch statements:** You can create Try / Catch block for expression by Source → Surround with try/catch
- **Javadoc Comments:** You can generate Javadoc comments for classes and methods with Source → Add Javadoc Comment
- **Superclass constructor:** Add the superclass constructors with Source → Add Constructor from Superclass

6.4: Miscellaneous Options

Using Java documentation

For new developers, to quickly get familiar with the Java API, Java provides API documentation.

The documentation also provides description and examples for all methods of each class.

It can be downloaded from http://docs.oracle.com/javase/8/docs/api/ for offline access.

To see Java documentation for any class or method, eclipse provides "javadoc" view.

To enable this view, select Windows ☐ Show View ☐ Javadoc.

You can also view the javadoc contents in HTML format by using shortcut key "Shift + F6".

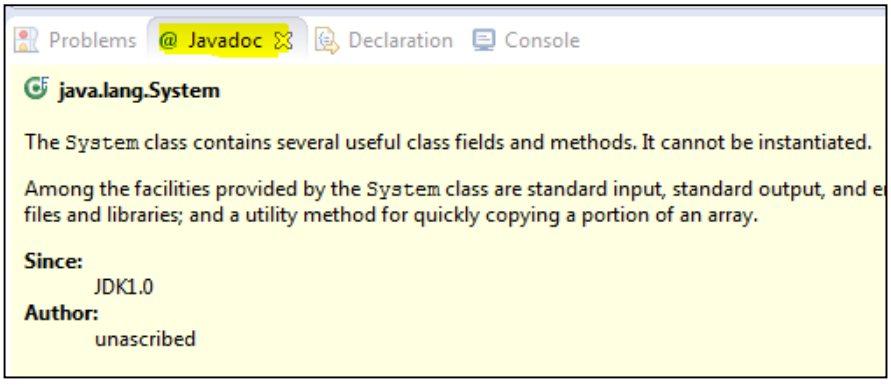
Using Java documentation:

Java API provides documentation which is good resource to get familiar with it. This documentation is valuable resource for programmers wishing to construct the applications in Java.

The documentation provides all information about the Java API. It includes list of classes or objects available to programmer.

Javadoc View:

Eclipse provides the javadoc view which shows the available documentation for selected class or method. To enable this view, select Windows → Show View → Javadoc.



Lab

Lab 1: Working with Java & Eclipse

Summary

In this lesson, you have learnt:

- The method to install Eclipse
- Process to create a Java Project with Eclipse
- Various useful features of Eclipse

Review Question

Question 1: Which of the following are true with Eclipse 4.4?

- **Option 1:** A Java Project in Eclipse has got a Java builder that can incrementally compile Java source files as they are changed
- **Option 2:** A workspace can have one project only
- **Option 3:** The source and class files can be kept in different folders

Question 2: To build all resources, even those that have not changed since the last build, you have to select the following option:

- **Option1:** Project → Build Project
- **Option2:** Project → Build All
- **Option3:** Project → Clean

Add the notes here.