Introduction to Eclipse

In this chapter, we will learn about very frequently used IDE like Eclipse. Knowledge of IDEs is equally important as knowledge of language. We will see basic information regarding use of eclipse and creating different types of projects and how it supports compilation and execution.

Need of IDE

IDE's are a migration form of primitive text editors, which uses full functionality technologies that quickly and efficiently edit the code. IDE is the Integrated Development Environment that provides the user interface for code development, testing and debugging features. It helps to organize the project artifacts that are relevant to the source code of the software application. It provides several tools and features to make the development easy and standardize based upon the programming language used.

It contains development tools such as text editors, code libraries, compilers, and test platforms and consists of at least build automation tools and a debugger.

Some IDE's will work on a specific programming language, and also they contain cross-language capabilities. IDE's such as Eclipse, ActiveState Komodo, IntelliJ IDEA, My Eclipse, Oracle JDeveloper, Net Beans, Codenvy and Microsoft Visual Studio supports multiple languages.

Benefits :

- IDE can develop software applications by using a set of tools, which makes it easier to write programs without spending much time on language syntax. IDE has the ability to correct syntaxes, gives a warning about memory leaks, assist in writing quality of code, etc. The main objective of using IDE is that it allows coding quickly and efficiently.
- IDE includes built-in compilers, which convert the program into machine-level code or byte code and saves a lot of time. Programming languages can even be selected in some cases.
- Some IDE's include server like Net Beans or GlassFish server to test the web applications easily.
- It helps to keep track of the code, generates code and allows searching.
- When creating applications, IDE manages resources such as library files, header files, etc., at specified locations. This includes pre-installed libraries for a specific programming language.

Installation

Installation

The Eclipse IDE (integrated development environment) provides strong support for Java developer. In 2022 the Eclipse IDE has approximately one millions downloads per month which makes it one of the leading IDEs for Java development.

Eclipse can be extended with additional software components called plug-ins. Pre-packaged Eclipse distributions provide a consistent set of functionality.

The Eclipse IDE for Java Developers distribution is designed to support standard Java development. It includes support for the Maven and Gradle build system and support for the Git version control system.

The Eclipse Foundation and its community of projects and contributors produce releases on a coordinated schedule that are often referred to as simultaneous release, coordinated release or release train of Eclipse. This page provides an index of existing simultaneous releases from the current and previous years.

Since the 2018-09 release, the cadence changed from one annual main release plus 3 update/service releases to a 13-week cycle with rolling releases (after every 3 months).

Before the 2018-09 release, each main release were named with starting character in the series of alphabets. e.g **Kepler, Luna, Mars, Neon, Oxygen etc.**

Recently the Eclipse Installer is introduced, a new and more efficient way to install Eclipse. It is a proper installer (no zip files), with a self-extracting download that leads you through the installation process.

Packages and zip files are still available on the downloads page of eclipse.org.

1. Download the Eclipse Installer

   Download Eclipse Installer from http://www.eclipse.org/downloads

   Eclipse is hosted on many mirrors around the world.
2. Start the Eclipse Installer executable

   Eclipse Installer should be executed. You may get a security warning to run this file.
3. Select the package to install

   The new Eclipse Installer shows the packages available to Eclipse users. Select and click on the package you want to install.
4. Select your installation folder

   Specify the folder where you want Eclipse to be installed. The default folder will be in your User directory.
5. Launch Eclipse

   Once the installation is complete you can now launch Eclipse.

Workspace, workbench, perspective

Workspace :

Workspace, workbench, perspective

Workspace :

The workspace is the physical location (file path) for storing meta-data and (optional) your development artifacts. The meta-data stored for the workspace contains preferences settings, plug-in specific meta data, logs etc.

You can choose the workspace during startup of Eclipse or via the File Switch Workspace Others menu entry.

Your projects, source files, images and other artifacts can be stored inside or outside your workspace. For example, if you use Git as version control system, you typically would store the Git repositories outside of the workspace.

Workbench :

The Workbench provides the user interface structure for Eclipse. The purpose of the Workbench is to facilitate the seamless integration of tools. The Workbench is responsible for the presentation and coordination of the user interface.

The Workbench contains one or more WorkbenchWindows, each of which contain zero or more WorkbenchPages.

Eclipse's main window, called the workbench, is built with a few common user interface elements. The two most important elements are views and editors.

Eclipse offers views and editors to navigate and change content. A view is used to work on a set of data, which is in a hierarchical structure. If data is changed via view, the underlying data is directly changed e.g Package explorer.

Workbench saves the state while exiting from Eclipse so that when eclipse is restarted it is opened in the same state.

Perspective :

Perspective is a combination of different views and editors suitable for a particular type of applications.

Eclipse provides different perspectives for different tasks. The available perspectives depend on your installation. For Java development you usually use the Java Perspective, but Eclipse has much more predefined perspectives, e.g., the Debug perspective.

Eclipse allows you to switch to another perspective via the Window > Perspective > Open Perspective menu

Open editors are typically shared between perspectives, i.e., if you have an editor open in the Java perspective for a certain class and switch to the Debug perspective, this editor stays open.

The main perspectives used for Java development are the Java perspective and the Debug perspective.

The Java perspective can be opened via Window > Perspective > Open Perspective > Java.

The Window > Perspective > Customize Perspective... menu entry allows you to adjust the selected perspective.

Eclipse project

A Java project contains source code and related files for building a Java program. It has an associated Java builder that can incrementally compile Java source files as they are changed.

A Java project also maintains a model of its contents. This model includes information about the type hierarchy, references and declarations of Java elements. This information is constantly updated as the user changes the Java source code. The updating of the internal Java project model is independent of the Java builder; in particular, when performing code modifications, if auto-build is turned off, the model will still reflect the present project contents.

Creating a project

Creating new project:

Project can be created from scratch by supplying the name and location of the project.

- On the main menu bar, click File > New Project. The New Project wizard opens.
- Select a category from the left column and then select the type of project to create from the right column. To assist in locating a particular wizard, the text field can be used to show only the wizards that match the entered text. Click Next.
- In the Project name field, type a name for your new project.
- (Optional) The project that you create will map to a directory structure in the file system. The default file system location is displayed in the Location field. If you want to create the project and its contained resources in a different location, clear the Use default location checkbox and specify the new location.
- If you want the new project to be dependent on one or more other projects, click Next and select the projects to be referenced.
- Click Finish. The new project is listed in one of the navigation views.

Importing existing projects :

You can use the Import Wizard to command link import an existing project into workspace.

- From the main menu bar, select command link File > Import.... The Import wizard opens.
- Select General > Existing Project into Workspace and click Next.
- Choose either Select root directory or Select archive file and click the associated Browse to locate the directory or file containing the projects.
- Under Projects select the project or projects which you would like to import.
- Click Finish to start the import.

Debugging in Eclipse

Need of debugging :

Debugging is a technique that is used to see your code execute line by line. That means you are able to see the execution of each line of your code and stop at any line of the code and analyze the code, the variables, and the values these variables carry at the time you have halted the execution.

Debugging is used extensively to find runtime errors in your code. Compiler errors are highlighted in the code itself in the Eclipse IDE and with compile-time errors, you cannot proceed to run your program.

However, if you have runtime exceptions, then they may not be highlighted in the code, instead, when you run the code, your program will fail due to this exception.

Performing debugging :

To take manual control of your program at the time of execution you need something called a breakpoint. A breakpoint is a way to tell the debugger from where you want to take control of your code.

The debug perspective will show the following window:

- Debug window: Right next to the project explorer the debug explorer is opened, in which the class being debugged is displayed.
- Class: This is the class that you want to debug.
- Variables: This section is where you can view the variables and how their state is changing during execution. By right-clicking on the variables displayed here, you can do multiple operations on them like change them or view their data type, etc.
- Breakpoint: In this section, you can view and change breakpoints (explained further). From here you can perform advanced operations on breakpoints such as defining conditions on them.
- Console: This is where you can see the execution happening.

Now, you have the program in debug mode and have placed the required breakpoints where we can run the code in debug mode.

Right-click on the class name from the project explorer and click on Debug As -> Java Application.

The line on which the breakpoint is applied is highlighted and the execution of code has stopped at that point.

Following keys will help in debug operation :

1. **Step into or F5:** Using this, you can execute the line of code you are at and move to the next line.
2. **Step over of F6:** In this case, the code will execute normally till you keep hitting F6 and in the end, you will get he exception as you do while normally executing.

Performing debugging :

To take manual control of your program at the time of execution you need something called a breakpoint. A breakpoint is a way to tell the debugger from where you want to take control of your code.

The debug perspective will show the following window:

- Debug window: Right next to the project explorer the debug explorer is opened, in which the class being debugged is displayed.
- Class: This is the class that you want to debug.
- Variables: This section is where you can view the variables and how their state is changing during execution. By right-clicking on the variables displayed here, you can do multiple operations on them like change them or view their data type, etc.
- Breakpoint: In this section, you can view and change breakpoints (explained further). From here you can perform advanced operations on breakpoints such as defining conditions on them.
- Console: This is where you can see the execution happening.

Now, you have the program in debug mode and have placed the required breakpoints where we can run the code in debug mode.

Right-click on the class name from the project explorer and click on Debug As -> Java Application.

The line on which the breakpoint is applied is highlighted and the execution of code has stopped at that point.

Following keys will help in debug operation :

1. **Step into or F5:** Using this, you can execute the line of code you are at and move to the next line.
2. **Step over of F6:** In this case, the code will execute normally till you keep hitting F6 and in the end, you will get he exception as you do while normally executing.
3. **Step out or Step return or F7:** This key will finish the execution of the method being debugged and return to the code from where this method is being called.
4. **Resume or F8:** This option will tell the debugger to continue executing the program until the next breakpoint is reached.