Integrated Development Environments (IDEs)

Definition from [1]:

> A development environment, or integrated development environment (IDE), provides comprehensive facilities to programmers for software construction by integrating a set of development tools. The choices of development environments can affect the efficiency and quality of software construction.
>
> In additional to basic code editing functions, modern IDEs often offer other features like compilation and error detection from within the editor, integration with source code control, build/ test/debugging tools, compressed or outline views of programs, automated code transforms, and support for refactoring.

Eclipse and Spring Tool Suite

In CO2006, we are using the Spring Tool Suite, which is a plugin over the Eclipse IDE. We have chosen this IDE because it is one of the most popular IDE according to the PyPL ranking. It also is cross-platform, extensible (via plugins), open-source and free - you will be able to continue using it after your graduation.

Below, the most important components of the Eclipse IDE, which we will use in the lab exercises, are presented from a conceptual point of view. These components will be discussed in more detail during a taught lab, including additional pointers. Components that are specific to the STS will be presented as they are needed during the module.

Sometimes the Eclipse IDE is also referred to as the Eclipse workbench.

The information below has been extracted from (the Eclipse Workbench User Guide).

Workspaces and Projects

Eclipse uses two basic concepts to manage your code:

- **Workspace**: A "working directory" for all of your projects. Think about it as a root directory that you can create for each module. And advantage of using workspaces is that they are self-contained: you can keep it on a flash drive, plug it into a computer, start Eclipse, select that workspace and work from there.
- **Projects**: A collection of related code, usually and independent program, enclosed in a folder. Strictly speaking a **folder** corresponds to the graphical element representing a project and a **directory** is a filesystem concept.

That is, a **project** is displayed as a folder in the workspace and is persisted in a directory in the filesytem. Whenever you create a project inside of Eclipse, Eclipse will create a new directory in your filesystem and a folder in your workspace with the same name and it may happen that:

- the name of the folder in your filesystem does not coincide with the name of the directory in the filesystem (this may happen if the project name is changed after creating the project);
- the folder is moved to a different location (this will happen when you share a project on git).

Views and Editors

A **view** is a visual component within the Workbench. It is typically used to navigate a list or hierarchy of information (such as the resources in the Workbench or Java packages), or display properties for the active editor. Modifications made in a view are saved immediately. The following views are likely to be very useful:

- Help view: where this information has been extracted from.
- Outline view: displays an outline of a structured file that is currently open in the editor area.
- Problems view: as you work with resources in the workbench, various builders may automatically log problems, errors, or warnings in the Problems view. For example, when you save a Java source file that contains syntax errors, those will be logged in the Problems view. When you double-click the icon for a problem, error, or warning, the editor for the associated resource automatically opens to the relevant line of code.
- Search view: This view displays the results of a search.

An **editor** is also a visual component within the Workbench. It is typically used to edit or browse a resource. The visual presentation might be text or a diagram. Typically, editors are launched by clicking on a resource in a view. Modifications made in an editor follow an open-save-close lifecycle model.

Some features are common to both views and editors. We use the term **part** to mean either a view or an editor. Parts can be active or inactive, but only one part can be active at any one time. The active part is the one whose title bar is highlighted. The active part is the target for common operations like cut, copy and paste. The active part also determines the contents of the status line. If an editor tab is not highlighted it indicates the editor is not active, however views may show information based on the last active editor.

Perspective

A **perspective** is a group of views and editors in the Workbench window. One or more perspectives can exist in a single Workbench window. Each perspective

contains one or more views and editors. Within a window, each perspective may have a different set of views but all perspectives share the same set of editors. The perspectives that you are likely to find useful are: Java, Debugging, Git, Spring.

The Java development tools (JDT) include several perspectives among which we will use: Java and Debug.

Java perspective

The **Java perspective** provides a Java editor, which supports syntax highlighting, content/code assist and integrated debugging features, and several views.

Some useful views are:

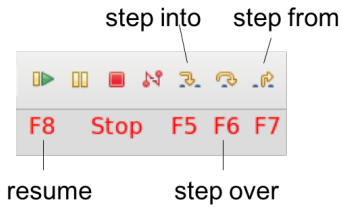
- the Console View which is used for showing the results of executing a Java program;
- the Package explorer.

The most important tasks are: launching a Java program

Debug perspective

The **Debug perspective** gives you access to a Java debugger, which allows you to control the execution of your program by setting breakpoints, suspending launched programs, stepping through your code, and examining the contents of variables. A **breakpoint** suspends the execution of a program at the location where the breakpoint is set. A breakpoint can be enabled by double-clicking on the ruler that appears in the left-hand side of the Java editor window (see here for further information).

The most important view is the Debug view, which includes all of the commands that can be used to execute debugging tasks. Some of the most important ones are: resume, step into, step over, step return, stop and relaunch.



The most important tasks are: launching a Java program in debug mode, which involves stepping through the execution of a program, inspecting values and evaluating expressions.

A final tip

There may not be *the best IDE* and its choice usually depends on the needs of the project in which you are working and most likely on the culture of the organisation/team that you work for. However, the main components that we have discussed above are prevalent in many of them and the earlier you get familiarised with them the faster you will be able to transition from one IDE to the other, becoming agile at adapting to the requirements of a new working environment.

Additional Resources

- :movie_camera: Eclipse Guided Tour for Java

References

References

[1] P. Bourque and R. E. Fairley. *Guide to the Software Engineering Body of Knowledge (SWEBOK(R)): Version 3.0*. IEEE Computer Society Press, Los Alamitos, CA, USA, 3rd edition, 2014.