P1L3 Integrated Development Environment

Notes 08/25/2014

Tools are a cornerstone of the software engineering discipline, and it is of paramount importance to know and use them.

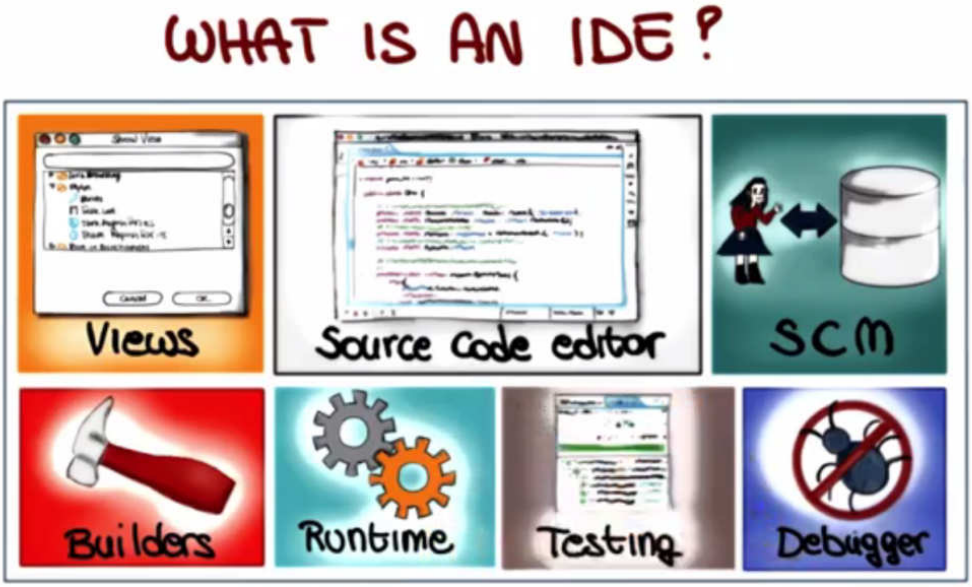
Integrated Development Environments (IDEs) are software applications that support developers in many of their everyday tasks, such as writing, compiling, and debugging code.

We will focus on a specific ID, Eclipse.

- *First present Eclipse*
- *Then get some hands-on experience through a demo.*

Eclipse Introduction

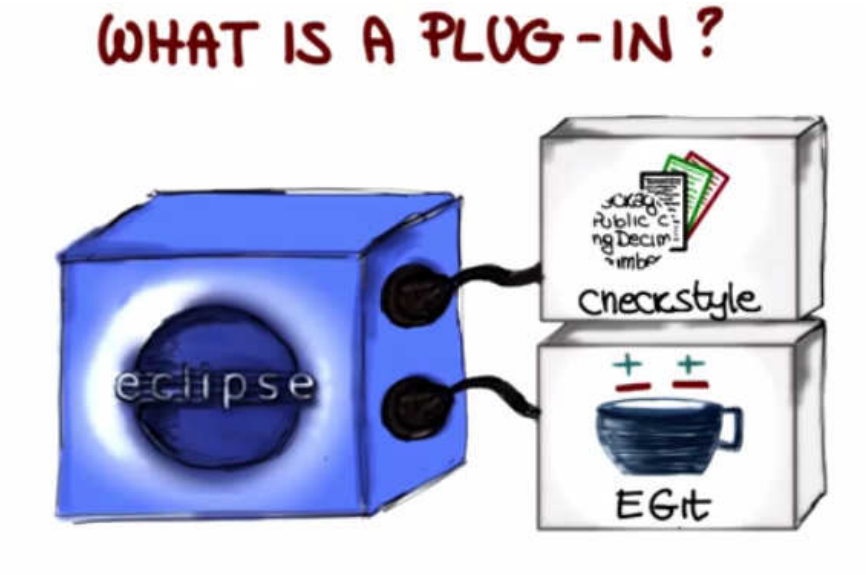
- Tools are fundamental in software engineering.
- IDEs are environments that give you support for your development activities, i.e.,
    - Writing code
    - Editing code
    - Compiling code
- Eclipse is an open, extensible development environment that was initially created by IBM and is now managed by the Eclipse Foundation
- Each version of Eclipse is named differently
    - Helios (2010)
    - Kepler (2013)
- Eclipse is open and multi-platform, which means that you can use Eclipse no matter what operating system you're using.
    - Eclipse runs on the most commonly used operating systems such as Mac OS, Windows, Linux,
    - No matter what you're using, you'll be able to install and run Eclipse
- There are many other great ideas
    - Microsoft Visual Studio
    - Netbeam



IDE Overview

- An IDE is a software application that supports software developers in many of their everyday tasks.
- It has many useful features.
    - Most IDEs provide views that can be used to navigate, project resources from different perspectives.
        - You might want to look at your code differently when you're writing code, and when you're debugging.
    - They also normally provide an intelligent source code editor.
        - An editor that will allow you to browse a documentation when you're writing code that uses a specific method
        - That will give you auto-completion when you start writing the name of an objection and you want to get the methods for that object.
        - And all of these things can be very useful while you're developing and can save you a lot of time.
    - Modern IDE's will also normally give you support for version control systems that you can use for softer configuration management.
    - IDEs also give you builders so they give you build automation tools
    - They give you runtime support
        - So that you can run your projects from within the IDE or some aspects of the execution.
    - They give you support for testing.
        - Many IDEs allow you to run tests from within the IDE and to check the results of the tests from within the IDE.

        - Normally, after you run your tests, if there are some test cases that fail, you can also use your IDEs to do debugging.
    - Many IDEs include graphical debuggers.
        - Debuggers will allow you to navigate through the code, set watch points, stop and restart the execution, inspect variables, and do all of the activities that help debugging and help you be more efficient and more effective when you do debugging.
- And into addition to all these features that are listed here IDEs can normally provide you even more features through a mechanism that is called plugins.



Plug Ins

- Most IDEs are extensible through the use of plug-ins.
- Plug-ins might be called differently on different platforms.
    - In Microsoft Visual Studio, plug-ins are normally called add-ins, but the concept is more or less the same.
- If we imagine our IDE as a box, a plug-in is additional functionality that you can actually plug into this box so that this box starts offering more features to the user.
- Example of plug-ins for Eclipse:
    - Checkstyle plug-in - helps you ensure that your Java code complies with a set of coding standards by inspecting the code and pointing out items that deviate from a defined set of coding rules.
        - This is a functionality the core of Eclipse doesn't have.

        - You can add the Checkstyle plug-in, and this functionality will become available in the IDE.
    - EGit plug-in which adds support for the Git version control system in Eclipse.

Eclipse Demo Create Java Project

- Some of the basic aspects of eclipse
    - How to run eclipse
    - How to select their workspace
    - How to create a project
    - How to create the class within the project
- Some more advanced aspects,
    - How to create builders
    - Run your project within Eclipse
    - How to use their Eclipse debugger.
- Eclipse is going to ask me for the location of my workspace and in this case, I selected a suitable directory and you can also use that checkbox on the left to avoid Eclipse for asking you again about where to put the workspace.
    - And the workspace is basically the directory where Eclipse will place all of your projects.
    - If it's the first time you start Eclipse you might get this Welcome screen.
        - It's not going to happen again on subsequent executions.
- What you want to do now is basically go to the Java Perspective which you can do by going to Window, open Perspective, and if the Perspective is not here, you'll have to click on Other. And at this point, that you can click on Java Perspective, then you click okay.
    - The perspective is basically, the visual work space where you will be operating.
    - So, after we selected perspective, we can actually close the welcome screen (to get it again, click on Help then open).
- And here, you see that you have this different areas
    - On the left you have the package explorer.
        - This is the area where your packages will be
    - You've got a task list, and an outline on the right.
    - And then you have underneath, the bottom, a problem, java doc and declaration views and we will see some of these views in actions later.
    - In the center you have a code editor, which is where you'll be writing, editing, and modifying, basically, your code.
- Let's start by creating a Java project.
    - Use either the context menu, or you can just use the menu, select new Java project. You'll be greeted by a new java project wizard and at this point in the wizard, you can select the name of your project.
    - If you use the default location for the project, it will be placed in the work space that you selected before.

    - Use the default Java Runtime Environment, which is Java 1.7 in this case.
    - Keep the selected default layout and then go to the next step.
    - We're first presented with the location of the source code for our project. The default is a directory SRC in my project, and for the output file, the directory bin; no need to change that
    - Under project, in case you need other projects to build your own, then you can specify them here.
    - Under Libraries, we can specify which libraries our project requires.
        - The Java library's already specified.
        - You can also add other jars (Java Archive), which can even be External jars.
    - Under Order and Export, And finally this is the tab that allows you to specify which part of you project or how your project will be exported,
    - Click finish.
- On the package explorer you should see the new project appear
- We can open the project by clicking on the triangle right next to it, and as you can see there is the SRC directory, where my source code will go, and there's also an indication that we're using the JRE, so that's the Java system library within our project.
- Under the hood
    - (In cmd prompt) we can go to the directory where the project was created. We can see the bin and src directories. And there's also some other files here that you can see this dot files that you will not normally, see.
    - And those are kind of bookkeeping Files. So these are files that contain information about your project and that are created automatically by Eclipse.
    - And, for example, will have various indication about the configuration of the project, some settings and the class path for the project.
    - You don't have to worry about this if you just want to go Eclipse since you're never going to mess with the command line.

Eclipse Demo Create a Class

- Now we can start creating a package.
    - A package is basically a way of organizing your classes into a hierarchy.
    - If you specify a package name such as edu.gatech, this means that you're creating really two packages, a package gatech inside package edu.
    - You can start creating classes inside your packages.
- Use the contextual menu, select New>Class, and you'll get another wizard that will allow you to specify the name of the class.
    - Call it Hello World.
    - There's many other parameters you can set, and in particular, you can define whether you want a main method in your class, where having a main method means that your

class can be the main class in your project, can be the one that is run when you run your project.
    - After we click the finish button we get the class; so we also get template code for the class
    - Go to the editor function, you can see that there is a to-do where you have to put your code, and here we are simply, basically printing, you know, the typical first program. We just going to print Hello World in Java.
        - System.out.println("Hello World");
    - And something you can note is that as we are typing, Eclipse gives us auto-complete suggestions, which is very helpful.
        - In case you don't remember the exact syntax, or the method, or you don't remember the parameters of the method, this is very helpful
- To run our code we can either click on the green arrow button, or we can right-click in the Call window and select Run As Java Application.
    - Eclipse will run our tool, and it will create a console that basically contains the textual output of the program. And as expected, the output is Hello World.

Eclipse Demo Run Configuration

- Let's see what happens exactly when you run a program within Eclipse.
- Go to the Run menu and select Run Configurations, and this brings up a windows where you can change your run configurations.
    - Well first of all, you can see that here on the left under Java application, Eclipse automatically created a Hello World run configuration for our program.
    - This is where you can configure the different parameters for your execution. For example, you can select the main class.
        - Here it's edu.gatech.HelloWorld.
        - You can define different program arguments. We don't have any for now.
            - You can also pass arguments to the virtual machine.
        - You can define which Java runtime environment you want to use
        - You can define the Class path
        - You can define other environmental options
        - So let's now try to pass some arguments to our program.
            - Go back to arguments
            - Under program arguments, type George, hit apply, and then run.
            - if you run the program of course, the output is not changing because the program does not use the argument but.
            - Modify the program so that now, instead of printing hello world, we will print hello followed by the first argument that I will pass to the program.
                - System.out.println("Hello " + args[0]);

- Run the program and you will get Hello George.

Eclipse Demo Debugging

- Create a new file called AddNumbers which takes two numbers, parses them into integers, adds them and prints the sum, supposedly, of the two numbers.
- Look at the run configuration for this program, and here you can see that we're passing two arguments, two and five, to the program.
- Run our program and see what happens. Result is incorrect
- We need to debug our program, or figure out what's wrong with it
- Add a break point by double-clicking on the side of the code.
    - The break point is basically a place where you're telling the debugger to stop during the execution because you want to inspect the state of the program.
- To start debugging, select Debug as Java Application from the Context menu, similar to what we were doing for running the program.
- This asks us whether we want to pass to the debug perspective, which is a perspective specifically designed for debugging. Say yes.
    - It shows us a different set of views
        - The view on the right-hand side, for example, shows the variables in scope and the break points that are currently active for the debugging session.
        - On the left it shows you where the editor is
        - You see the outline of the program
        - Console at the bottom
- Execute one line by clicking on the Step Over button at the top, and this will execute the line that is currently highlighted and therefore it will move to the next line.
- If I move the mouse over a variable, you can see the value of the variable
- You can do the same thing in the variables windows on the right; if you click it, it will tell you what is the value of the variable, and in case of more complex variables, you can even expand it and get more details.
- Notice that we are doing multiplication instead of addition.
- If you want to stop the debugger because you're done with your debugging session, you can either click on the Terminate button or you can also just simply tell the debugger to continue the execution, to resume the execution until the program terminates naturally.
- To go back to normal view, click on "Java" in the upper right corner instead of the selected "Debug"