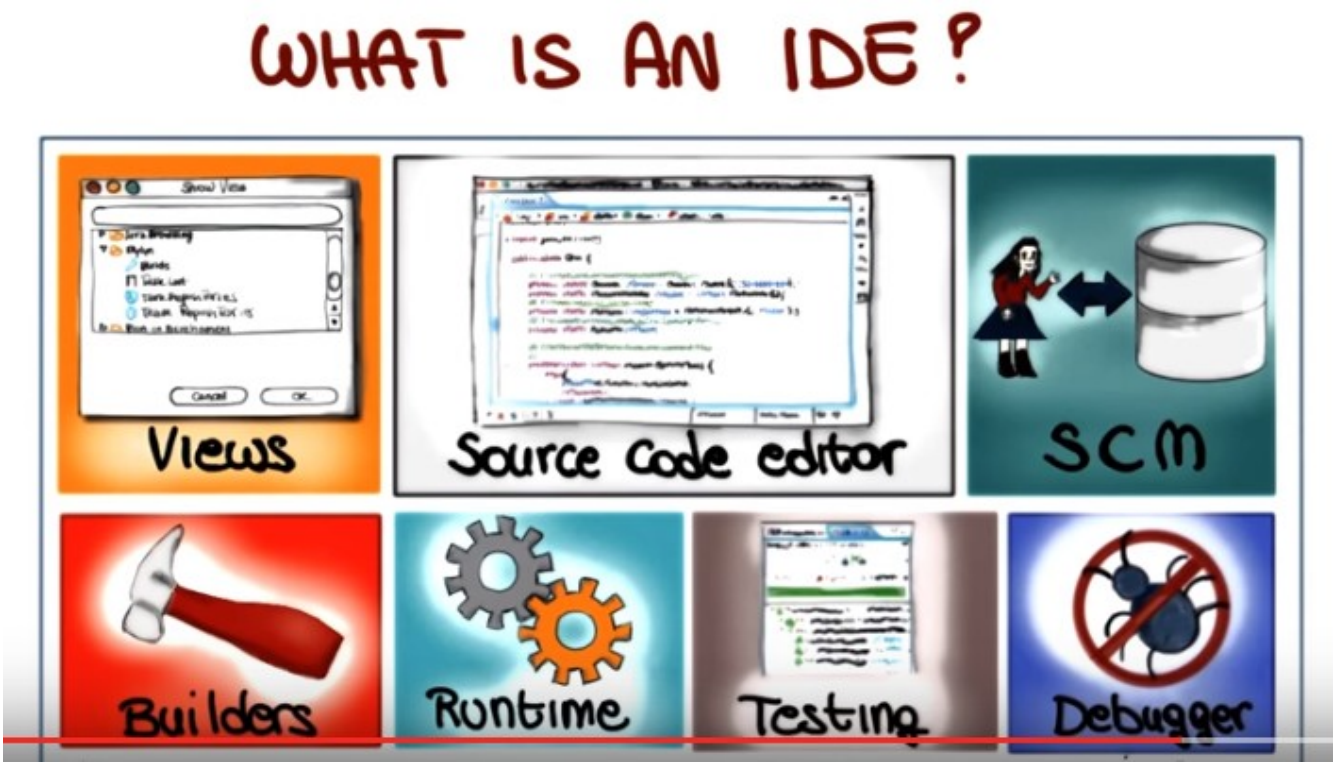
Import existing projects (may need the .classpath and .project files to be able to import):
File->Import->General->Existing Projects into Workspace->Select root directory (if the folder src is in folder Assignment3 中, then select Assignment3 as the root directory)
Make sure that (1) “Assignment3” is selected under “Projects:” and (2) “Copy projects into workspace” is not selected
Click “Finish”

1. Hi, and welcome to the first of several lessons on tools of the trade. I'm very excited about these lessons, because I believe that tools are a cornerstone of the software engineering discipline, and it is of paramount importance to know and use them. In this lesson, we will talk about integrated development environments, normally called IDEs. And these are software applications that support developers in many of their everyday tasks, such as writing, compiling, and debugging code. And to make the discussion more concrete we will focus on a specific ID, Eclipse. We will first present Eclipse, and then get some hands-on experience through a demo.

2. As I just told you, tools are fundamental in software engineering. And I will stress this concept over and over, throughout the class. And today we're going to talk about a tool that is especially important, which is integrated development environments, or IDEs. And you're probably familiar with IDEs. So IDEs are environments that give you support for your development activities. For example, for writing code, editing code, compiling code, and so on. And we will focus specifically on one particular IDE, which is called Eclipse. And what I'm showing here is the two splash screens for two versions of eclipse, Helios and Kepler. Eclipse is an open, extensible development environment that was initially created by IBM and is now managed by the Eclipse Foundation. And of course, there are many other great ideas such as for example, Microsoft Visual Studio or Netbeam. We will be using Eclipse because it is open and because it is multi-platform, which means that you can use Eclipse no matter what operating system we're using. So we can see that the most commonly used operating system, such as Mac OS, Windows, Linux, Eclipse runs on any of these environments. Therefore, no matter what you're using, you'll be able to install Eclipse, run Eclipse, and follow the class.

3. So, now let's look in a little more detail to what is an IDE. 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. For example, 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. For example, an editor that will allow you to browse a new implementation when you're writing a code that is uses a specific method, or that will give you autocompletion 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 then you can use for softer configuration management. And we're going to discuss in detail version control systems in the next tools of the trade lesson, and we're also going to see how it can be integrated within an IDE. 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 and, for example, of service, some aspects of the execution. In addition to giving you support for the runtime, 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. Not only that. 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 which points, stop and restart the execution. Inspect variables, and do all of the activities that help debugging. And, to 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 mechanishm that is called plugins.



4. In fact most IDEs are extensible through the use of plug-ins. And by the way, note that plug-ins might be called differently on different platforms. For example, if you're using a Microsoft Visual Studio, plug-ins are normally called add-ins, but the concept is more or less the same. So, what is a plug-in? (就是(給 Eclipse 等)外加的功能) Well, let's imagine our IDE to build this 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. For example, you can add to Eclipse the Checkstyle plug-in. Which, paraphrasing the Checkstyle website, 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. Again, 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. Another example of plug-in is the EGit plug-in which adds support for the Git version control system in Eclipse. And actually this is something that we'll cover in detail, we'll have a demo, and we will actually use it throughout the class, so I'm not going to say anything more about the EGit plug-in for now. But again, what the plug-in will do is to add the Git functionality to Eclipse. A functionality that is not in the core of Eclipse and that is available to the user after you add the plug-in.

5. In the rest of this lesson we're going to look at eclipse and try to get more familiar with eclipse in a hands on manner through a demo. In the demo we will cover some of the basic aspects of eclipse like how to run eclipse, how to select their workspace, how to create a project, how to create the class within the project and so on. I'll also cover some more advanced aspects, like how to create builders, run your project within Eclipse, and how to use their Eclipse debugger. So let's get to the demo. So let's start Eclipse. 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 place the directory.

Where, Eclipse will place all of your projects. So, now when you start Eclipse, if it's the first time you might get this Welcome screen. It's not going to happen again on subsequent executions, but I just wanted to make sure that I covered all the bases. And so, whatcha want to do here is to basically go to the job of Perspective which you can do by clicking over there or you can also use the menus. So in this case we will have to go to Window, open Perspective(右上角的田字鍵也有這個功能), 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 (在我電腦上就是 Java), then you click okay. And 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. And here, you see that you have this different areas and on the left You have the package explorer. This is the area where your packages will be, you've gotta task list, and an outline on the right which we'll cover later. 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. And here in the center you have the area. Which is called a code editor, which is where you'll be writing, editing, and modifying, basically, your code. This is where most of the action takes place. So let's start by creating a Java project (即 File->New->Java Project). And to do that we can use either the context menu, or you can just use the menu, select new Java project. You'll be greeted by this, wizard, and. And at this point in the wizard, you can select the name of your project. I'm just going to call it a very simple way my project. And I going to use the default location for the project, as you can see it will be placed in the work space that I selected before. I'm going to also use the default. Java Runtime Environment, which is Java SE 1.7 in this case.

I'm going to kill the selected default layout and the, then I'm going to go to the next step. Here, 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. So repeat, we're now going to change that. Here in case you need other projects to build your own, then you can specify them here. Here we are building a simple project, so there's no need for that. And here we can specify which libraries our project

requires. As you can see, the Java library's already specified. And you can also add other jars, which can even be External jars. And finally this is the tab that allows you to specify which part of you project. So how your project will be exported, so lets not worry about that for now. Lets click finish. And as you can see here on the package explorer, my project appeared. So now we can open the project by clicking on the tree angle 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 directory within our project. And this is just for people who are interested in what happens you know, under the hood. So if you don't care about that, you can just skip this part. So basically here I'm showing you how 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 [UNKNOWN] 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, we'll have various indication about the configuration of the project, some settings and the class path for the project. And, as I said, you don't have to worry about this if you just want to go Eclipse since you're never going to mess with the [UNKNOWN].

6. So now that we know, we saw what happens under the hood, and as I said, don't worry about it if you don't care about that part. Now we can go back to Eclipse, and we can start creating a package (即在 src 上點右鍵，然後 New->Package). A package is basically a way of organizing your classes into a hierarchy (Java 書上講過). In this case, I'm going to specify the package name as edu.gatech, which means that I'm creating really two packages, a package gatech inside package edu.

深刻理解 package 和文件夾: 和 package 在電腦裡看來, 就是一個文件夾, 但它跟普通的文件夾還不一樣. 同一個 package 中可以建多個放代碼的文件夾(即 source folder), 例如本例中, 可以建一個跟 src 平行的 source folder, 比如叫 test (即在電腦上看來 MyProject 中有 src 和 test 兩個文件夾), 方法是, 以下的 MyProject/src/edu.gatech.HelloWorld.java 建好後, 選 File->New->Source folder. 注意必須這樣弄, 如果只是手動地在 MyProject 中(或 MyProject/src/edu.gatechl 中)手動建一個叫 test 的文件夾, 則在 Eclipse 中沒法往 test 中加 java 文件, 即以下新建 class 時, 跟本都看不到 test 這個文件夾. 按這樣建好 test 這個 source folder 後, 可以在 test 中新建 class 文件或 Junit test class 文件, 最終結果是: MyProject 中有 src 和 test 兩個文件夾, 更进一步, MyProject 中有 src/edu.gatech 和 tes/edu.gatech 兩個文件夾, 這兩個文件夾中的 java 文件都在 edu.gatech 這個 package 中(盡管這兩個 edu.gatech package 分別在 src 和 test 文件夾中, 但它們卻是同一個 package).

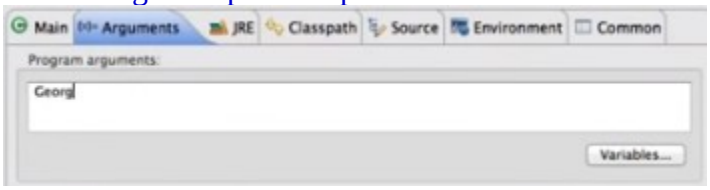
And I can start creating classes inside my packages (即在 edu.gatech 上點右鍵, 然後 New-> Class). So here, I can use the contextual menu, select New>Class, and I'll get another wizard that will allow me to specify the name of the class. I'm not very creative here, so I'm just going to call it Hello World. There's 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 button, the Finish button, we, we get the class. So we also get template code for the class, as you can see here, so we 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. And something you can note is that as we are taping, Eclipse gives us a auto complete suggestions, which is very helpful. For example, in case you don't remember the exact syntax, or the method, or you don't remember the parameters of the method. Which is, you know, often the case especially where you work with large libraries. So having that feature can really, really help you. So now if we want to run our code we can either click on the button up here,

or we can right-click in the Call window and select Run As Java Application. (注意不用 compile, 可以直接點 run, 即 javac Solutions.java 和 java Solutions 這兩步是一步完成的)And if we do that, Eclipse will run our tool, and it will create, as you can see here, a Consolvio that basically contains the textual output of my program. And as expected, the output is Hello World.

7. So now that we have run our program, let's see what happens exactly when you run a program within Eclipse. And to do that I'm going to use the menu over here which is the Run menu and I'm going to select Run Configurations,

and this brings up a windows where you can change or 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. And this is where you can configure the different parameters for your execution. For example, you can select the main class. So here it's, obviously, edo.gettech.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, Classpath and other environmental options. So let's now try to pass some arguments to our program. So for example here, I am just going to write George as a possible parameter.



I say Apply so that modify the configuration and if i run the program of course, the output is not changing because my program does not use the argument but. Let's see if we do use the argument, what happens. So I'm going to slide him with the final program so that now, instead of printing hello world, he will print hello followed by the first argument that I will pass to the program. And if I do that, and I go and I run the program, what I get is exactly what I was expecting, which is Hello George.

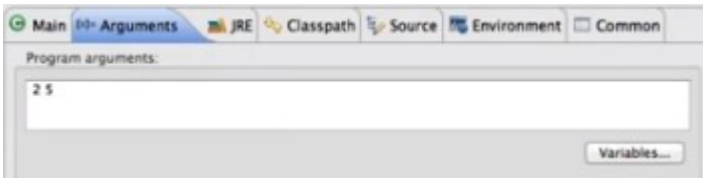
```
public static void main(String[] args) {
      System.out.println("Helo " + args[0]);

}
```

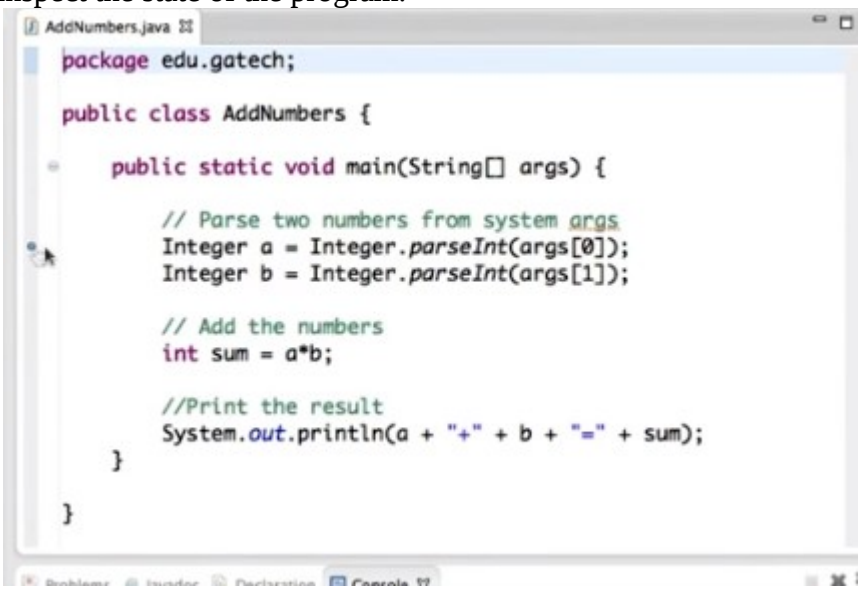
以上代碼輸出的是: Helo George. 所以上面那個框中輸入的 parameter(如 George)即 args[0]的值, 注意 args 數組是 main 函數的參數. 這裡的實參 George 應該默認就是個 String, 完整的寫法為"George". 後面的例子也證明了這點(輸入為 123 時, 要在 main 函數中 Integer a = Integer.*parseInt*(args[0]))
)

So this is the way in which you can pass arguments to your execution, which is something that might come in handy for some other projects. When you need to run some code with an argument.

8. Now let's look at how we can do debugging within Eclipse. I created a new file called AddNumbers which I'm showing here. It takes two numbers, parses them into integers, adds them and prints the sum, supposedly, of the two numbers. Now we 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 (兩個 arguments 時, 用空格隔開).



So now let's run our program and see what happens. And the result says that 2 plus 5 is equal to 10, which is not exactly correct. So we need to debug our program. We need to figure out what's wrong with the program, why the wrong result was, produced. So we're going to add a break point here by double-clicking here on the side of the code. And the break point is basically a place where I'm telling my debugger to stop during the execution (break point 的作用就是 debug 開始時, 就直接跳到那一行) because I want to inspect the state of the program.



So to start debugging, we select Debug as Java Application from the Context menu (即在代碼窗口中點右鍵, 然後 Debug as->1 Java Application), similar to what we were doing for running the program. And as you can see, this asks us whether we want to pass to the debug perspective, which is a, a perspective specifically designed for debugging. We say yes. And as you see here, it shows us, it's like a different, set of views, so we can see the code down here with an indication of where the execution is. And of

course the execution stopped at the break point, which is exactly where we told the debugger to stop. So let's look at some of the other views in this perspective. The view here on the right-hand side, for example, shows the variables in scope and the break points that are currently active for the debugging session. This is where the editorial is at. The outline of the program and the console at the bottom. So now let's execute one line by clicking on the Step Over button here at the top,

and this will execute the line that is currently highlighted and therefore it will move to the next line. And as you can see, one nice feature is that if I move the mouse over a variable, I can see the value of the variable. And the same thing I can do if I look at the variables windows here on the right (即右上角的窗口). If I click it, it will tell me what is the value of the variable, and in case of more complex variables you can even expand it and get more details. So now let's step over another line by clicking again this button, and as you can see now we get to the line that is actually performing the sum, supposedly, so now let's do the same thing that we did before, and let's mouse over b, and we can see that the value of b is five, as expected. So now let's step over this line as well, and execute the actual sum. And doing the mouseover thing, we can see that the value of sum is ten, which is not right, of course. In fact, if we check a gain we can see that value of A is two. The value of B is five and therefore it's clear that there's something wrong going on here, and at this point we can notice that here we are doing multiplication instead of addition. And therefore that's what the error is. And this is clearly a very simple case. Right? A case in which probably you just needed to look at the code and you didn't need the debugger. But you probably got the idea right? So this can be extremely useful when you're debugging, when you're studying more complex programs. If you want to stop the debugger because you're done with your debugging session as in this case, you can either click here 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 (從 Debugger 出來後, 頁面的佈局跟进去前會不一樣, that's OK). So, in this case, we're going to click here just to show what happens. And what happens is that, you know, the execution will just continue until the program exits. So now let's say that we want to fix this problem that we just discovered. So we replace the multiplication with an addition, we save the program, and we execute the program again by clicking on this button. And at this point, unsurprisingly, we get the right result as shown in the console.

From online:

A:
In the Eclipse Helios Java Package Explorer, I see the Java class icons display a small question mark to the right of the 'J', something like [J?].

B:
It means the class is not yet added to the repository.
If your project was checked-out (most probably a CVS project) and you added a new class file, it will have the ? Icon.
For other CVS Label Decorations, check http://help.eclipse.org/help33/index.jsp?topic=/org.eclipse.platform.doc.user/reference/ref-cvs-decorations.htm

A:
Aw, yes! Thanks, this answered my question. I've committed the package and class to CVS and the question marks are gone.

我的 notes:
若有一些 java 文件在同一個文件夾中, 且它們在同一個 package 中, 則在 Eclipse 中只需 run 那一個有 main 函數的文件即可, 其餘文件會自動被編譯. 但若在我的 terminal 中弄的話, 好像要先 javac 所有文件, 再 java 有 main 函數的文件.

From 6300 project group:
If you put files in the root directory of your project (where Eclipse makes the ".project" file), you can use them without anypath