

A large, two-story, light-colored building with a red-tiled roof and a central tower, surrounded by green grass and trees under a clear blue sky.

MAHARISHI UNIVERSITY of MANAGEMENT

Engaging the Managing Intelligence of Nature

Computer Science Department

**CS390 Fundamental Programming
Practices (FPP)
Professor Paul Corazza**



© 2016 Maharishi University of Management, Fairfield, Iowa

All rights reserved. No part of this slide presentation may be reproduced or utilized in any form or by any means, electronic or mechanical, including photocopying or recording, or by any information storage and retrieval system, without permission in writing from Maharishi University of Management.

Lecture 1: Introduction to Java And the Eclipse Development Environment

Pulling the Arrow Back to Hit the Target

Wholeness of the Lesson

Java is an object-oriented highly portable programming language that arose as an easy alternative to the once dominant, but error-prone, C++ language. Eclipse is one of many open source, powerful but easy-to-use integrated development environments for use with Java and related technologies. Working from deeper levels of intelligence allows one to accomplish more with fewer mistakes and less effort.

About Java

- ***Brief History.*** The Java language began as a language for programming electronic devices, though the original project was never completed. Its creator was James Gosling, of Sun Microsystems. The language was developed privately starting in 1991, and was made publicly available in 1994. In 2009, Oracle bought the rights to Java from Sun Microsystems.
- ***Interpreted Language.*** When you "compile" Java code, the result is not executable binary code, targeted to a particular machine; instead, the result is **bytecode**, having a portable intermediate code format. The bytecode is then executed by running an **interpreter**, called the **Java Virtual Machine (JVM)**. This approach makes Java code highly portable; Java will run on any platform for which a JVM has been created.

About Java

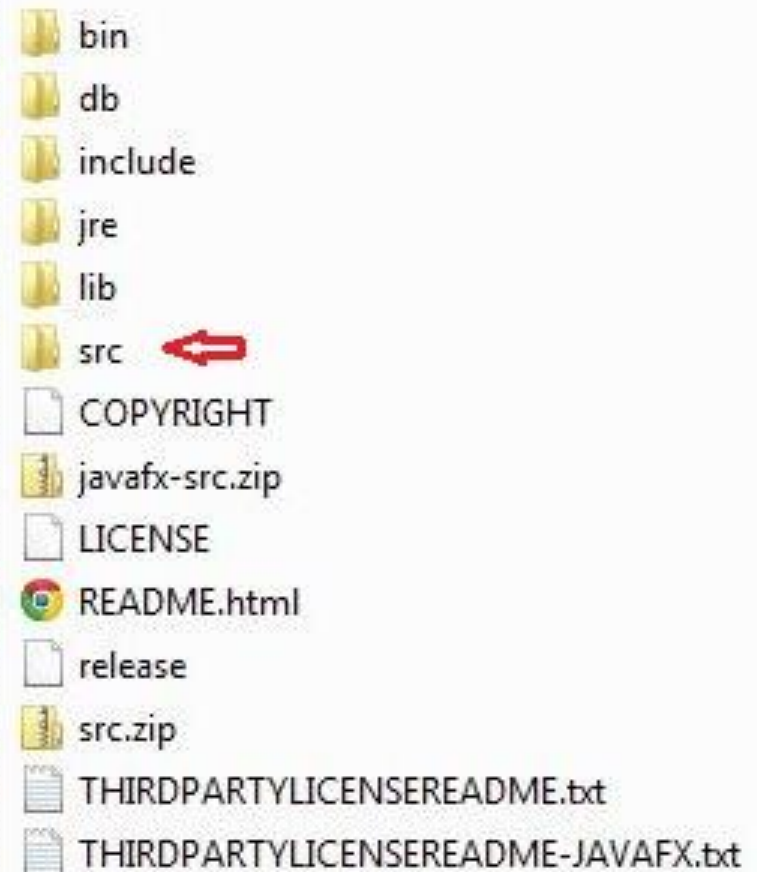
- ***JIT.*** Interpreters run much more slowly than native binary executable code. Java performance is boosted considerably in modern-day JVMs because of the inclusion of a **just-in-time compiler**; this feature compiles frequently occurring bytecode sequences into native binary code and caches the results, re-using the cache as needed. The result is that Java runs almost as fast as C++ in typical cases.
- ***No Pointers.*** Unlike C and C++, Java does not make use of pointers; developers are not required to manage the memory usage of their applications. Memory management is handled automatically in Java by means of a **garbage collector**. The garbage collector is run by the JVM at different times during program execution to return to memory all unused object references.

About Java

- ***Java Is an OOP Language.*** Java is an OO programming language, and succeeds in this better than C++, which (for the sake of backwards compatibility with C) supports a non-OO programming style (in C++, OO programming is "optional").
- ***Convenient Libraries.*** Java provides convenient libraries for handling files and streams, networking, http, gui development, and database connection. Compared to languages like C and C++, the increased ease of use is significant.
- ***Good Security Model.*** Java has a relatively good security model to support use over networks and distributed environments. Though security holes are still found sometimes, these are rare and quickly patched. The fuss about *applet security holes* was exaggerated; applets are far safer than the Windows executables that are downloaded to (and that infect) Windows machines all the time without much discussion about the potential dangers.

The JDK Distribution

- It's a good idea to unzip src.zip and place in a source folder (called “src” in the screen capture above) – this will allow you to see how Oracle has implemented its library classes. You can also unzip javafx-src.zip to look at JavaFX source code.
- Oracle provides online documentation of all the Java library classes. For jdk1.8, the link is <https://docs.oracle.com/javase/8/docs/api/>



Working with the JDK

TIP: If, to your PATH environment variable, you add the path to the `javac` compiler and to `java.exe`, compiling and running java programs from the console is much easier. Typical path to these executables (the first is for the 64 bit distribution, the second for the 32 bit distribution):

```
C:\Program Files\Java\jdk1.8.0_45\bin
```

```
C:\Program Files (x86)\Java\jdk1.8.0_45\bin
```

To **change** the system environment variables, follow these steps: In the left panel of windows explorer, right-click Computer and click Properties. In the System Properties **window**, click Advanced System Settings > Environment Variables

Working with the JDK

- Create a “hello world” example by typing the following lines of code in WordPad (or Notepad), and save in a convenient location as Hello.java.

```
class Hello {  
    public static void main(String[] args) {  
        System.out.println("Hello World");  
    }  
}
```

- Then compile by typing the following command in the console:

```
javac Hello.java
```

- If a compiler error occurs, the compiler attempts to indicate clearly where the mistake occurred.
- When compilation is successful, a new file `Hello.class` has been created. Execute the code with the command

```
java Hello
```

Main Point

Java is an object-oriented programming language that is easier to use, less error-prone, and more portable (and nowadays, more popular) than C++. Java code is compiled to *bytecode*, which can then be converted to native code on a target platform, by way of a JVM interpreter. The inefficiency of an interpreter has largely been eliminated through the use of the *just-in-time compiler*, which compiles frequently occurring bytecode sequences to native code and caches these for further use, at runtime. Any language has the power to reveal or obscure the truth – as Maharishi says in SCI, "it is the power of speech that it can bind the boundless."

Integrated Development Environments

- A good IDE supports compiling, running, and debugging code with tools that are integrated and typically easy to use. For a large Java project, an IDE is indispensable.
- Good choices of IDE are NetBeans, IBM Rational Application Developer (formerly WebSphere Application Developer), Borland's JBuilder, JetBrains' IntelliJ
- Another excellent choice, which has become an industry standard, is the open-source IDE Eclipse, written entirely in Java. Among IDEs for Java, in recent years, Eclipse is the most widely used. We will use Eclipse in this course.

The Eclipse IDE

- Getting started. To use JSE 8, you need Eclipse Luna (or later); earlier versions of Eclipse do not support JSE 8.
- Features of the IDE [demo]
- Try creating Hello in Eclipse, and running

Viewing Source Code

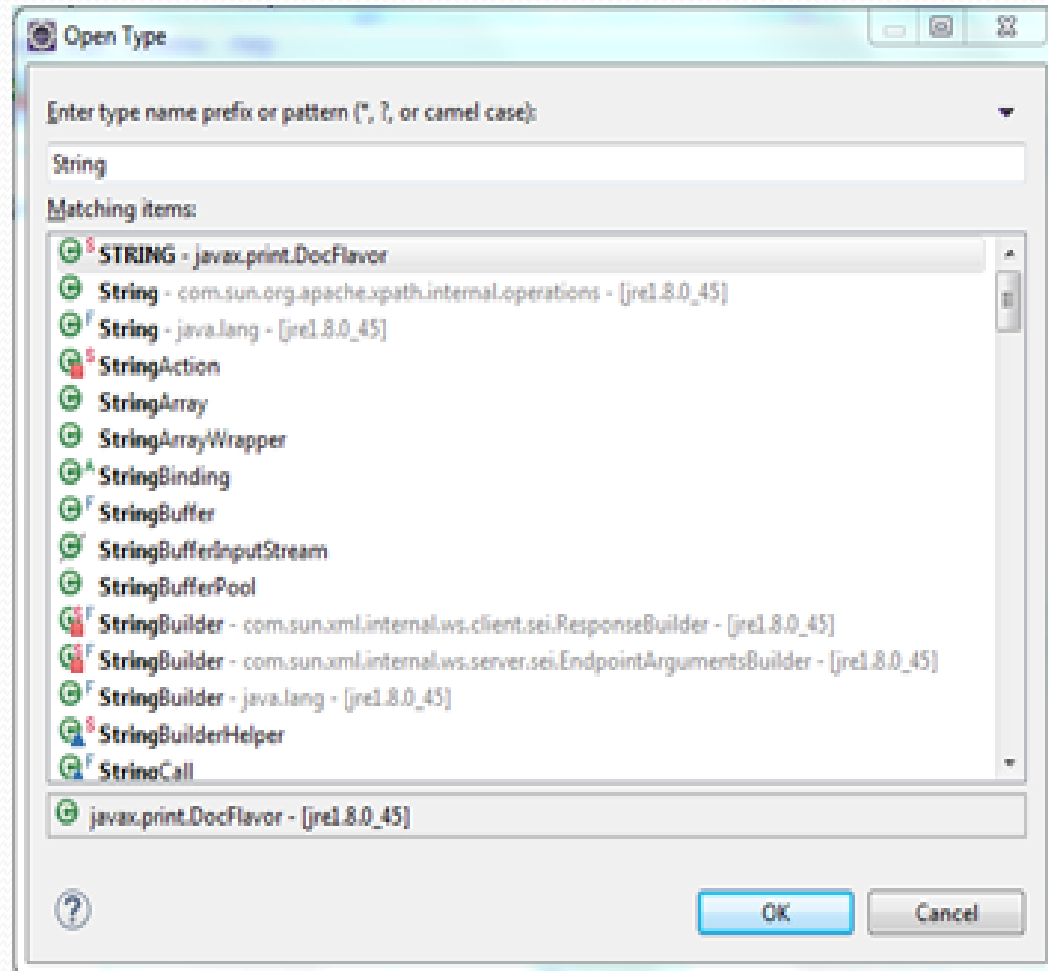
Allows you to see how Java has implemented their library code.

Steps:

1. Open any project in Eclipse and type
Ctrl Shift T

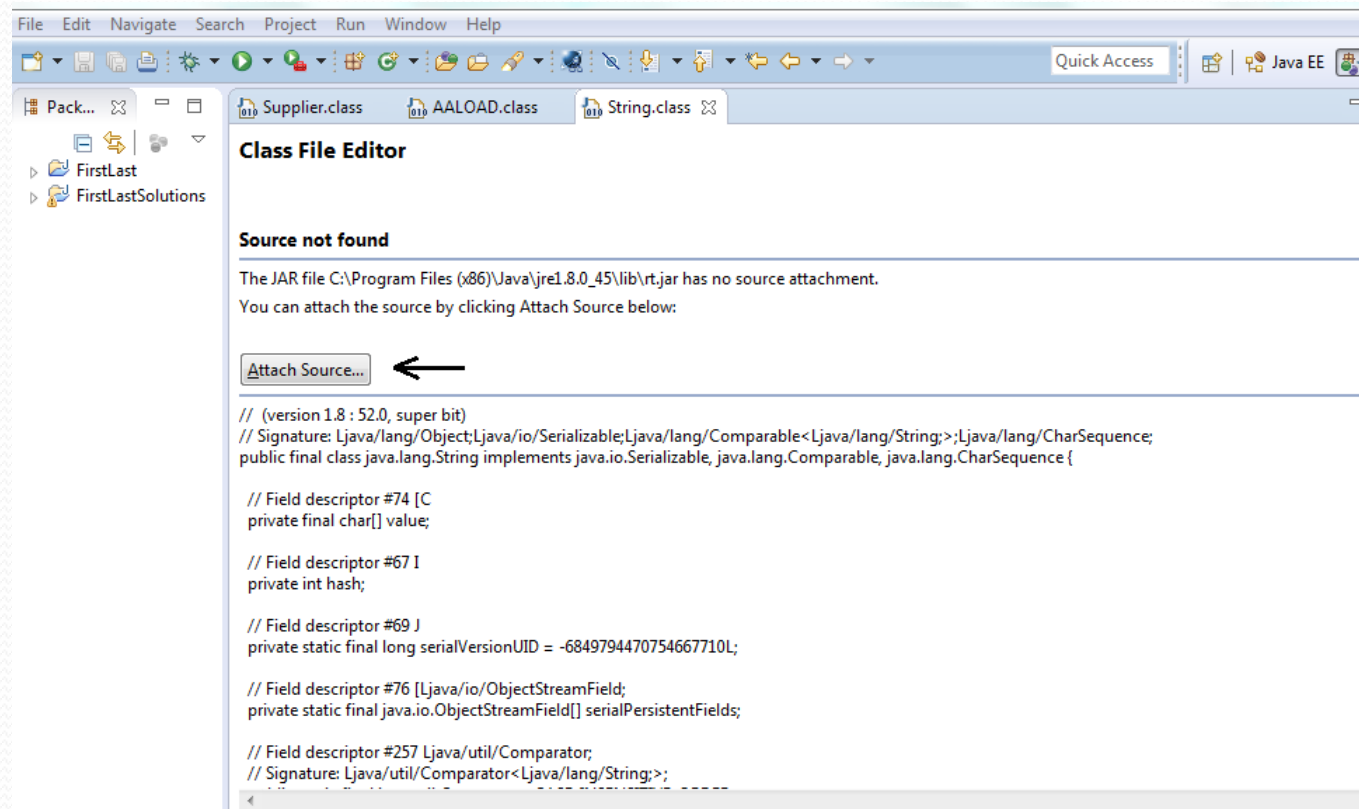
Brings up this screen:

[Note: “String” was typed into search window]



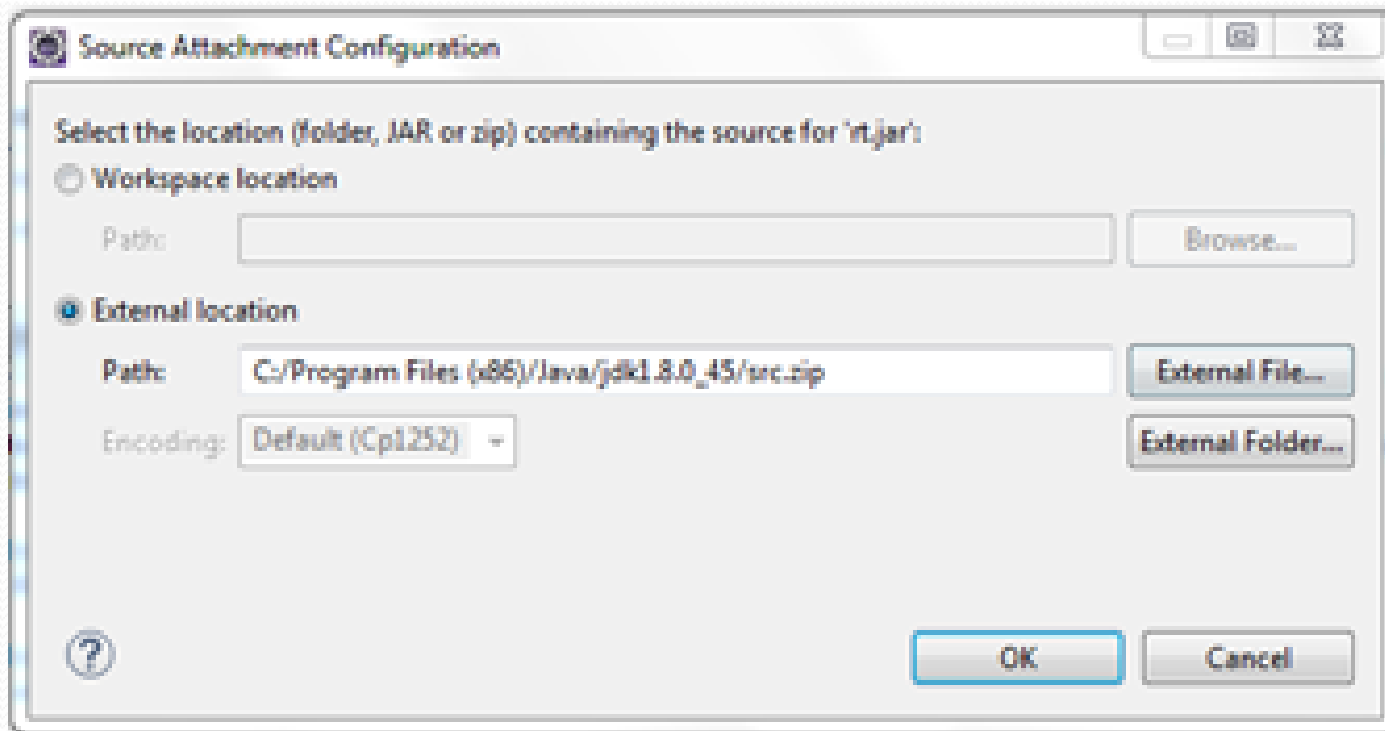
Viewing Source Code 2

2. Clicking OK lets you see *partial* info of the class file (of String in this case)



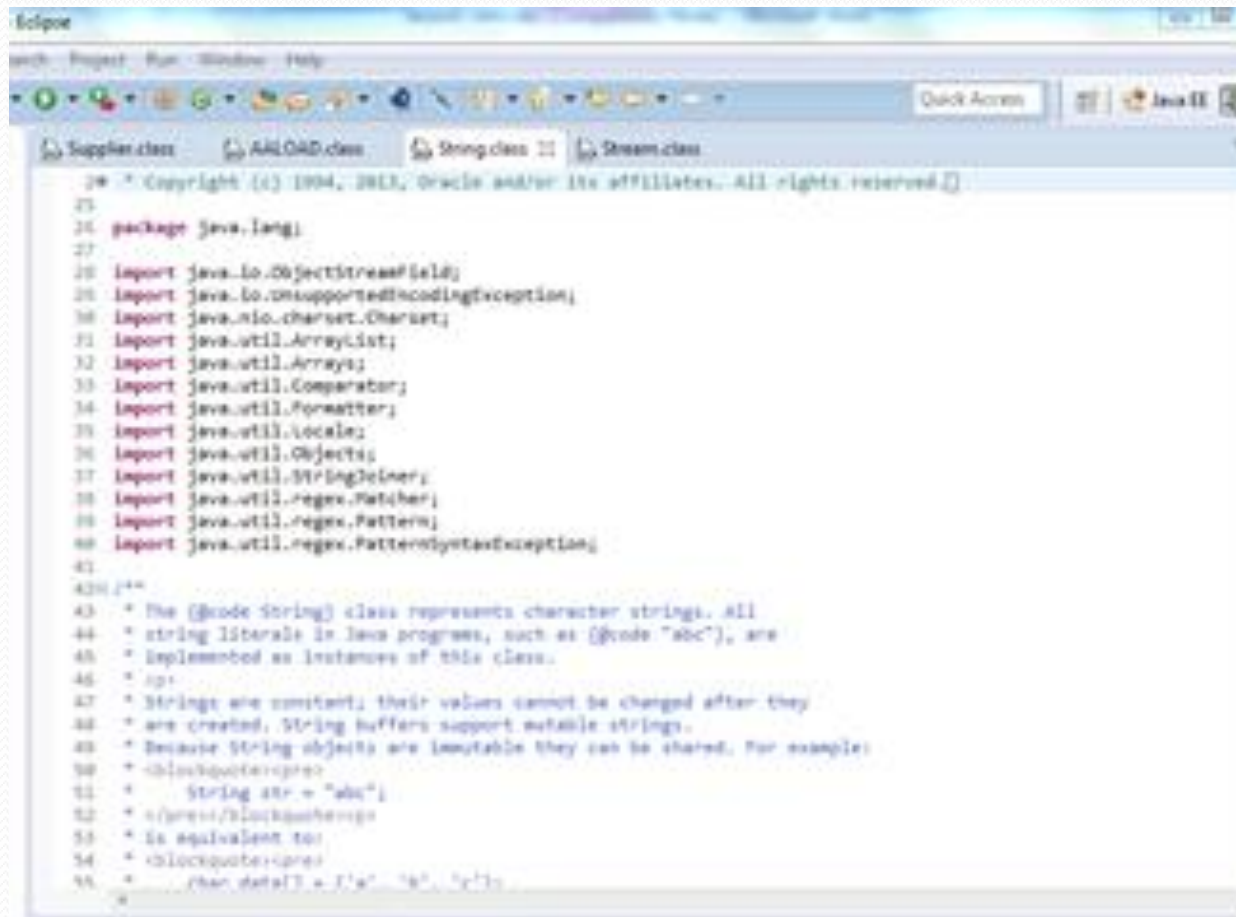
Viewing Source Code 3

3. Click Attach Source and navigate to src.zip in your jdk distribution (do only once – next time just do steps 1 and 2)



Viewing Source Code 4

You can now see the source code for your selected type



The screenshot shows the Eclipse IDE with the 'String.class' tab selected. The source code is displayed in the editor, showing the package declaration, imports, and the class documentation. The code is as follows:

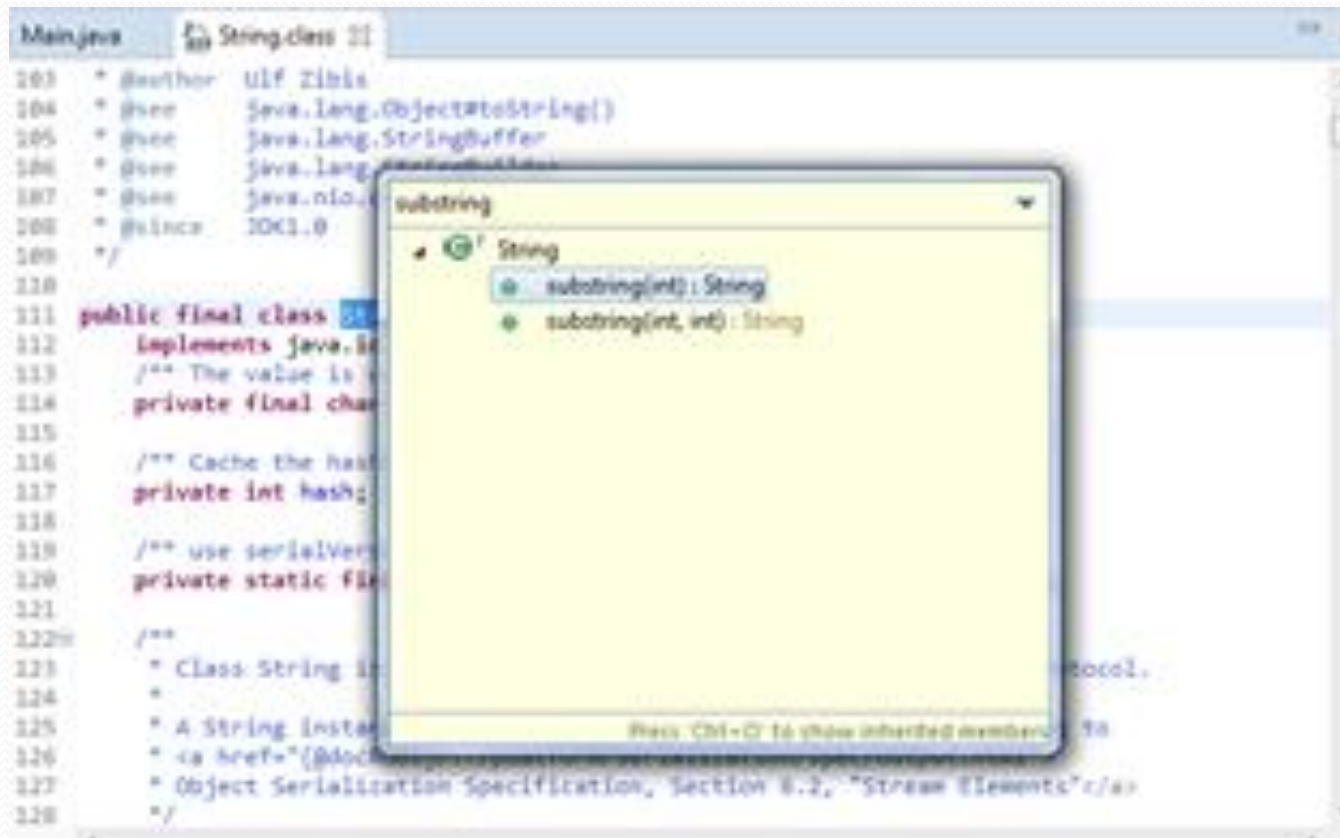
```
24  * Copyright (c) 1994, 2013, Oracle and/or its affiliates. All rights reserved.
25
26  package java.lang;
27
28  import java.io.ObjectStreamField;
29  import java.io.UnsupportedEncodingException;
30  import java.nio.charset.Charset;
31  import java.util.ArrayList;
32  import java.util.Arrays;
33  import java.util.Comparator;
34  import java.util.Formatter;
35  import java.util.Locale;
36  import java.util.Objects;
37  import java.util.StringJoiner;
38  import java.util.regex.Matcher;
39  import java.util.regex.Pattern;
40  import java.util.regex.PatternSyntaxException;
41
42  /**
43   * The {@code String} class represents character strings. All
44   * string literals in Java programs, such as "abc", are
45   * implemented as instances of this class.
46   *
47   * Strings are immutable: their values cannot be changed after they
48   * are created. String buffers support mutable strings.
49   * Because String objects are immutable they can be shared. For example:
50   *
51   * 

```
String str = "abc";
52 // ...
53 // Equivalent to:
54 // char data[] = {'a', 'b', 'c'};
```


55   */
```

Viewing Source Code 5

- To locate a method inside the class (say, “substring()” in String), type Ctrl-O and type in search string

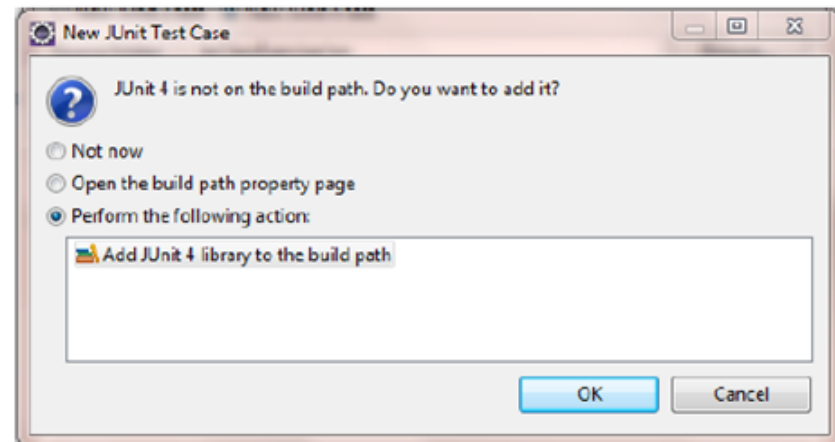


Including Unit Testing in Your Work Environment

- A quick way to test your code as you develop is to run it from the main method as we have been doing
- A better way that is more reusable and useful for larger-scale team development is to have a parallel test project and use JUnit

Steps to set up JUnit

- Right click the default package where your Hello.java class is, and create a JUnit Test Case TestHello.java by clicking New→JUnit Test Case. (After we introduce the package concept, we will put the tests in a separate package.)
- Name it TestHello and click finish. You will see the following popup window.



- Select “Perform the following action” → Add JUnit 4 library to the build path → OK
- Then you will see that JUnit 4 has been added to the Java Build Path by right clicking your project name → properties → Java Build Path.

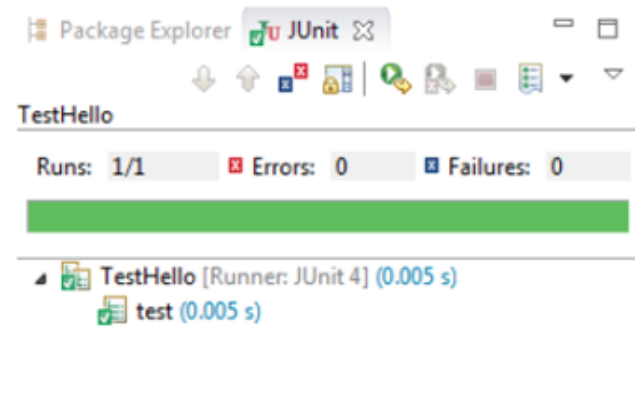
Including Unit Testing in Your Work Environment

- Create a testHello method in TestHello.java (see below) and remove the method test() that has been provided by default:

```
@Test
public void testHello() {
    assertEquals("Hello", Hello.hello());
}
```

If you have compiler error, you can hover over to the workspace where you have error and Eclipse will give you hints of what to do. (In this case, Add import 'org.junit.Assert.assertEquals'.)

- Run your JUnit Test Case by right clicking the empty space of the file → Run As → JUnit Test. If you see the result like below, it means you have successfully created your first unit test. 😊



Running Applets

- Applets are small Java programs designed to run as a window inside a web browser.
- Java was originally designed as an Applet-creating language.
- Misleading information has resulted in an unjustified worry about security of applets. At this time (2016), Firefox is the only browser that supports running of applets.
- An alternative is to run applets using the utility `appletviewer.exe` (located in the `bin` directory of the `jdk` distribution).

Running the Demo Applet


Use bfs_solution.html in Demo directory of Lesson 1 (folder lesson9_stackqueue must also be present)

1. Run in Firefox

- Program Files > Java > Configure Java > Security
- Make sure “High” and “Enable Java content” are selected
- Edit Site List > paste full path to the html file in format like this:
file:///D:/courses/FPP\FPP-March2016/ectureSlides/Lesson1-Intro
/demos/bfs_solution.html
- Ignore the warnings.
- Click OK
- Drag the html file into Firefox
- Click “Allow and Remember” when asked
- Click “Activate Java” if asked
- If you get a message “Your version of Java is out of date”, click Update – but do not let it remove previous versions of Java.
- If necessary, drag the html file with the applet into Firefox once again
- It will ask “Do you want to run this application”? Click Run
- The applet should now run

Running the Demo Applet 2

Run the applet using appletviewer

- Place bfs_solution.html, along with lesson9_stackqueue, somewhere on your hard drive.
- Open a command window (click  and type cmd.exe and press Enter)
- CD to this directory
- Type in the window appletviewer bfs_solution.html
- The applet should run.

Main Point

Eclipse is a leading, open-source, 100% Java, integrated development environment, which provides excellent support for editing, compiling, running, and debugging Java applications. By analogy, to create a good life, we need to handle inner life and at the same time, structure a life-supporting environment – the goal is to live 200% of life.

Summary

- After the JDK has been installed, it is possible to use `javac.exe` and `java.exe`, located in the `bin` directory of the JDK distribution, to compile and run Java code.
- Developing code is much easier using an IDE – the most widely used IDE for Java is Eclipse.
- Eclipse allows you to edit, compile, debug, and run Java code in an integrated environment. It also lets you view Java library code (`ctrl-shift-T`, and `attach source`).
- JUnit is one of the most popular Java add-ons for unit-testing Java code, and is supported in the Eclipse environment.
- Java applets can be run from the commandline using `appletviewer` (in the `bin` directory) or from within Firefox once security settings have been set

Connecting the Parts of Knowledge With the Wholeness of Knowledge

1. Using Java, highly functional applications can be built more quickly and with fewer mistakes than is typically possible using C or C++.
 2. To optimize the use of Java's features, IDE's such as Eclipse ease the work of the developer by handling in the background many routine tasks.
-
3. **Transcendental Consciousness:** To be successful, action must be based on the field of pure intelligence, which is located at the source of thought.
 4. **Wholeness moving within itself:** In Unity Consciousness, the pure intelligence located in TC is found pervading all of creation, from gross to subtle.