

# Introduction

# Objectives

- Appreciate Java as a language,
- Write, compile and execute simple java program in command prompt and eclipse

# Table of Content

Java Language	Portable and platform independent
Simple Hello World in Java	JVM: perspectives
Environment to compile and execute	Interpreter vs JIT
Setup JDK	Error handling in java
JDK installation directory	Automatic garbage collection
Compilation & execution: command prompt	Classpath
JRE and bytecode	Commenting source code
Introducing Eclipse Helios 3.6 IDE	Some Guidelines for Doc comments
Running Different Workspace	javadoc
UI Overview	Nesting comments
Features of Java Language	Annotations

# Java Language

- Java was created by a team of members called "Green" led by James Arthur Gosling.
- When Java was created in 1991, it was originally called Oak.
- It is a free and open source software (FOSS) under GNU General Public License(GPL)
- First version of Java was released in 1995.
- Java is an Object oriented language, simple, portable, platform independent, robust and secure.
- We will be focusing on Java 1.6.

# Java Programming Language: Designers



Bill Joy

- BSD Unix guy from Berkeley
- founded Sun Microsystems (early 80s)



James Gosling

- early fame as the author of “Gosling Emacs” (killed by GNU)
- then onto Sun’s “NeWS” window system (killed by X)

# Flavors Of Java

- JSE
    - Java Standard Edition formerly known as J2SE.
    - This forms the core part of Java language.
  - JEE
    - Java Enterprise Edition formerly known as J2EE.
    - These are the set of packages that are used to develop distributed enterprise-scale applications.
    - These applications are deployed on JEE application servers.
  - JME
    - Java Micro Edition formerly known as J2ME.
- These are the set of packages used to develop application for mobile devices and embedded systems. Our focus

# Simple Hello World in Java

Hello.java

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

Special statement used to display data on console. 'println' causes the next print statement to be printed in the next line.

main() is a method from where program execution begins.

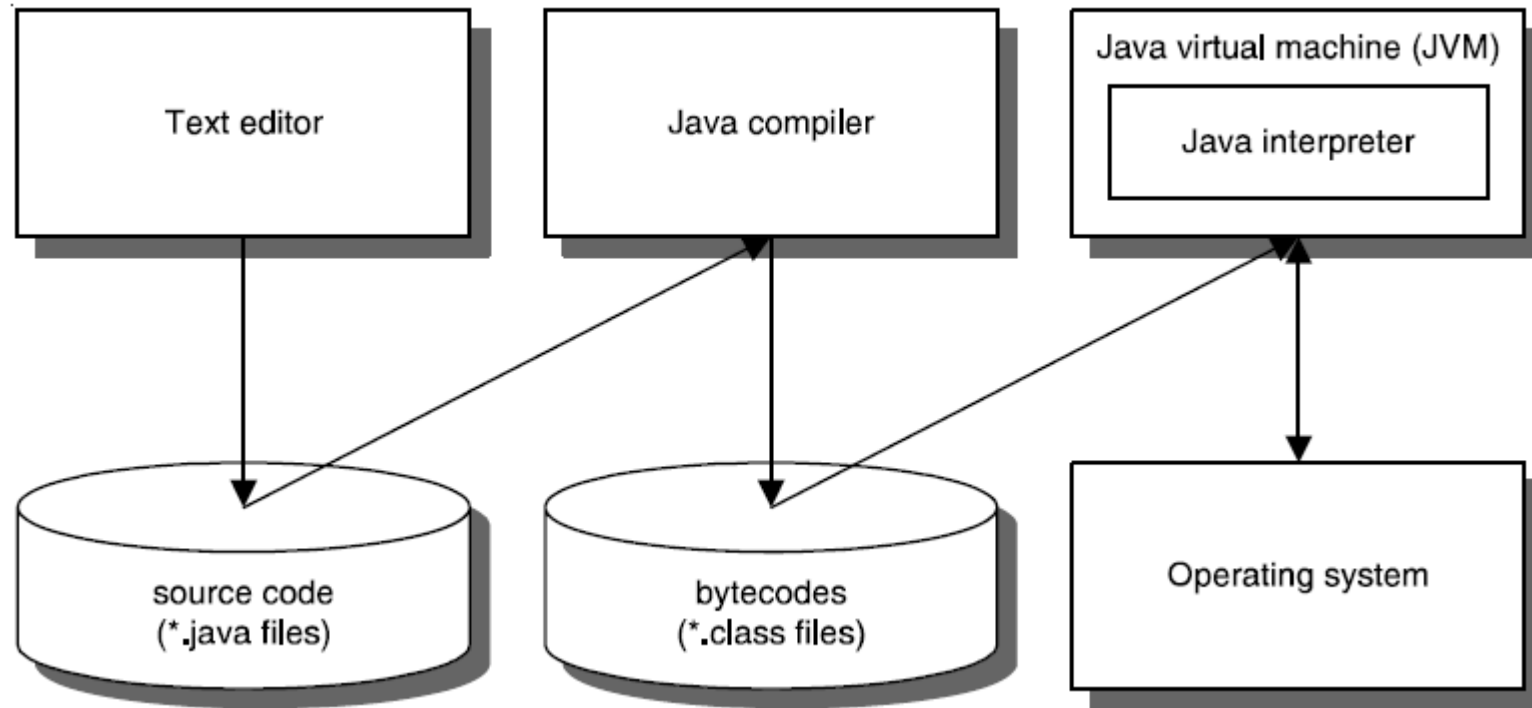
# Environment to compile and execute

- Compile java programs
  - From command prompt
  - Through an IDE (Integrated development environment)
    - Eclipse → Apache
    - NetBeans → Oracle SDN
    - JBuilder → Borland
    - Integrated Development Environment → IBM

 *What is an IDE?*



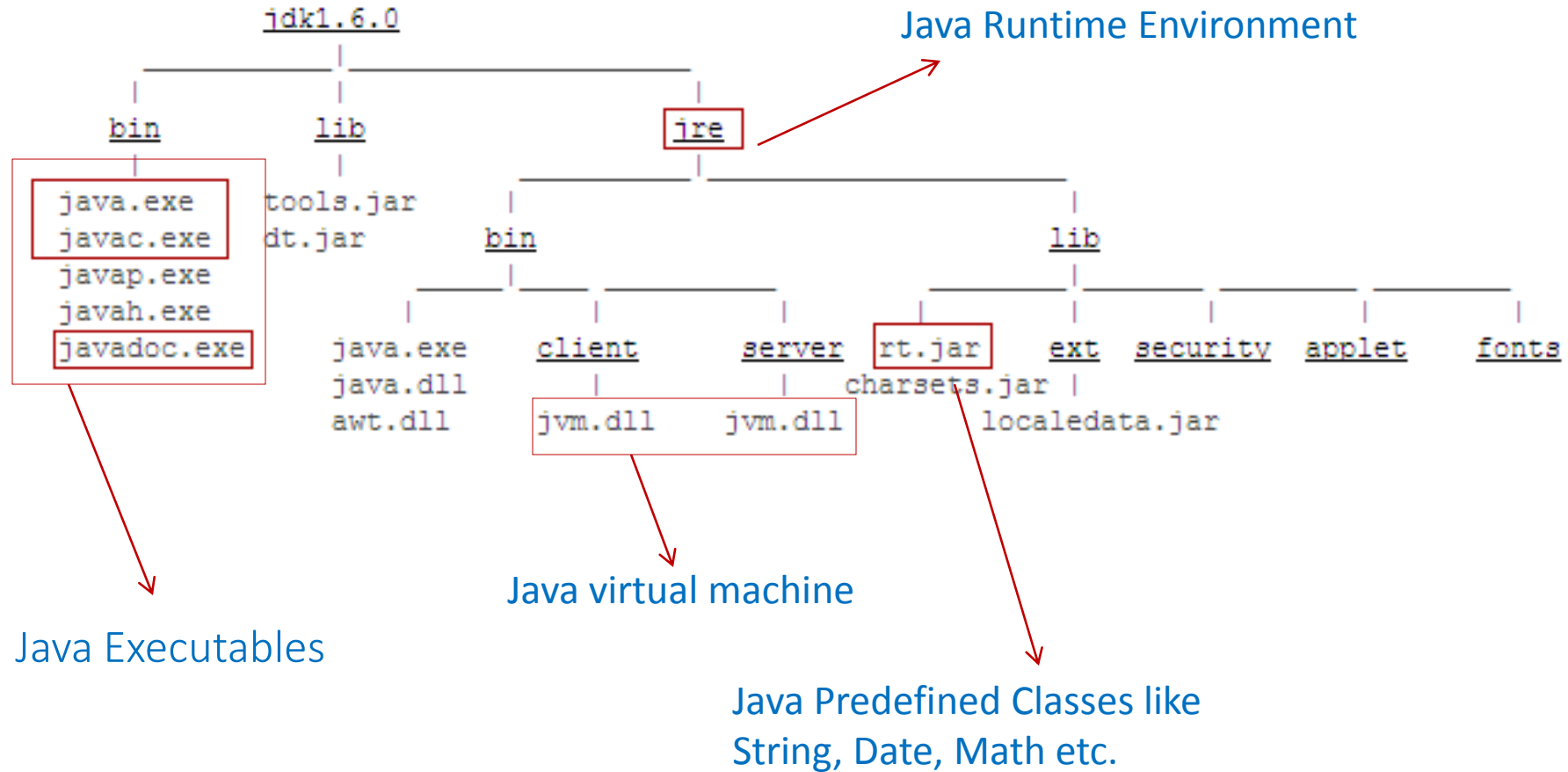
# Flow



# Setup JDK

- <http://java.com/en/download/index.jsp> or find appropriate link in <http://www.oracle.com/technetwork/indexes/downloads>
- Download Java based on the type of OS
  - Windows
  - Linux
  - Mac OS
  - Solaris
- Install JDK

# JDK installation directory

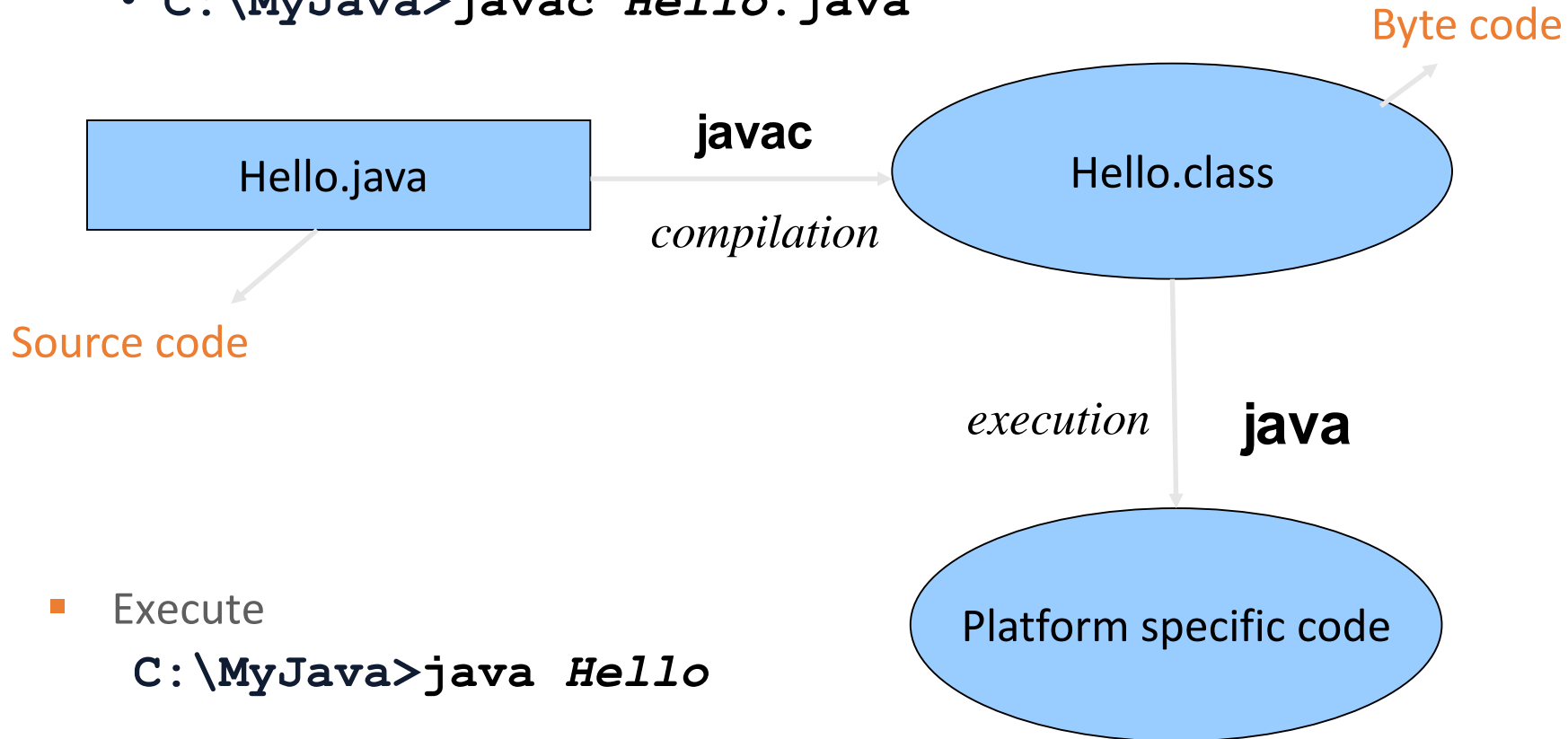


# Compilation & execution: command prompt

Save the file as Hello.java. A public class must be saved in the same name as class name.

- **Compile**

- `C:\MyJava>javac Hello.java`



- **Execute**

`C:\MyJava>java Hello`

# JRE and bytecode

- Java Bytecode is produced when Java programs are compiled.
- To execute Java program, JRE must be installed in the system
- JRE or Java Runtime environment contains
  - Java Virtual Machine
  - Standard class libraries (APIs)
  - Java Plug-in
  - Java Webstart
- JRE gets installed automatically when JDK is installed. JRE can be installed independent of JDK as well. This will mean that Java programs can be executed in that system.

# Introducing Eclipse Luna IDE

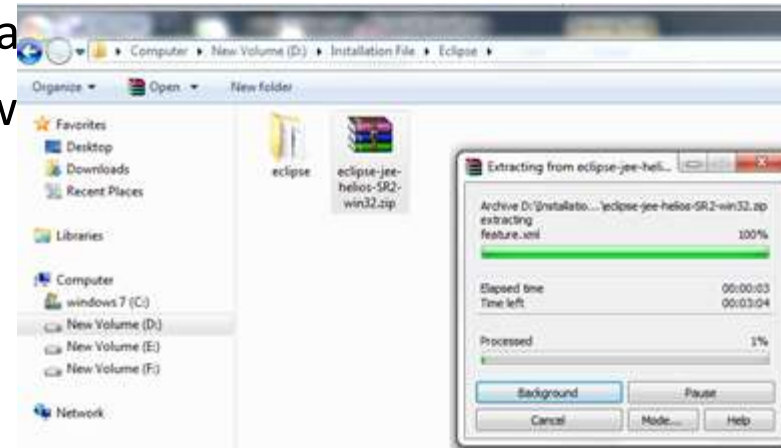
- IDE is where the Java programs are written, compiled and executed.
- Eclipse is an open source project
  - Launched in November 2001
  - Designed to help developers with specific development tasks
- GUI and non-GUI application development support
- Easy integration of tools
- Supported by multiple operating systems like Windows, Mac, Linux, Solaris, IBM AIX and HP-UX.

# Eclipse as an IDE

- Java Development Tooling (JDT) is used for building Java code
- Provides set of workbench plug-ins for manipulating Java code
  - Java projects, packages, classes, methods, ....
- Java compiler is built in
  - Used for compiling Java code
  - Creates errors (special markers of code) if compilation fails
- Numerous content types support
  - Java, HTML, C, XML, ...

# Activity: Installing Eclipse

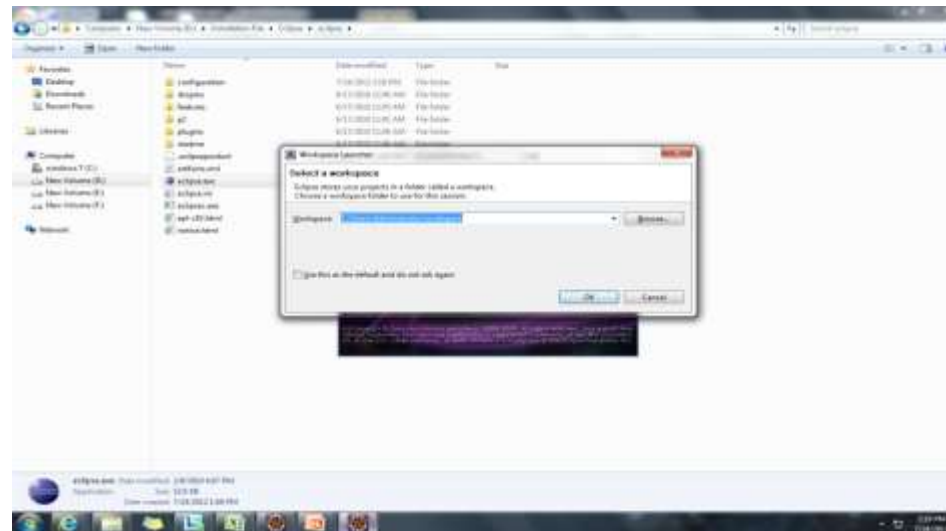
- Download Eclipse's Install Zip File from <http://www.eclipse.org>
- Click on the Download from the main page on <http://www.eclipse.org>
  - Choose the best site from which to download
  - Choose the latest build for download
  - Choose the platform for download a
  - Specify the location to save the dow
- Unzip the downloaded file to the directory of your choice



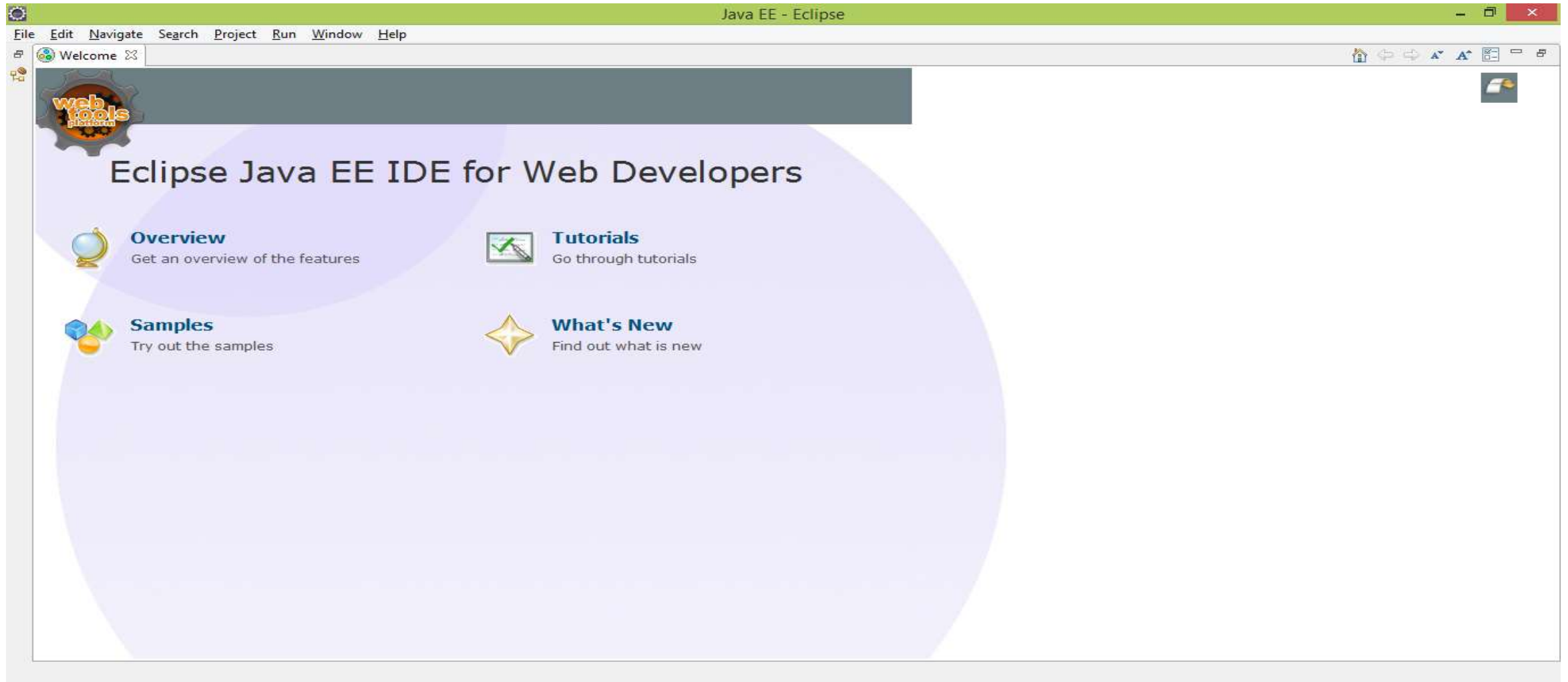


# Activity: Specifying Workspace

- By double-clicking on the eclipse.exe file, the workspace launcher will pop up.
- The workspace is a place where the user defined data – projects and resources such as folders and files are stored.
- Always when eclipse is launched, it prompts for a workspace location to open with.
- This prompt could be turned off or on as per the user's choice.



When Eclipse is run, a Welcome page opens



# Running Different Workspace

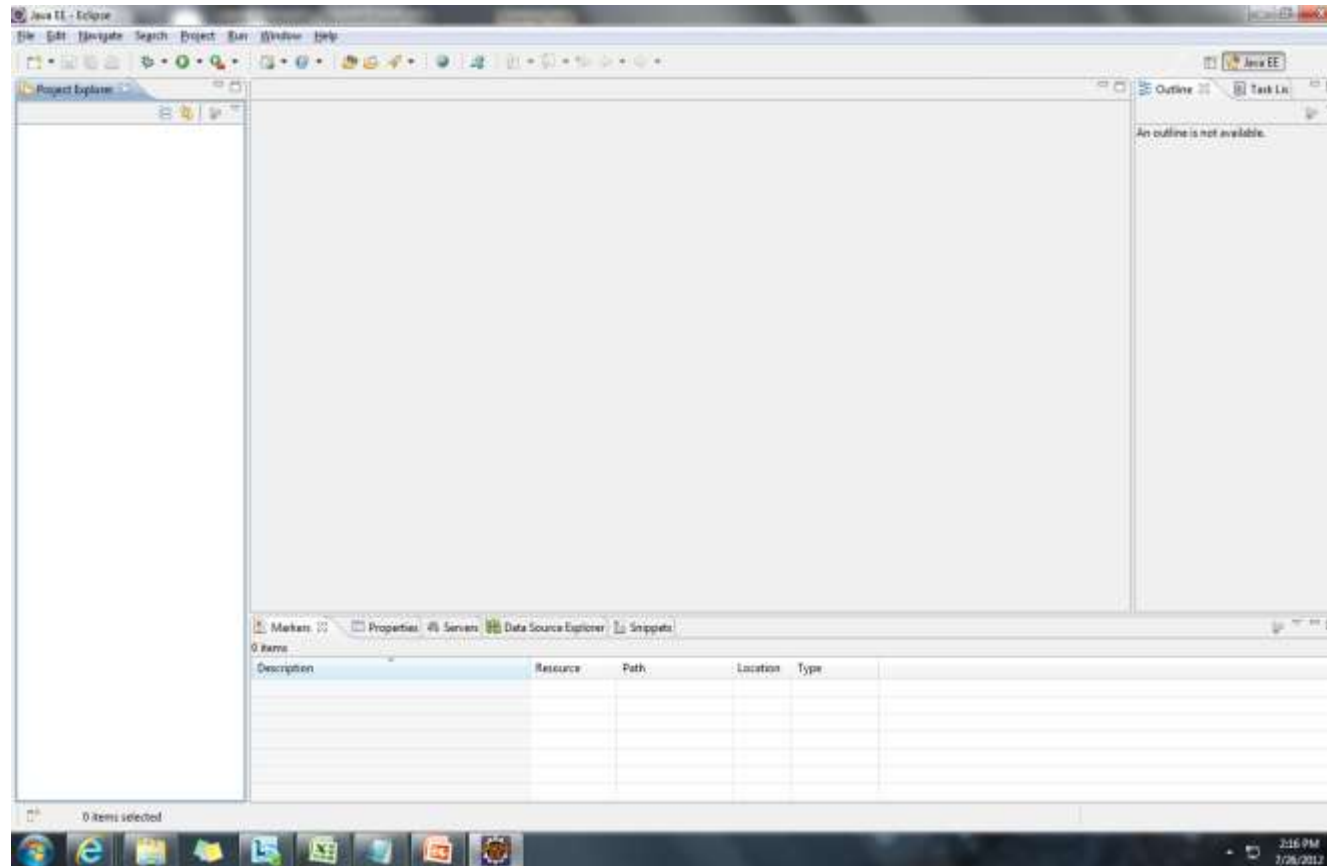
- It is possible to run different workspaces using command prompt
  - -data argument must be used with eclipse.exe
  - Workspace location must be specified
- Useful for grouping project specific data
- Multiple workspaces can run at the same time



*Please refer to Eclipse Shortcuts provided in the reference material while using Eclipse*

# UI Overview

Closing the Welcome page gets you to the Eclipse workbench user interface



# Activity : Create and Execute a Java project

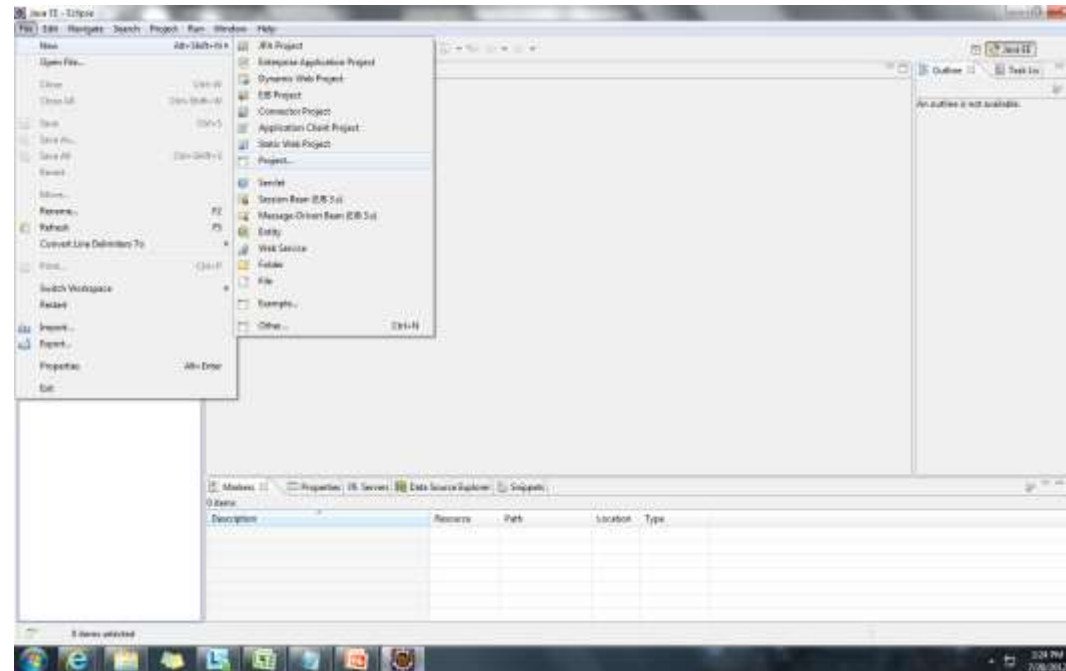
- Writing a simple HelloWorld program in Java using Eclipse.

Step 1: Start Eclipse.

Step 2: Create a new workspace called sample and select Ok.

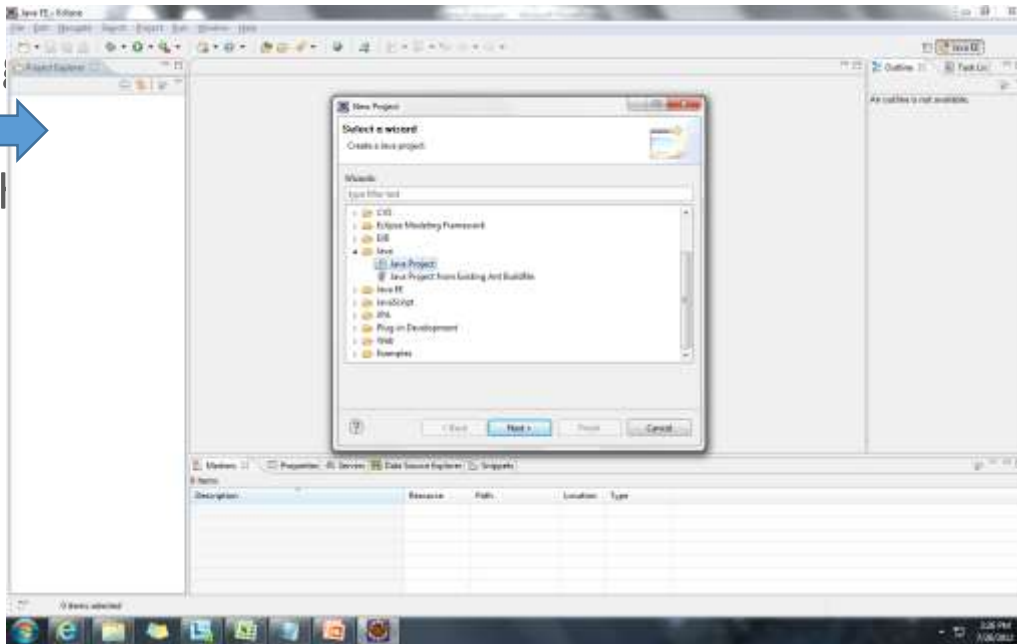
Step 3: Close the Welcome page

Step 4: File→New → Project

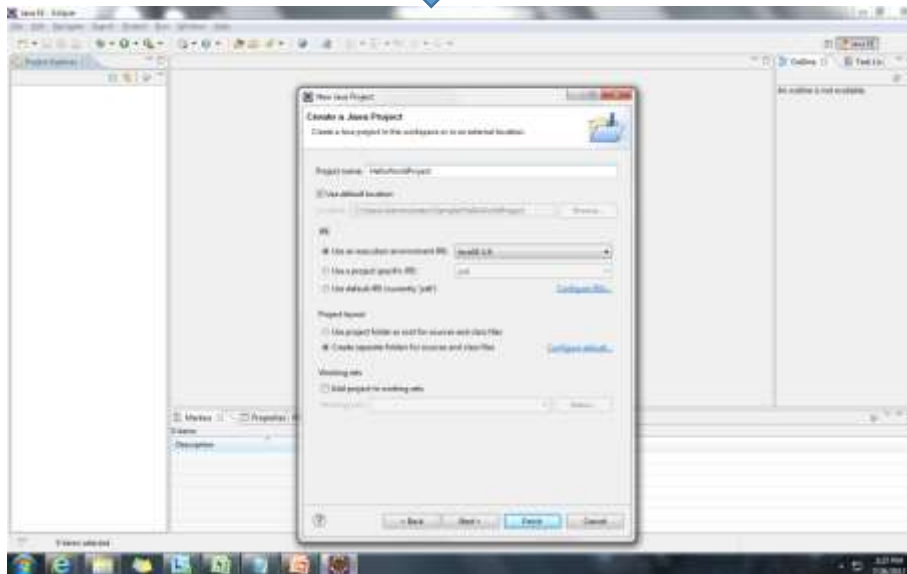


Step 5: Select "Java" in the category list.

Step 6: Select "Java Project" in the project list. Click "Next".



Step 7: Enter a project name into the Project name field, for example, "Hello World Project". Click "Finish".



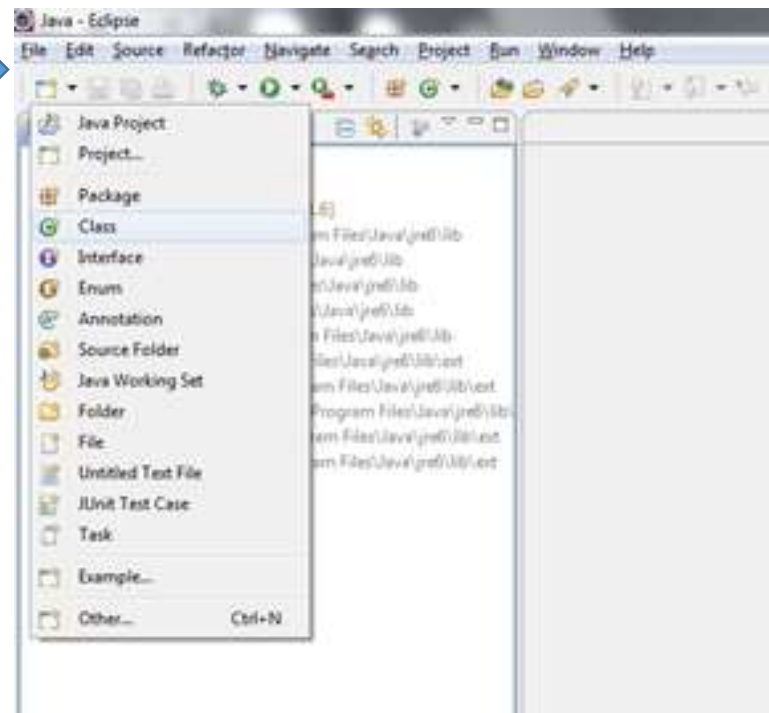
It will ask you if you want the Java perspective to open. Select Yes.



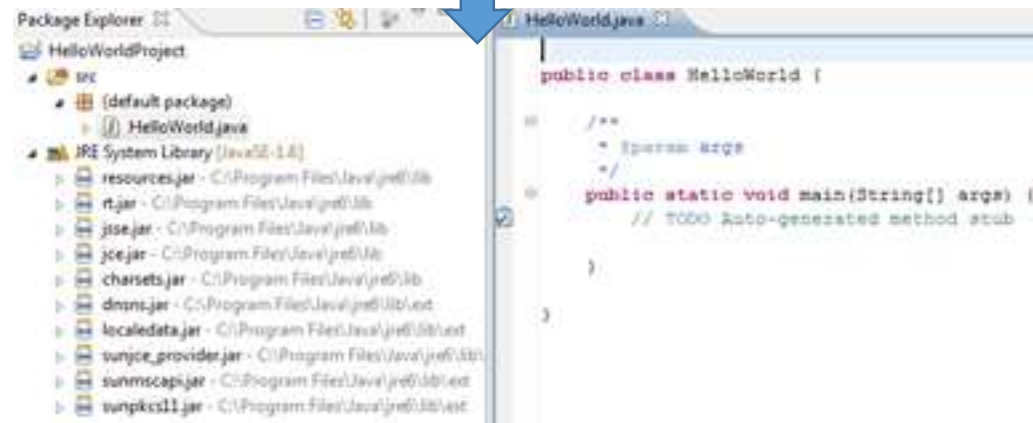
Step 8: Select New → Class option



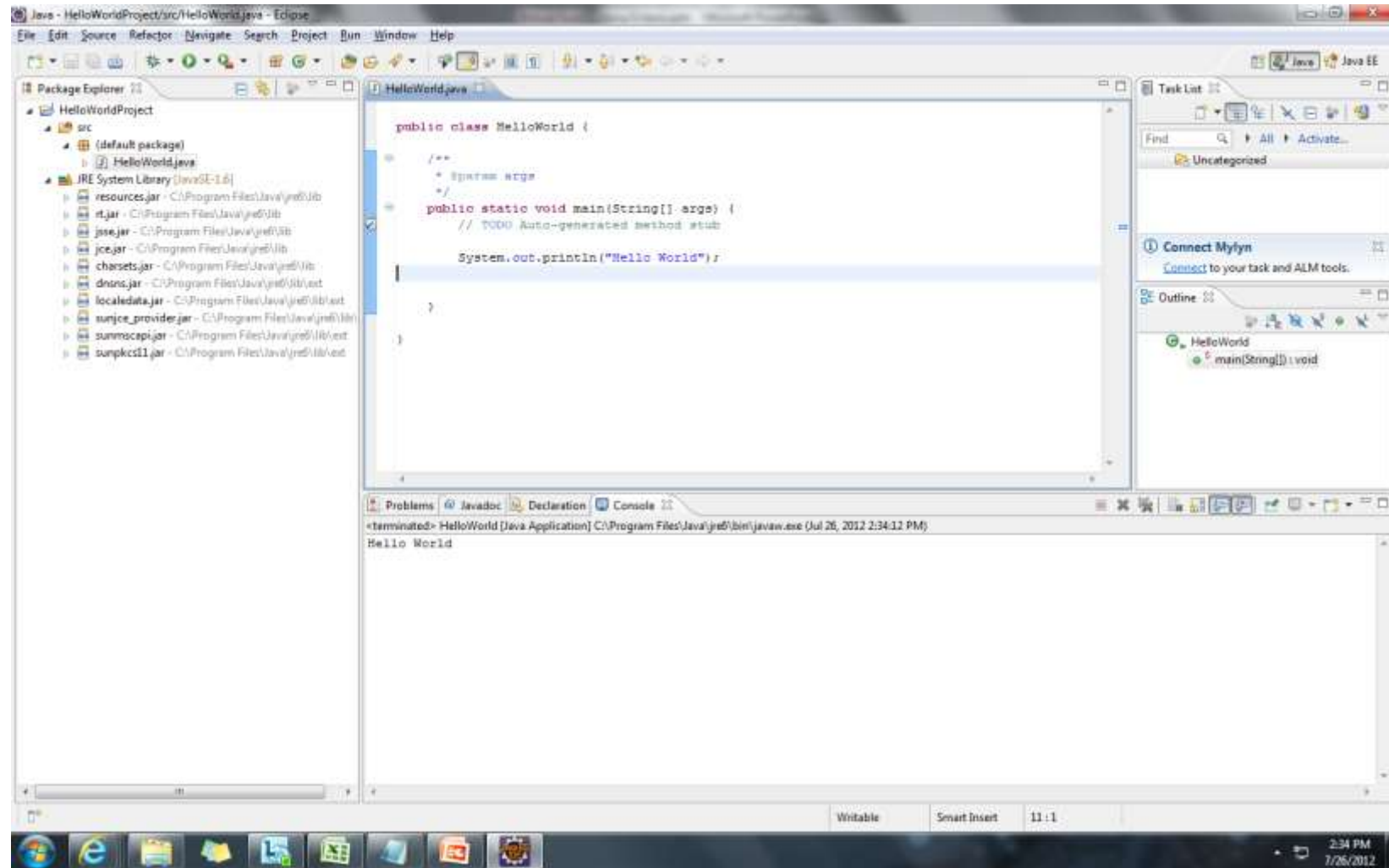
Step 9: Specify the class name "HelloWorld" and select "public static void main(String[] args)" check box. Click Finish.



Step 10: In the main method enter the following line.  
System.out.println("Hello World"); Save it.



Step 11: Select Run → Run.





# Exercise

- *Write a java program to display the following*

\* \* \*

\* \*

\*

- Use print and println statements.*
- The class file of this program should be automatically placed inside “Design” folder while compiling.*
- Display the version used for compiling.*

*( 15 mins)*

# Features of Java Language

- Simple
  - Object oriented language
- Portable and platform independent
- Robust
- Multithreaded
- Dynamic Linking
- Secure
- Performance

# Portable and platform independent

- Java Code can be compiled anywhere
- Bytecode can be executed anywhere



*Can you make any guesses which statement above is for portable and which one for platform independence?*

# Portable

- When java code executes, its behavior is exactly same in any java-aware system.
- There are no platform-specific code in java programs that causes compilation problems in any other OS.
- This makes Java programs are portable.

# Platform Independent

- A Java program requires JVM (part of JRE) to execute Java code. When java application starts to executes, **Java Virtual Machine** also starts.
- Bytecode has instructions that **Java Virtual Machine** can understand and execute.
- JVM converts the Bytecode to machine specific code.
- Java Bytecode can be copied on to any machine that has JVM and executed. This is what makes Java **Platform Independent**.
- **“Write Once, Run any where”**

 *Is JVM platform independent?*

# JVM: perspectives

- JVM can be looked as
  - a runtime instance: JVM life cycle begins when applications starts to run ( that is when main method is called) and ends when the application ends.
  - the abstract specification: Specification that Java team at Sun (Oracle) provides which tells JVM makers how they must design JVM for their OS.
  - a concrete implementation: JVM that is built specifically targeted for an OS based on abstract specification .

# Interpreter vs JIT

- Java Bytecodes were originally designed to be interpreted by JVM meaning bytecode are translated to machine code without it being stored anywhere.
- Since **bytecode verifier** (which is part of JVM) performs runtime checks, line by line execution was important.
- Since speed became an issue, Just-in-Time Compilation (JIT) came into being. JIT converts chunks of code, stores it temporarily in memory and then executes the converted code.
- JIT compilers are typically bundled with or are a part of a virtual machine and do the conversion to native code at runtime, on demand.
- The compiler also does automatic register allocation and some optimization when it produces the bytecodes.
- Therefore, JIT is hybrid compiler.

# Error handling in java

- **Compilation error** : generated by the compiler. Examples of situation when it arises:
  - incorrect syntax, bracket mismatch, if keywords are used as variables
  - Using uninitialized variables
  - Unreachable code: **while (false) { ... }**
  - Strong type checking
- **Run-time error** : generated by the JVM.
  - Examples of situation when it arises:
    - Attempt to divide an integral value by 0 or access array index beyond the defined range.
    - Trying to access a class that does not exist at runtime. (What happens if you delete Hello.class and then run Hello.
  - Java has a strong exception handling mechanism that allows programmers to handle such errors and come out of the situation gracefully.



## Automatic garbage collection

- The garbage collector is a tool that attempts to free unreferenced memory (memory occupied by objects that are no longer in use by the program) in programs.
- *Automatic garbage collection* is an integral part of Java and its run-time system.
- Java technology has no pointers. So there is no question of allocation and freeing by programmers.
- Does that mean Java does not support dynamic memory allocation?
- No. It means that Java takes care of memory and relieves the programmers from memory-related hassles.
- `java -verbose:gc` can be used to get more information about garbage collection process.

# Classpath

- Classpath is an environment variable that java compiler and JVM (system class loader) use to locate the classes in the file system.
- To set the classpath temporarily in command line

**SET CLASSPATH=directory;%CLASSPATH%; .**

- Command to set classpath while compiling

`javac -classpath dir1;dir2 Someclass.java`

- Example: `javac -classpath C:/MyJava`

- Command to specify un-compiled source file to be compiled and included in the classpath

• `javac -sourcepath dir1;dir2 Someclass`

- Example: `javac -sourcepath C:/MyJava`

- Providing classpath while executing

• `java -classpath directory1;directory2`

```
public class A{  
    public static void main(String[] args)  
    {  
        String curDir = System.getProperty("user.dir") ;  
        System.out.println(curDir) ;  
    }  
}
```

*This code is in C:\MyJava folder. If this code is executed from C:\MyJava, it prints the current directory.*

*What command will make sure that this code can be executed from any location?*  
*(10 mins)*

# Commenting source code

*Question: Why should you comment your code?  
How much should you comment?*

- 3 types of comment

- Single line comment : //

- ```
// this is a single line comment
```

- Multi-line comment: /\* \*/

- ```
/* this is multi
```

- ```
line comment */
```

- Documentation Comment (Doc comment): /\*\* \*/


- ```
/** This class is used to represent a stack.
```

- ```
 * @author Murali
```

- ```
 * @version 1.0, 08/16/2010
```

- ```
 */
```

# Some Guidelines for Doc comments

- Who owns and edits the Doc Comments: usually the programmer
- Doc comments for classes, methods, fields:
  - Brief description of what it does. In case longer description is required, link to an external document (word, pdf) can be included.
  - For methods, whether they are thread-safe and what runtime exception they throw must be specified.  *Revisit this after threads and exception*
- Proper indentation of documentation for better readability.
- Entities for the less-than (<) and greater-than (>) symbols should be written &lt; and &gt;. Likewise, the ampersand (&) should be written &amp;.



For More Information refer

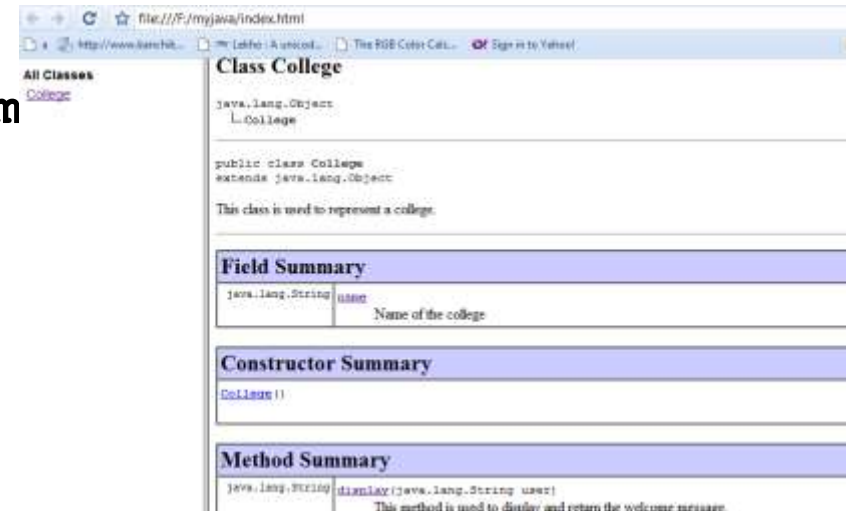
<http://download.oracle.com/javase/1.4.2/docs/tooldocs/windows/javadoc.html#documentationcomments>

 *Visit this after finishing Java*

# javadoc

- Tool that is used to produce HTML pages by parsing through the documentation comment in java source code.
- Produces documentation for public and **protected** classes/members.
- Works on entire **package** or a single source file
- Usage on source file
  - `javadoc sourcefilename`

`javadoc College.java`



# Nesting comments

- Valid Nesting

```
/*  // */          /** // */  
///* */          // /** */
```

- Invalid Nesting

/\* \*/ and /\*\* \*/ nested with itself and nested with each other gives error.

```
/* /* */ */  
/** /* */ */
```

# Annotations

- *Annotations* are extra text starting with @ symbol that are added in the Java program to provide information about a program.
- Annotations do not directly affect program semantics.
- They provide information about the program to tools and libraries, which can in turn affect the semantics of the running program.
- Annotations can be read from source files, class files, or reflectively at run time.



# Uses

- Annotations for the compiler
  - **Example:**
    - **@SuppressWarnings** : can be used by the compiler to detect errors or suppress warnings
- Annotations for the application servers and tools
  - Application server tools can process annotation information to generate code etc for services it provides like security etc.
  - **javadoc** tool uses annotations like
    - **@author, @version, @param, @return**
- Runtime processing — Some annotations are available to be examined at runtime.

# Exercise

- *Provide java doc comments for the class created in the previous exercise and generate HTML document files. Use documentation annotation.*

*(10 mins)*

# Summary

- Java was created by a team of members called "Green" led by James Arthur Gosling in 1991 and it was originally called as Oak.
- IDE is where the Java programs are written, compiled and executed.
- Simple, Object oriented language, Portable and platform independent, Robust, Multithreaded, Dynamic Linking, Secure and Performance are some of the features of Java Language
- Automatic garbage collection is an integral part of Java.
- Classpath environment variable is used to locate classes in the file system.
- Javadoc is a tool that is used to produce HTML pages by parsing through the documentation comment in java source code.
- Annotations are extra text starting with @ symbol that is added in the Java program to provide information about a program.