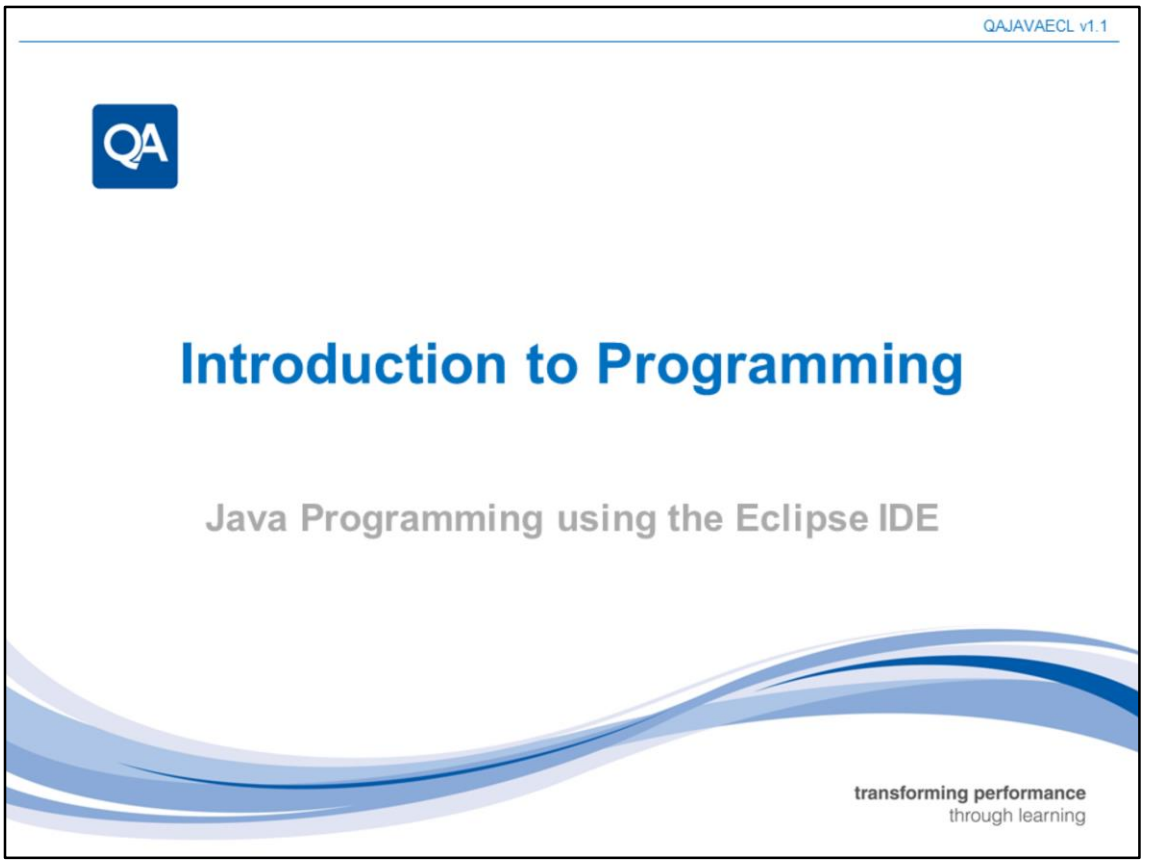

QAJAVAECL v1.1
Introduction to Programming
Java Programming using the Eclipse IDE
transforming performance
through learning

Outline

- **Introduction to programming and Java**
- **Installing Java and Eclipse**
- **Hello World!**
  - Input/Output
  - Packages
- **Running our program**

Objectives

- **By the end of this session we should be able to**
  - Describe what a program is
  - Describe what Java is and understand the basics of how Java code gets run
  - Be able to install Java and Eclipse
  - Write our first “Hello World” program

Outline

- **Introduction to programming and Java**
- **Installing Java and Eclipse**
- **Hello World!**
  - Input/Output
  - Packages
- **Running our program**

QAJAVAECL v1.1

What is a program?

- **A program is a sequence of instructions for something to carry out**
  - Get a mug
  - Find the coffee
  - Open the coffee
  - Put a teaspoon of coffee in the mug
  - Add water
  - Add milk (and sugar)
- **Programs describe the process of doing something**
  - Doesn't need to be for a computer!
  - Use a language both parties understand
  - Contain and manipulate data
- **These days the instructions for a computer are more often written in a high level language**

What is programming?

- **Programming is the act of telling a computer what to do**
  - We need to be able to describe what we want in a series of steps
  - We need to write these steps using a programming language
  - This is then translated (compiled) from high level code to a low level format the computer is able to understand
  - Eventually this is a series of instructions in binary which is executed by the computer

What is Java?

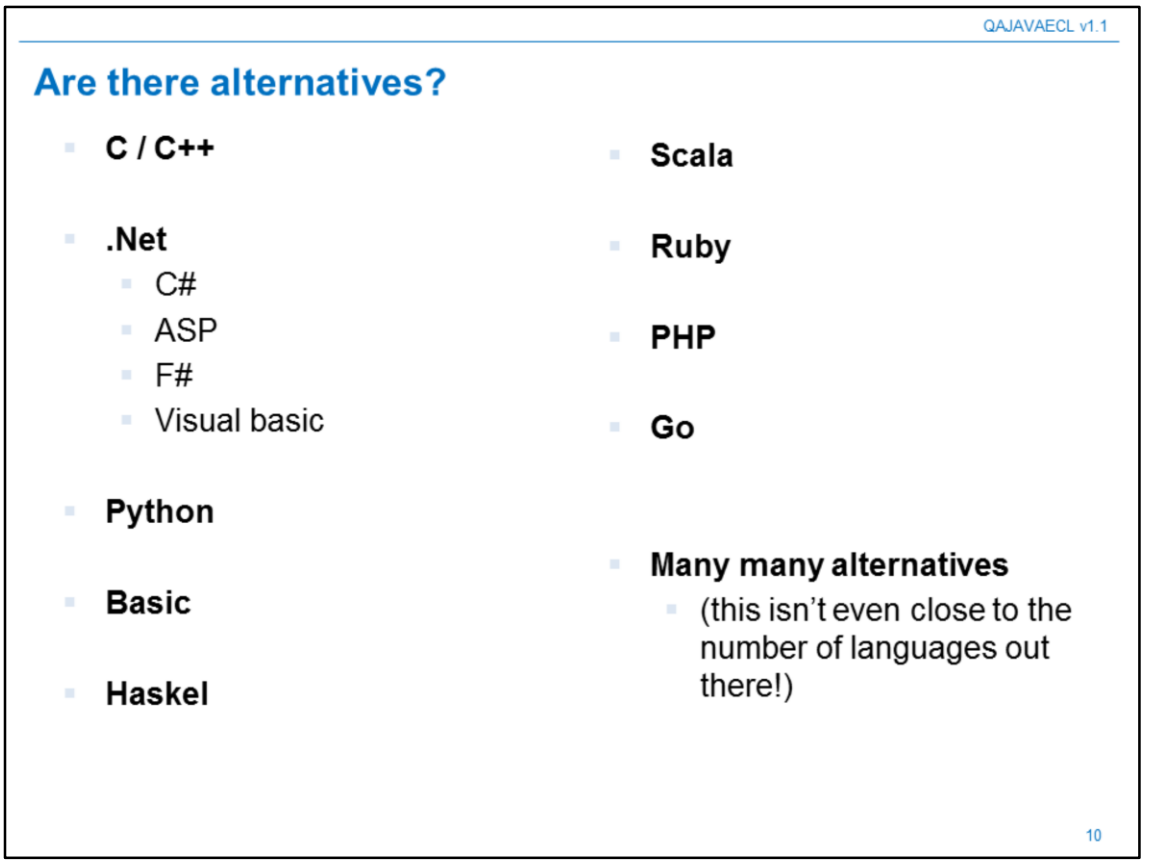
- **Object Oriented Programming Language**
- **First released in 1995**
  - Used to be owned by Sun Microsystems
  - Now owned by Oracle
  - Now up to version 8!
- **Popular due to the "write once run anywhere" philosophy**
  - All code compiles down to bytecode which is run on the Java Virtual Machine (JVM)
  - Operating system independent
  - There are JVMs for just about any type of hardware
- **Used as a teaching language in many universities**

Why is Java popular?

- **HUGE number of libraries available**
  - Can also connect with other languages easily (Scala, Clojure, Groovy)
- **Well-supported**
  - A lot of people and guides available to help
  - (Everyone seems to have done a bit of it)
- **Portability**
  - Take the exact same code and run on different servers
- **Java frameworks**

Java Design Criteria

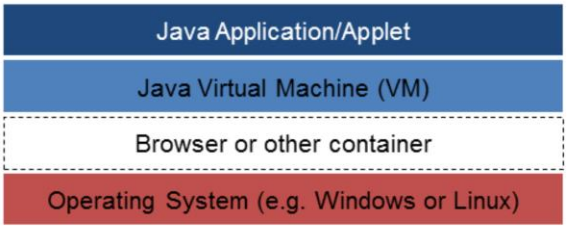
- **Platform-independence**
  - Java source code (xyz.java) compiles to processor-independent, platform agnostic 'bytecodes' (xyz.class)
  - Bytecodes then compiled at runtime using platform specific JIT compiler, so same program runs on any platform supporting Java
- **Robust**
  - Fully object oriented, every line of code belongs to a class
  - Strict type checking by compiler
  - Built-in and enforced exception handling
  - No pointers
- **Small and fast**
  - Each class is only loaded if needed
  - Built-in multi-threading
- **Secure**
  - Runtime environment can restrict what a Java class can do



http://en.wikipedia.org/wiki/List_of_programming_languages has a more comprehensive list, but even that won't be complete!

How does Java work?

- **Java code is compiled into Java byte code**
  - This is the .class file
- **This is then run on the Java virtual machine (JVM)**
- **The JVM translates between the byte code and the underlying operating system**
- **This makes java code portable**
  - The JVM implementation needs to be specific to the processor and operating system, but a compiled java class does not.



11

What is the JVM?

- **The Java Virtual Machine loads the java classes that are present on the class path**
  - This can be the .class files you've written
  - .jar files you've included
  - On the command line we include the parameter –cp
  - In an IDE we add the jar files to the build path
- **The JVM verifies all the classes it loads**
  - Checks for any errors
- **It also holds the security manager**
  - This restricts access to the host machine or other files dependant on properties

JRE vs JDK

- **JRE: Java Runtime Environment**
  - What we need to run a Java program
  - Installed on most machines
- **JDK: Java Development Kit**
  - The Java compiler and other tools we need to be able to create Java programs
  - We need a JDK on any development system!

Java Versions

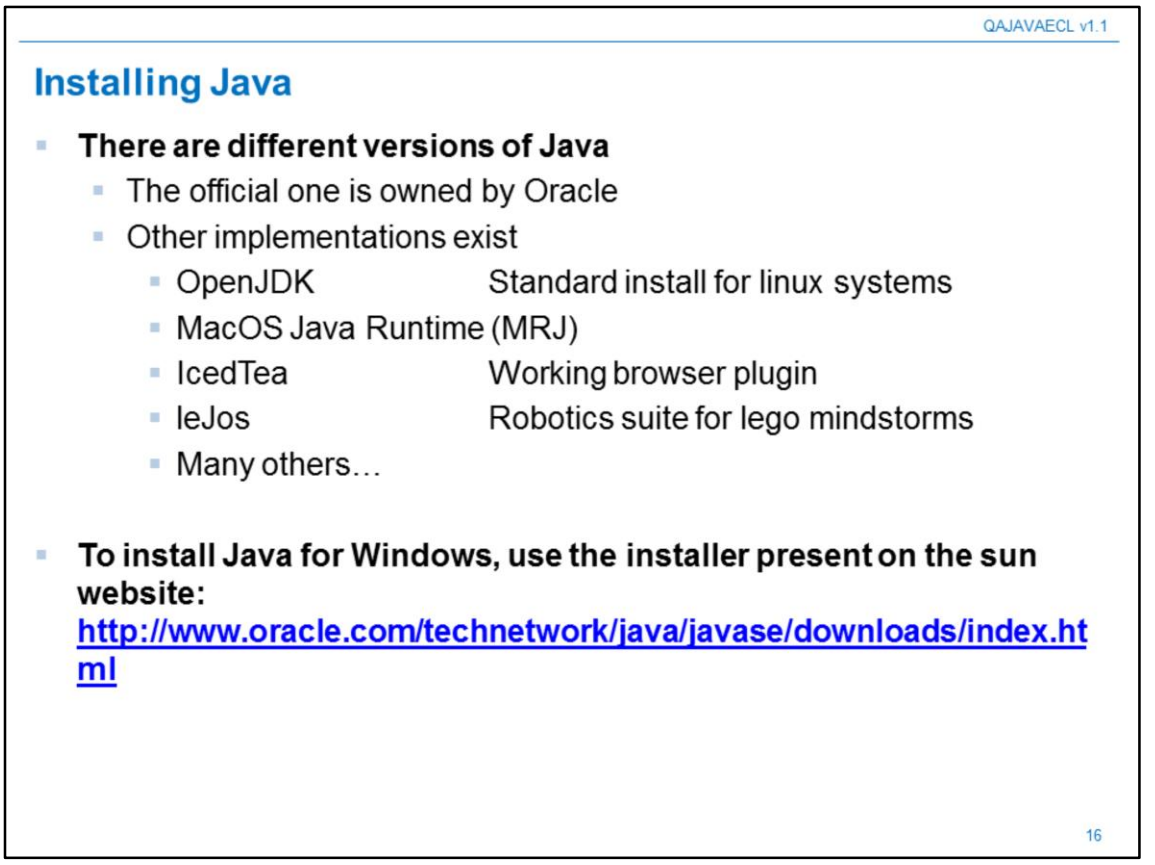
- **There are many different versions, every one has included some new functionality**

Version Number	
1	The original version!
2	
3	
4	Assertions
5	Enums, Generics
6	@Override
7	Diamonds <>
8	Lambdas

- **The 1.8 JDK will also allows you to compile code using version 1.3 onwards**

Outline

- Introduction to programming and Java
- **Installing Java and Eclipse**
- Hello World!
  - Input/Output
  - Packages
- Running our program



http://en.wikipedia.org/wiki/List_of_Java_virtual_machines has a long list of JVM implementations

What is an IDE?

- **Integrated Development Environment**
- **Provide tools to make writing code easier**
  - Auto-completion
  - Error checking
  - Debugging
  - Code inspection
  - Refactoring
  - Auto generation of source code
- **Don't need to use one**
  - Can just use a text editor and the command line
  - Doing so is likely to be less productive

IDE Options

- **Netbeans**
  - Owned by Oracle
  - Support for many languages
- **Eclipse**
  - Open source project
  - Support for many languages
  - Different tools available
- **IntelliJ**
  - Becoming more popular
  - Main IDE for Android Studio
- **Many others**

Installing Eclipse

- Eclipse doesn't require 'installing'
- Download a zip file from the Eclipse website:
  - https://eclipse.org
- Unzip it to the location you want to store the program
- It doesn't need to add anything to the system registry, so it can be run from a USB stick or external drive without any problems

What is Eclipse?

- **Eclipse is an open source IDE**
- **Managed and distributed by the Eclipse Organisation**
- **Based on projects**

- **More than just an IDE for Java**

Eclipse speak for itself (www.eclipse.org).

…

Eclipse is a community for individuals and organizations who wish to collaborate on commercially-friendly open source software.

…

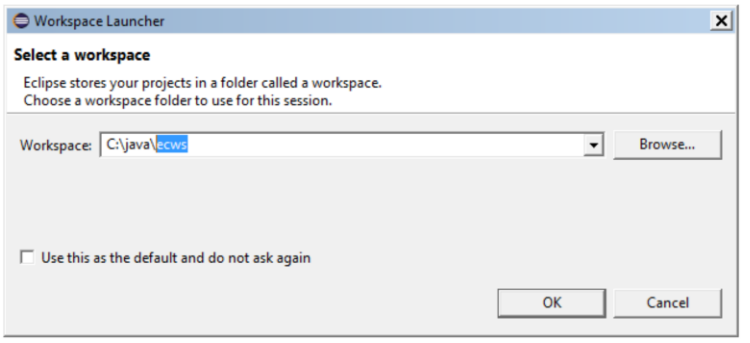
The Eclipse Project was originally created by IBM in November 2001 and supported by a consortium of software vendors. The Eclipse Foundation was created in January 2004 as an independent not-for-profit corporation to act as the steward of the Eclipse community…

Java developers will be working at project level, typically using the package explorer to navigate through the code.

The Eclipse Foundation and hence Eclipse itself supports other languages. Currently, they are providing environments for C/C++ and PHP.

Workspaces

- **Developer's work organised in Workspaces**
  - Name and Location chosen on entry
  - Defaults to this workspace



- **All work saved under this folder**
  - Organised in projects

Eclipse is based around workspaces and projects.

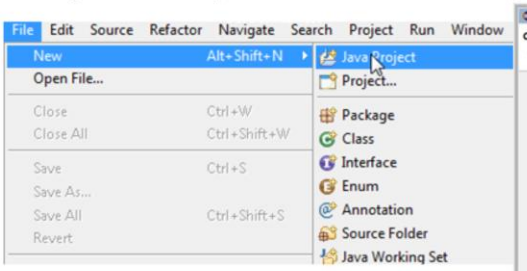
The workspace is just that – an area, i.e. folder in the file system, dedicated to the individual developer for his/her applications. Eclipse uses the project to hold each application. This depends on the user setting up a workspace on entry into Eclipse. Luckily, the user/developer normally has to do this once on first entry into the product – the workspace location is then chosen by default on all subsequent entries.

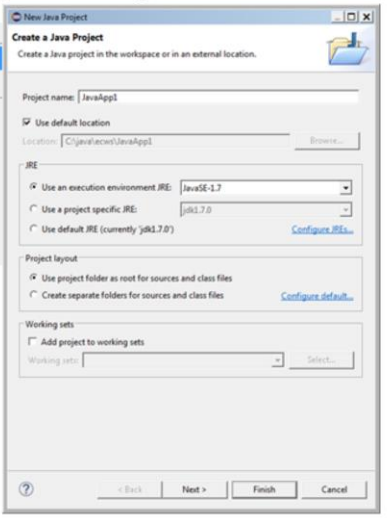
Be aware that the choice of location is very important. All project code, created and imported, will reside in the workspace folder.

Whenever the developer chooses to create a project it is, by default, contained in the workspace's folder as an aptly-named sub-folder.

Projects

- **Each application is organised within a project**
  - Category chosen via dialog(s) using File | New
  - Choosing Java Project leads to New Project Dialog





- **Choose Name (Mandatory!)**
  - JSE
  - Layout
  - Etc …
- **Named sub-folder created in workspace**

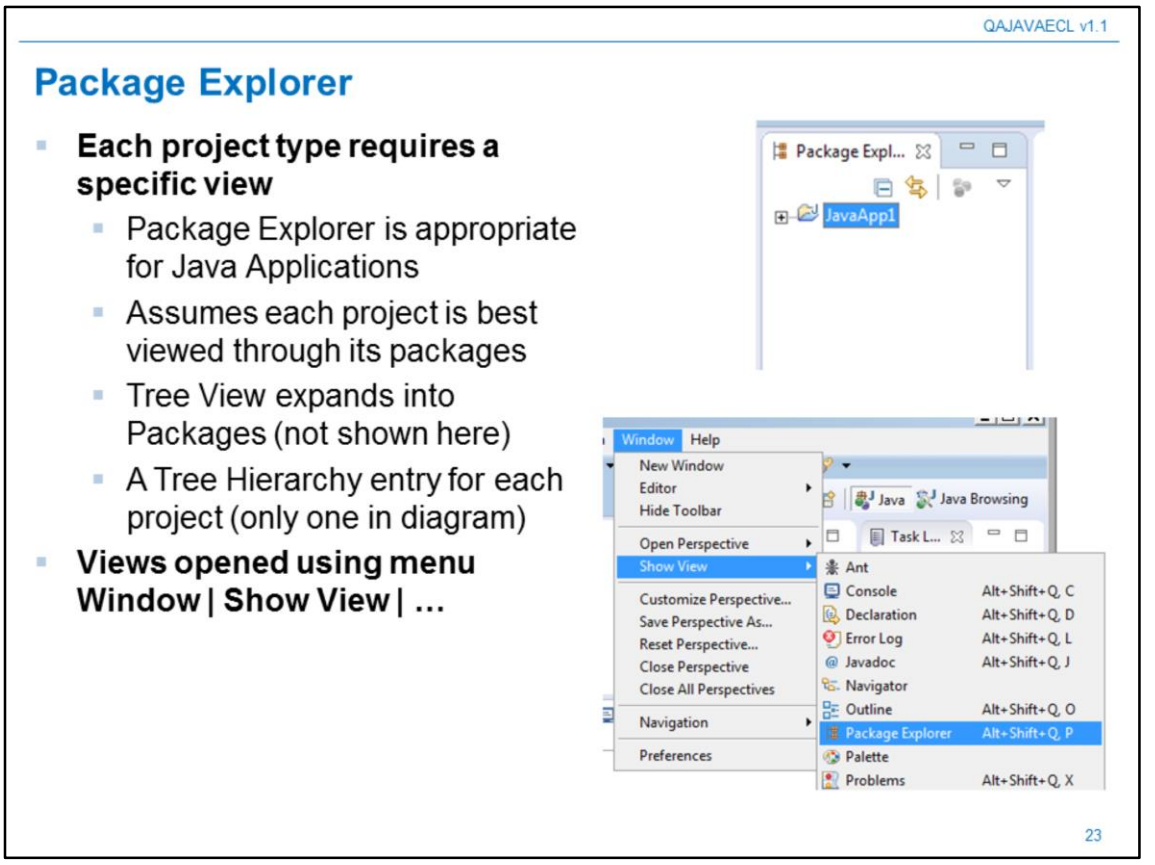
22

The Eclipse project is the main focus point for the developer user.

Choosing the category of project is the first task on selecting File | New. This course will only involve the Java Project type, but other types like web-based projects etc. are available dependent on which Eclipse IDE has been installed.

After choosing a name, which will be the identity of the sub-folder created in the workspace, other options are available which will configure the project as desired. Developer can chose a JSE, typically between Eclipse's own internal one and a pre-installed JDK located in the developer's file-system. The other main choice is how to arrange the file hierarchy for the source and class byte files. The location of these are heavily dependent on the packages within the application. The topic of packages will be covered at a later stage in the course.

Whenever the developer chooses to create a project it is, by default, contained in the workspace's folder as a aptly-named sub-folder. All files containing source code and utilities are saved within the folder as appropriate.



The project will be viewed though an appropriate view window. Developers are encouraged to use the Package Explorer for all Java based projects. On opening Eclipse for the first time, it happens to be already open and should be left open for most development tasks.

It can easily be re-opened using the Window | Show View menu option.

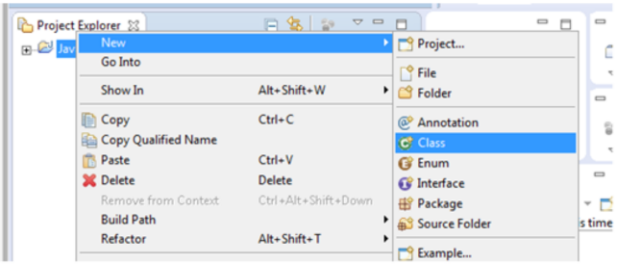
Eclipse also uses Perspectives. These are a category of view options, again, most appropriate for certain styles of application. Java Application projects are best suited to the Java aspect! This, again, is the open used when one first enters Eclipse.

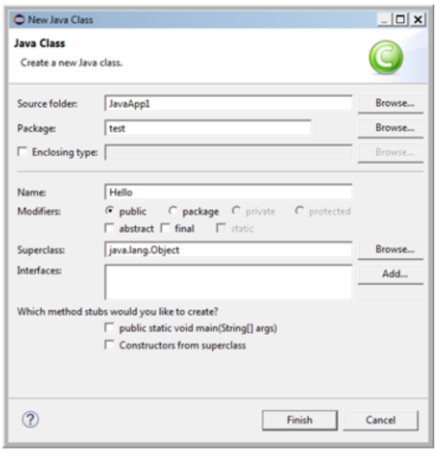
It is unlikely that you should need to change either the perspective or another view window throughout this course.

QAJAVAECL v1.1

Adding Items to Projects

- **Projects start 'empty**
  - Tree contains JSE only
- **For new items, highlight project and open context menu New | Class**
- **Opens up the New Java Class dialog**
  - Enter package and Class names (always)
  - Change modifiers and/or Superclass (occasionally)
- **For existing items we use Import (later)**





24

It is not surprising that all newly created projects will effectively be empty of functionality. Its entry in the package explorer can be expanded, but it will only show the JSE information. This will depend on the choice made during the project creation.

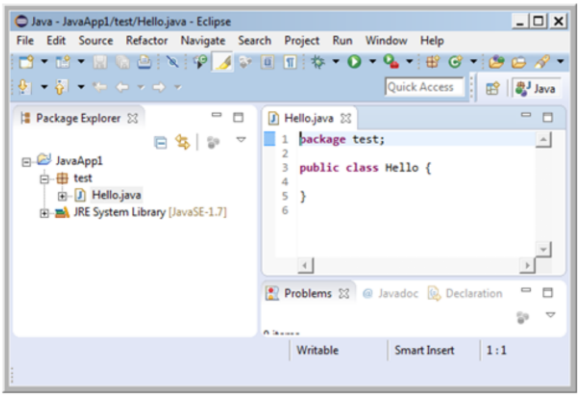
Adding items is an easy task. Having highlighted the project in the explorer, open the context menu (usually a Right-Mouse down click) and select the New option. This will open up a sub-menu with the dozen or so possible items that are appropriate for the project category. The most common one would be the class. Other items could be text/xml files (File), folders (Folder) to store such files or other data files, an empty package (Package) in anticipation of adding further classes later, etc …

The selection of class will display the New Class Dialogue, where the developer can name the class, including its package. There are other options, but these will be left until later in the course (or later in your Java career!)

The above procedure is required are adding new items to the project. If the developer For existing items we need to use the Import facility. This is covered later in this chapter.

Class File

- **Class file is added to the project's tree structure**
  - Contained in the chosen-named package
  - Open in the main pane (by default)
  - Eclipse supplies an empty class for you



- **Hello.java file now available for editing**

25

Once the class has been added to the project, the source file containing the class appears within the project's tree in the package explorer. Initially the file is open in the main pane of the IDE, but this may be closed (and opened again).

Eclipse puts the mandatory code in the file for you:

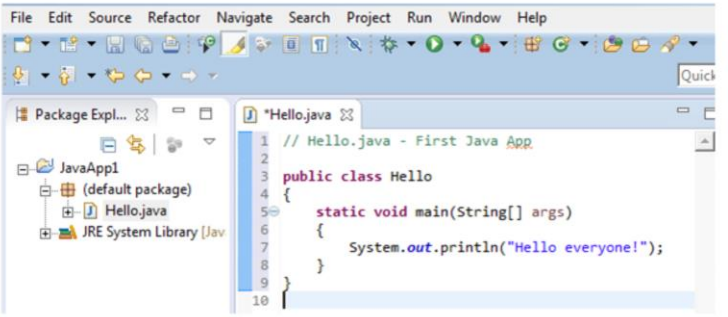
```
package test;


public class Hello {


}
```

The package statement has to be the first non-comment line of source in the file. The empty class is simply the obvious starter code for the class you have chosen.

The code generated will depend on the various options chosen when creating the class. Some of these options will be discussed during the exercise and ensuing exercises.

Getting Started

- **All code is added via the main pane**
  - Simply type java code
  - Colour and Font 'coded'



- **The application is ready to run**

The developer can now edit the file. The example illustrates the minimum required code to have a runnable application.

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

Apart from the individual colour and font choice, the colour and font coding strategy is pretty much standard across development IDEs. The settings can be changed as required, using menu Windows | Preferences | General | Colors and Fonts We will keep the default – as shown here.

Here are the most common ones …

The font is Consolas 10.

This colour is used for Java keyword.

This colour is used for constants and literals.

Italic is used for statics.

The terminology used here will covered in the next few chapters.

Exercise Time

- **Download and install Java and Eclipse**
  - Check that they are both working and create a new project