



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**

Anatomy of a Java Program – Writing our first class

- **The JVM looks for a main method**
 - This is a specific method that sets up and runs the entire program
 - Must be public: Otherwise it cannot be seen from outside the class
 - Must be static: Otherwise we need to create an object
 - (Don't worry about this too much yet)
 - Must have the parameters: String[] args: These are the arguments passed to the program

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

3

This program will print out to the screen “Hello World” when it is run. We'll break down the specific parts over the next few slides, so don't worry if it doesn't make sense straight away.

The important thing to remember is that this pattern is fixed. A main method will always have the same method signature as we see here. It will always be of the form:

```
public static void main(String[] args) {  
    ...  
}
```

Anatomy of a Java Program – Classes?

- **The first line of our program is the class identifier**
 - Public means that other parts of a java program can see this class
 - Classes are normally declared public unless there is a very specific reason not to
 - Class is a keyword to tell java that this is a class description
 - Main is the name of this class
 - It did not need to be called main, it can be anything you like

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

Anatomy of a Java Program – The method call

- **Public** – means that this method can be seen outside the class
- **Static** – means that we don't need to create an instance of this class before calling the method
- **Void** – the return type – we're not returning anything so we use the keyword void
- **Main** – the method name. For a main method the name of this **is** important

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

Input - Method Parameters

- The method parameters are what we are passing to the method for it to use
- In the main method, this comes from either the command line when we run the program or in an IDE it comes from the run settings
- **String[] args** – there is an array of String objects passed in, with the name 'args'

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

Anatomy of a Java Program - Comments

- **Comments are ignored by the java compiler**
- **They are there to let people reading the code know what is going on**
- **Java has three types of comment**
 - `//` - single line comment
 - `/* ... */` - Multiline comments
 - `/** ... */` - Javadoc comments. Read by the Javadoc tool to create documentation

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

Anatomy of a Java Program - Output

- **The final line prints everything out to the command line**
 - `System` – Refers to a java class called `System`
 - `out` – Call a method in the `System` class. It is static as we're not creating a system object ahead of time
 - `println` – Call the `println` method which prints things to the screen
 - `"Hello World"` – The input parameters for the `println` method, this is the string we are printing out
 - `;` - Very important! Every line of Java code ends with a semi colon

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


Packages

- **Hierarchical grouping structure for sorting Java files**
 - Each package contains a collection of classes
 - Avoid name clashes
- **It is not best practice to create a class outside a package**
- **Lots of different built in packages**

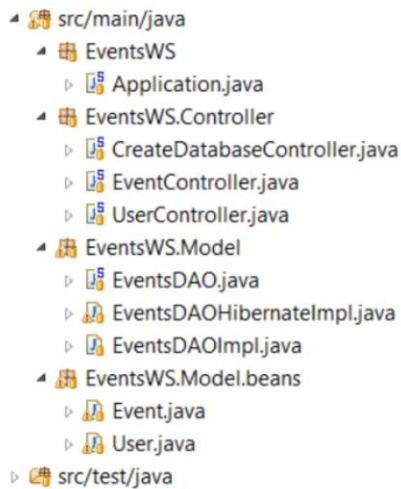
Function	Name	Example Classes
Language	java.lang	Object, String, Thread, Runtime, System
Swing	javax.swing	Jframe, more up to date UI components
I/O	java.io	InputStream, OutputStream, File
Utilities	java.util	Date, Stack, Vector, ArrayList

9

This list isn't exhaustive, there are loads of different packages

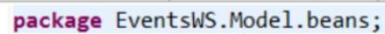
Putting classes in packages

- **Create a package**
 - This is created as a set of folders on your system
- **Add the package declaration to a class in that folder**



The screenshot shows the Eclipse IDE's project explorer on the left. The project structure is as follows:

- src/main/java
 - EventsWS
 - Application.java
 - EventsWS.Controller
 - CreateDatabaseController.java
 - EventController.java
 - UserController.java
 - EventsWS.Model
 - EventsDAO.java
 - EventsDAOHibernateImpl.java
 - EventsDAOImpl.java
 - EventsWS.Model.beans
 - Event.java
 - User.java
- src/test/java



The screenshot shows a Java code editor with the following package declaration highlighted in blue:

```
package EventsWS.Model.beans;
```

Outline

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

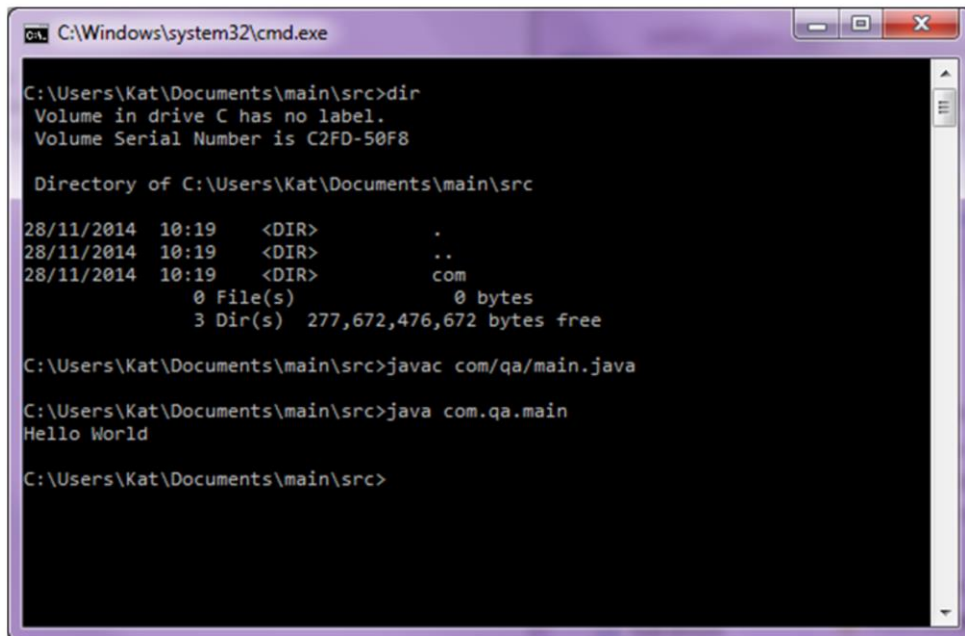
Running a program from the command line

- **Running a java program has two steps**
 - Compile using javac
 - Run using java
- **When using javac we want to go to the base directory for our program**
 - This is the src directory that eclipse created for us
 - We need to give the file the full path for javac to find it

```
javac com/qa/Main.java
```

 - This will create a .class file in the same directory.
- **To run it we need to use the full package of the file (but not the extension)**

```
java com.qa.Main
```



A screenshot of a Windows command prompt window titled "C:\Windows\system32\cmd.exe". The window shows the following commands and output:

```
C:\Users\Kat\Documents\main\src>dir
Volume in drive C has no label.
Volume Serial Number is C2FD-50F8

Directory of C:\Users\Kat\Documents\main\src

28/11/2014  10:19    <DIR>          .
28/11/2014  10:19    <DIR>          ..
28/11/2014  10:19    <DIR>          com
               0 File(s)                0 bytes
               3 Dir(s)  277,672,476,672 bytes free

C:\Users\Kat\Documents\main\src>javac com/qa/main.java

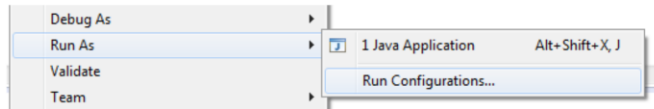
C:\Users\Kat\Documents\main\src>java com.qa.main
Hello World

C:\Users\Kat\Documents\main\src>
```

Running a program from eclipse

- **Eclipse compiles the program for us continually**
 - This is how the syntax highlighting works
- **We have three options for running the program**
 - Right click on the project or file with the main class in and select 'Run As' and then 'Java Application' from the context menu

- Press Ctrl + F11



- Go to the menu bar at the top, select the run menu
 - If you have run the program before, then we can use "Run Last Launched"
 - If you haven't then there is a similar "Run As" menu option

Programming time!

- **Write our first Hello World program**
 - Try compiling and running it both from the command line and the IDE!

Summary

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