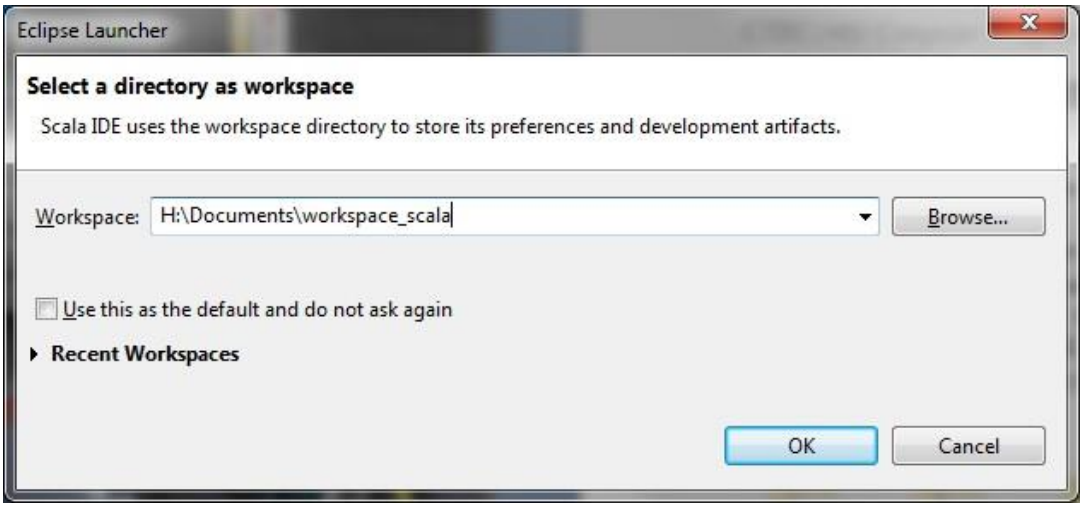
Getting started with ScalaIDE for Eclipse

ScalaIDE for Eclipse is the development environment through which you will be developing, compiling and running your Scala programs. To get started open up ScalaIDE from the start menu, you should be prompted with the Eclipse Launcher.



This launcher is asking you where you would like to store the Scala programs that you will develop within Eclipse – this is called a workspace, which simply represents a chosen folder on the underlying file system. For now you may use the default location on your `H:` drive. **Note**: This exercise assumes you are working on University machines – if you are working on your own laptop, please use an equivalent folder on e.g. your local `C:` drive, where applicable.

Once loaded, the welcome screen has various options. Click on the workbench icon, which should be displayed towards the top right-hand corner of the screen. At this point, you will see the workbench, which may at first look a little overwhelming. An Integrated Development Environment (IDE) is a sophisticated application that aids developer productivity, however, we only need to focus on the basic components for now.

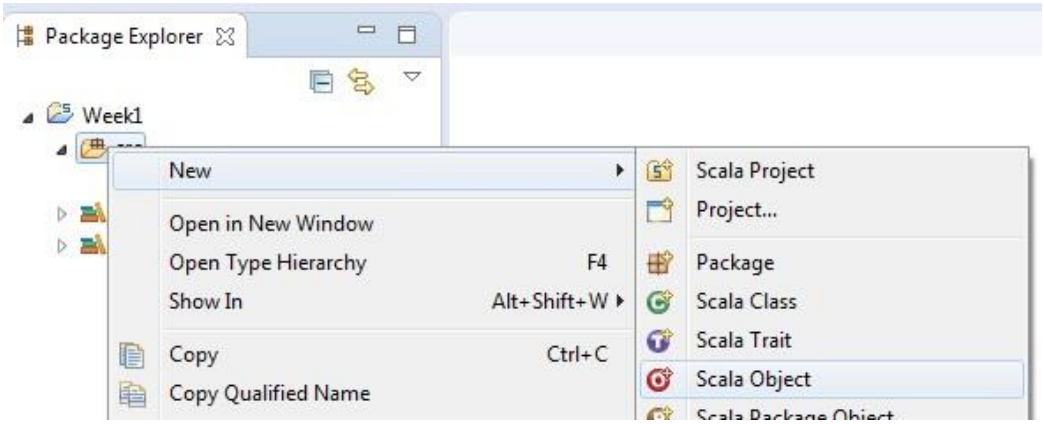
Creating your first Scala program

Before you can create your first Scala program you will need to create a new Scala Project. An Eclipse workspace may contain a number of projects, which may themselves contain a number of computer programs. On the menu go to **File > New > Scala Project** and give the project a name; let's simply call it **Week1**. You may use the default settings and click finish.

The project will appear in the Package Explorer on the left-hand side. You can think of this as the file manager within Eclipse where your projects and Scala programs will be listed. Click the arrow to the left of your newly created project to view its contents. You will notice that Scala libraries are included in your project as you will be developing Scala programs. Additionally, you will see the JRE (Java Runtime Environment) libraries as Scala programs run on the Java Virtual Machine (JVM).

Now right-click on the "src" (shorthand for source code) folder and select: **New > Scala Object**. You will be prompted with a dialogue asking you to provide a name for this new file. For now

you may simply think of a Scala Object as being a Scala program and therefore the name you provide will be the name of your program. Name the program **HelloWorld** and click finish.



You will see your newly created program (HelloWorld.scala) appear within your project underneath something called a default package. When working on larger applications a project may contain numerous programs and these can be logically separated into different packages. For now the default package is suffice, meaning each of your programs will simply sit directly within the source folder.

In the central area of the workbench you will now see a template Scala program that has opened up in the editor. A source code editor is a text editor specifically designed for writing computer programs. At the moment, the program does nothing. Update it so that it contains the following:

```
object HelloWorld {
  def main(args: Array[String]): Unit = {
    println("Hello world!")
  }
}
```

You will notice the syntax highlighting that the editor provides making it easier to identify different parts of the program. When ran this program will output a simple message saying "Hello world!". For now go to the menu **File > Save** (or press `Ctrl+S`) to save it.

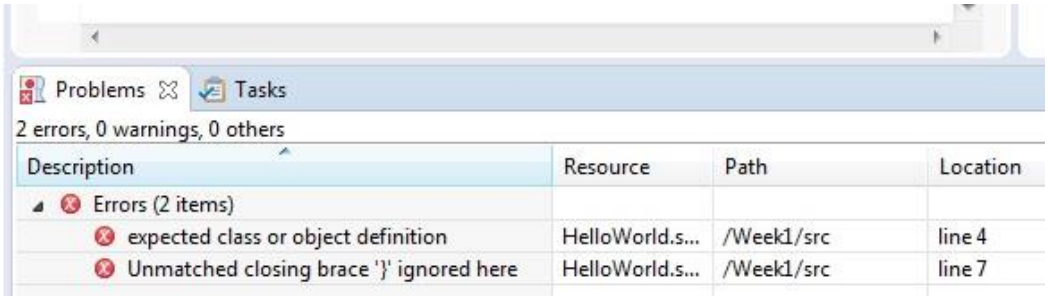
Compiling and running a program

You can now compile and run the program. Compiling a program converts the source code you have written into a byte code executable that can be ran on the JVM. It so happens that your programs are compiled automatically as the Project > Build Automatically menu option is selected by default. If this option was unchecked you could explicitly compile your program using the Build Project option from the same menu. Nothing obvious would happen, although this will have compiled your program behind the scenes

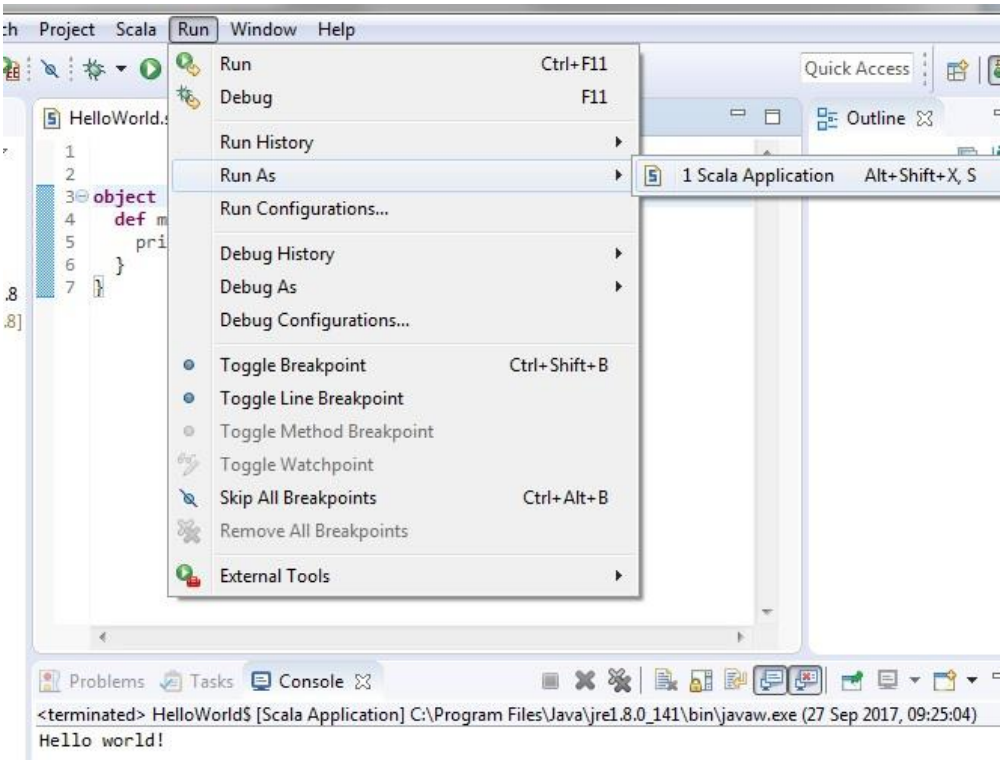
These points are reinforced if you purposely introduce a syntax error into your program. Remove one of the curly braces and save your program – you will see some red crosses have appeared. This is because your program has been compiled automatically and the compiler is notifying you that your code does not conform to the syntax of the Scala language.

If you hover over or click on the errors you will see the specific issue. You can also view current

compilation errors within the Problems window if you click on the Problems tab towards the bottom of the workbench.



Remove any compilation errors by returning the program back to its previous state. You can now run your program by going to the menu **Run > Run As > Scala Application**.



You will now see a Console window appear towards the bottom of the workbench. This is where all standard output from your programs will be written to, and also provides a basis for standard input from the keyboard. Your first Scala program has simply output the message "Hello World".

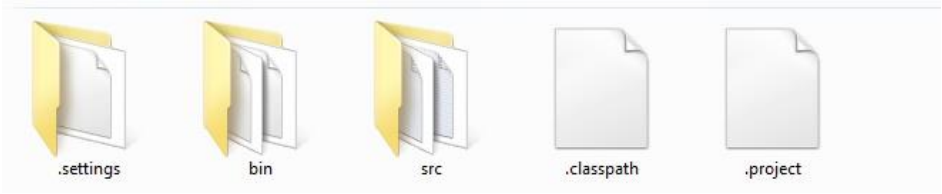
Notes - Basic concepts:

- `Object` – A typical Scala program is an object and has a name which should be capitalised.
- `main(...)` – A Scala program is executable if it has a main method, all executable code should be placed within the curly braces { } forming this code block.
- `println(...)` – outputs the content of the String parameter in double quotes to the standard output stream that appears inside the console window within Eclipse.

Understanding the structure of a Scala project

When you create Scala projects and programs within Eclipse it is important to recognise where you are ultimately storing your work.

On the file system, go to the folder that you created for your current Eclipse workspace, e.g. `H:\Documents\workspace_scala`. Within this, you should see a folder called **Week1**, which represents the project you created. Opening up this folder will allow you to see all the related files, notably you will see a **src** and **bin** folder. The files and folder whose name starts with a `.` (e.g. `.settings`) are generally used for storing configuration details for a given project.



The src folder (shorthand for source) contains your HelloWorld.scala source code file, whereas the bin folder (shorthand for binary) contains a HelloWorld.class and HelloWorld$.class file, which were produced when the source file was compiled. These files contain byte code, which can be executed and understood by the JVM.

Summary - Eclipse project structure:

- Creating a project in Eclipse causes a new folder to be created within the current workspace location on the file system.
- Creating a Scala program object file within Eclipse causes a corresponding .scala source code file to be placed within the src folder of the current project.
- Running a Scala program causes the source file to be saved and compiled, which in turn generates or updates one or more corresponding .class byte code files, which are then executed on the JVM instance running on the machine.

You can now start to see some of the benefits of running programs within Eclipse - it does a lot of the donkey work for you.

Launching applications and using content assist

Let's now look at some other useful features within Eclipse. Right-click on the src folder and select **New > Scala Object** and name this file **DemoProgram**. You now have two Scala programs (i.e. source code files) within your project, which represent two independent executable Scala applications. You will note the interchangeable use of the terms application, program and source code file. For now each Scala application you create will only involve a single source code file, however later in the module you may see more complex applications that are made up of multiple source code files.

As previously stated, an executable Scala application requires a main method definition. Whilst

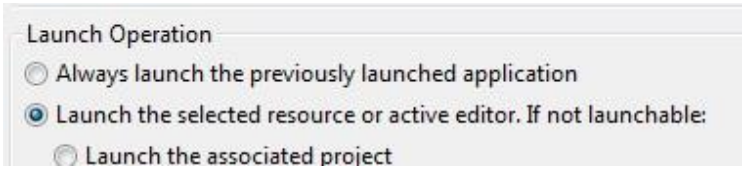
you can copy/paste this or type it manually there is a way to generate it within Eclipse by using something called content assist. In certain scenarios content assist is automatically enabled, however you can use it at any time by pressing **`Ctrl+Space`** whilst in the editor. If you do this and then start typing "main" you will see a dialog popup with suggestions that will eventually be refined to a single option matching the main method we require. You can preview the code it will generate and then use it by either double-clicking on it pressing enter.

Within this main method write the following statement:

```
println("A message to convert into uppercase")
```

We will modify this code shortly. For now let's run it to check your program works as you would expect. If you click on the run button (see below) or go to the Run menu and select Run Last Launched it will run your previous HelloWorld program.

To run this new program you need to select Run > Run As > Scala Application again. There is a simpler alternative that lets you run whichever program is currently in focus in the editor. To enable this go to the menu **Window > Preferences**, then dropdown **Run/Debug** and select **Launching**. Towards the bottom of this area change the selection to "Launch the selected resource or active editor" then click Apply and OK.

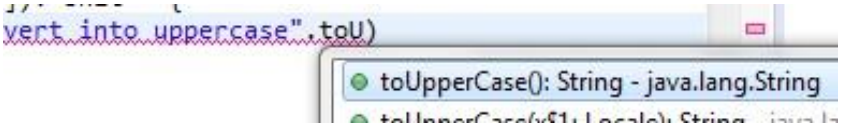


When you go back to the Run menu you can now simply select run, or alternatively click on the run button (or use the shortcut `Ctrl+F11`), to launch the current program you are working on.

Once you are happy you can run your program and view the output within the console window, revisit the source code and change the content of the `println` method to be:

```
"A message to convert into uppercase".toUpperCase()
```

When you type the `.` character following the string this will automatically cause a content assist dialog to appear and as you start to type it will filter and refine the available options. You only need to type "toU" to get the option you require. Run the program again to see the effect of this.



In summary, content assist cannot make decisions for you or help you in all of the code that you will write but it can assist you in certain circumstances and also comes with useful templates for commonly required definitions such as the main method.

Enabling line numbers, adjusting font size and code indentation

It's useful to be able to view line numbers within the source code editor; this allows specific lines of code to be identified and discussed between developers and can also help you to pinpoint the location of certain compilation errors. To enable them go to the Window > Preferences menu and then dropdown **General**, **Editors** and within the **Text Editors** area you will see a checkbox to "Show line numbers".

You will now see line numbers in the editor. Purposely introduce a syntax error by e.g. removing a curly brace in your program and then saving it. Now view the Problems window towards the bottom of the workbench (where the console is), you will now see the error location outlines a specific line number that you can correlate to the line number displayed in the editor. If you try and run a program that has compilation errors you will get a message stating that your project has build errors. If you continue it will either run the previous version that had successfully compiled or an alternative program you last ran.

Another useful customisation that you may wish to make is to adjust the font size within the editor. To do this go to Window > Preferences, then dropdown **General**, **Appearance** and select **Colors and Fonts**. In this area you need to dropdown **Basic** and scroll down to **Text Font** (which should be the last entry), selecting this and then clicking edit. You can now adjust the font size, e.g. to size 12, and apply the changes. You can of course choose a font size of your choice. You may also like to use Eclipse in full screen mode which you can do by going to **Window > Appearance > Toggle Full Screen** (or `Alt+F11`).

When writing computer programs there are many good practices or style guides that you should adhere to in order to make your code easier to read and understand. By using common styles different developers can more easily maintain each other's code. One of these areas is code indentation. The amount of whitespace is largely irrelevant to the compiler but can significantly aid readability.

If you look at your two existing Scala programs you will notice both have indentation applied as you copied some code into Eclipse that was already formatted in this way. Furthermore, as you type in the editor, and more specifically go onto the next line, the cursor should automatically be positioned with the correct level of indentation. As a general rule you should follow the "2-space convention" as noted in the official Scala style guide:

https://docs.scala-lang.org/style/indentation.html

For now you can consider that each code block (i.e. an area of code within curly braces `{ }`) should have its contents indented by two spaces. In your current programs you have the outermost code block defining the program itself and a main method code block required to run the program within this. Any code written inside the main method should itself be indented. When you start working with slightly larger and more complex programs throughout the module the benefits of this will become even more apparent as it will be much easier to identify logical pathways of execution throughout the program, which in itself can help with debugging errors.

Occasionally when you write code you will not initially indent it correctly. You may also copy/paste code that is not indented from another source. Let's demonstrate this now and then look at a fix. Copy the code below into your HelloWorld.scala program replacing the current content with:

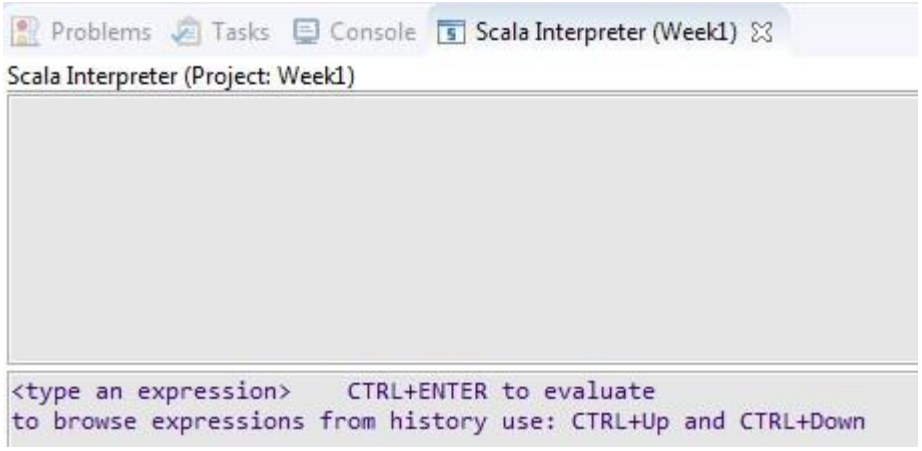
```
object HelloWorld {
def main(args: Array[String]): Unit = {
println("Hello world!")
}
}
```

Save and run the program, it should still work, but the code looks a little untidy! Fortunately instead of having the fix each individual line with the correct level of indentation yourself there is a shortcut to correct indentation within Eclipse. The simplest way to do this is to highlight all of your code within the editor (either by using your mouse or by pressing `Ctrl+A`) and then press **`Ctrl+Shift+F`** on your keyboard - the shortcut for formatting source code. You will now see the indentation has been corrected.

Opening the Scala interpreter within Eclipse

You have seen how you can write simple Scala programs within Eclipse. When you are learning a new programming language or want to experiment with some code without having to write a full-blown program you can alternatively use the Scala interpreter. This is an interactive environment that allows you to type single lines of code or expressions and then have them read and evaluated with the result being printed for you. This is why it is sometimes referred to as REPL (Read-Evaluate-Print Loop).

Whilst it is possible to run the interpreter from within a command prompt in Windows or a unix terminal in Mac/Linux, you can also conveniently run it inside of Eclipse too. To do so go to the menu **Window > Show View > Scala Interpreter**. A new window will appear towards the bottom of the workbench and you will need to select a project to associate the interpreter with. In this case select your Week1 project.



To make the interpreter window bigger you can either double-click on the Scala Interpreter tab so that it expands across the entire workbench (an action that can be reversed by double-clicking on the same tab again). Alternatively, if you still wish to see the code editor above, you can hover your mouse slightly above of the Scala Interpreter tab and then drag the window upwards.

Inside the bottom of the interpreter area you will see instructions on how to have an expression evaluated as well as ways of revisiting previous expressions you have typed. If you type the expression `1 + 2` and press `Ctrl+ENTER` you will see a result of:

```
res0: Int = 3
```

Whilst the answer of 3 will be no surprise, this result demonstrates that the number three is a whole number associated with the `Int` data type and has been placed in a temporary variable with the name `res0`. We can therefore reuse this result by referencing that variable, e.g. now try typing `res0 * 2`. Can you see what the `*` operator does?

We will revisit the interpreter periodically throughout the module and you can use it yourself to aid your understanding of the Scala langauage.

Importing projects into a workspace

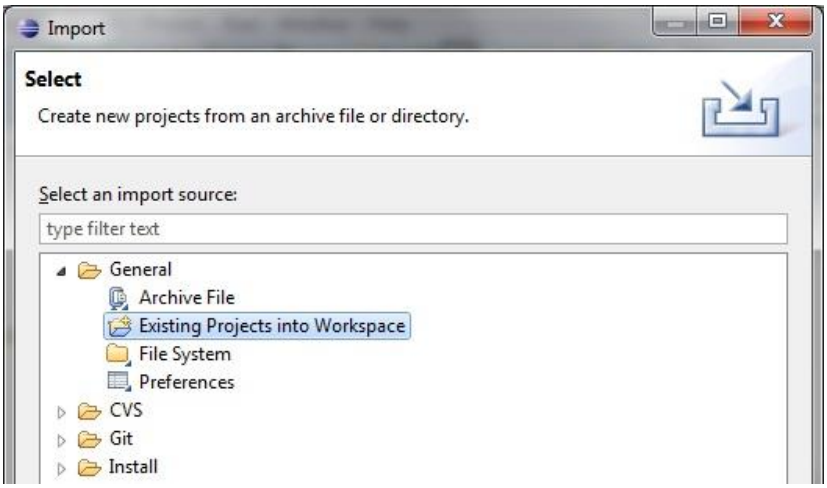
Later in the module you may be given an existing Scala project and be required to set it up within your workspace. Furthermore, you may wish to periodically transfer your work from the University machines to your own PC or vice versa. You can achieve these tasks by understanding how projects can be imported into your workspace.

Within Eclipse, go to the package explorer and right-click on the root folder of your project Week1, then select delete. You will get a prompt that asks you if you want to remove the project from the workspace. There is an additional option that you can select to remove the contents on disk - DO NOT select this. For the purpose of this example we want to see how a project can be removed from a workspace and then added back into it. So click "OK", but DO NOT select "Delete project contents on disk...".

You will now see that the project has been removed from within the package explorer, but if you look back at the file system (via the windows explorer), your project (and its contents) will still exist on your `H:` drive.

Now to retrieve the project back into Eclipse you will need to do an import. Go to the menu **File > Import... > General > Existing Projects into Workspace**.

You will then be faced with a dialogue asking you to locate the project that you wish to import. Select the **Browse** button and locate your project root folder, which will be called Week1. You will notice an option called “Copy projects into workspace” - you only need to select this if your project is located elsewhere on the file system and you wanted to physically copy it into the workspace folder instead of linking to it. In this case the project folder is already inside the workspace. After selecting Finish you should now see your project has been imported back into Eclipse and is visible in the package explorer window again.



Whilst we have ended up exactly where we started you have now seen how you could copy one of your project folders and import it into Eclipse on a different machine. If you simply want to do this for individual Scala programs you can drag and drop them into the src folder of your project in Eclipse, or alternatively save them in this folder on the file system and then right-click and refresh your project in the IDE.

Summary - Eclipse file management:

- Removing a project within Eclipse does not necessarily remove it on the file system.
- For a project to be transferred between workspaces or different computers, the project root folder should be copied into the destination workspace folder on the file system and then imported into Eclipse.
- Individual Scala source files can be copied into projects by placing them within the src folder of the project on the file system and then refreshing the project within Eclipse.