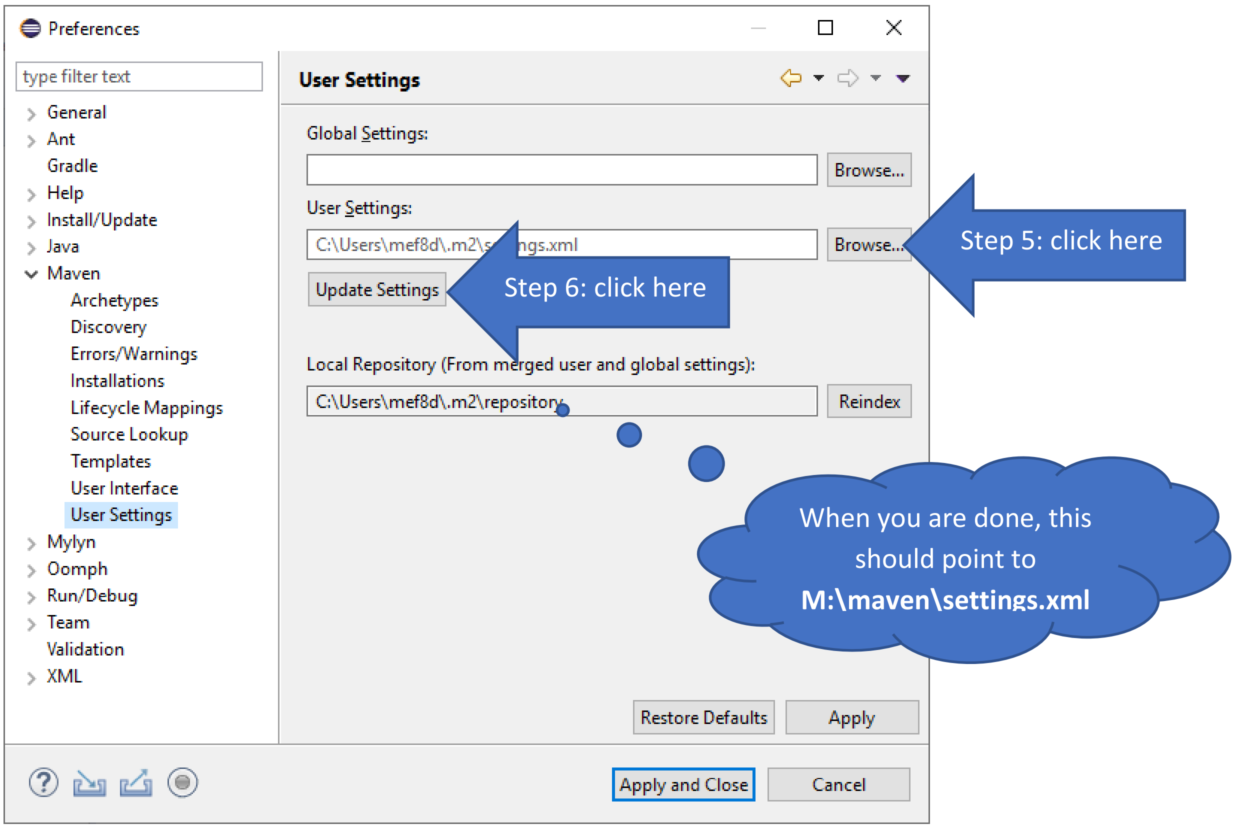
Maven exercises using Eclipse

VERY IMPORTANT: IF YOU ARE USING THE LAB MACHINES TO DO THESE TASKS, YOU MUST FOLLOW THESE INSTRUCTIONS FIRST TO MAKE SURE THAT MAVEN IS USING YOUR OWN FILE SPACE INSTEAD OF THE C:\ DRIVE ON YOUR COMPUTER!!! (These instructions are not necessary if you are using your own laptop).

1. Create a directory **M:\maven**
2. Create a file in that directory called **settings.xml**, cut and paste **this exact content** into the file, and then save it.

```
<settings xmlns="http://maven.apache.org/SETTINGS/1.0.0"

          xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

          xsi:schemaLocation="http://maven.apache.org/SETTINGS/1.0.0

                              https://maven.apache.org/xsd/settings-1.0.0.xsd">

  <localRepository>M:\maven\repository</localRepository>

</settings>
```

3. Now open Eclipse, open the **Window** menu, and go to **Preferences**
4. Expand **Maven** in the window, and click on **User Settings**. The window should look like this:



Global Settings:

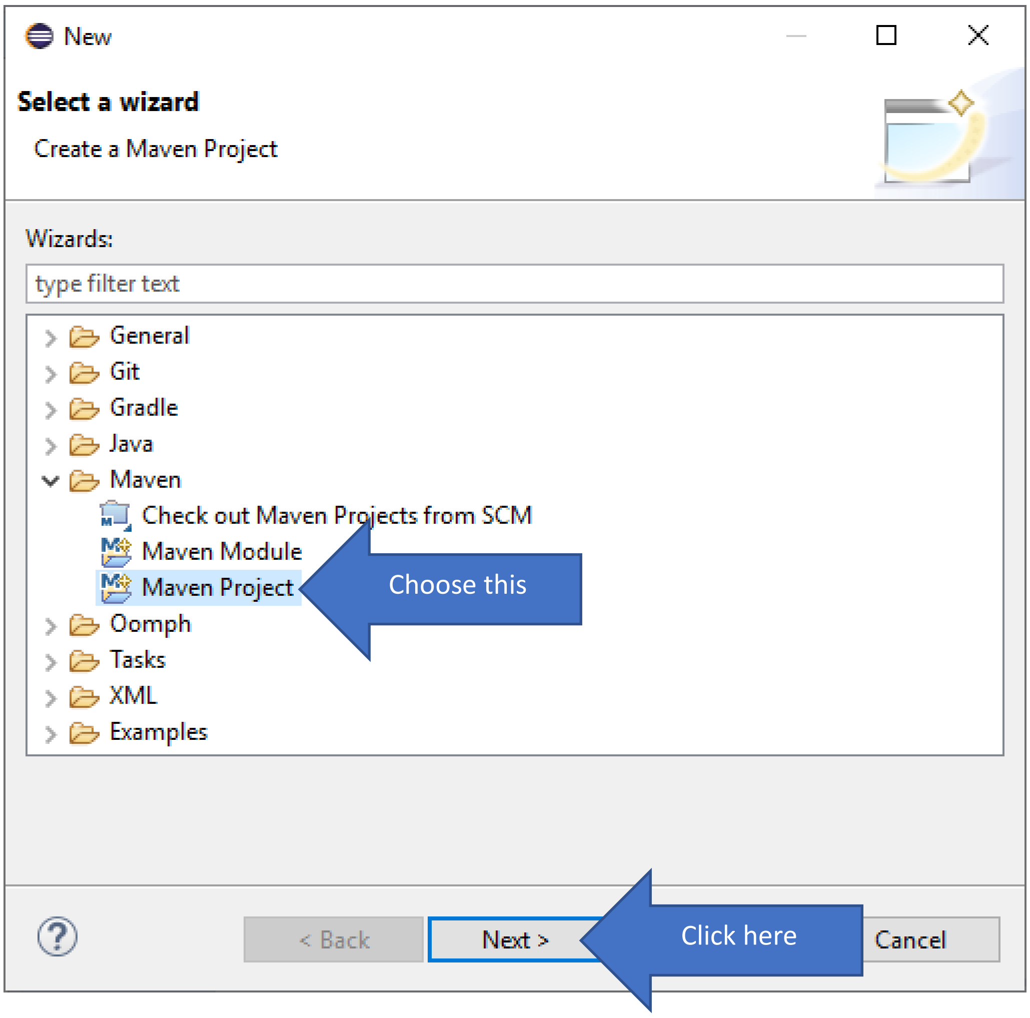
User Settings: C:\Users\mef8d\.m2\settings.xml

Local Repository (From merged user and global settings): C:\Users\mef8d\.m2\repository

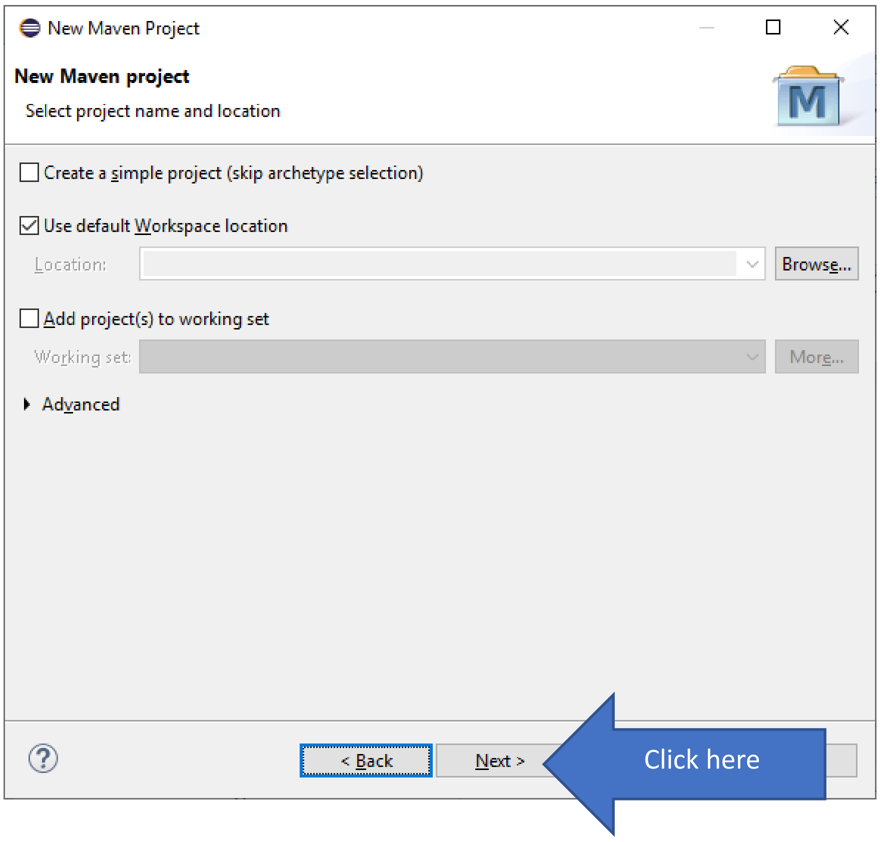
5. Click on the **Browse...** button beside the **User Settings** screen (indicated above), browse to the **M:\maven\settings.xml** file you created above, and click **Open**.
6. When you are back at the **User Settings** screen, press the **Update Settings** button. The "Local Repository" line should change to point to **M:\maven\repository** instead of whatever it was before.

Task 1: Creating a Maven project

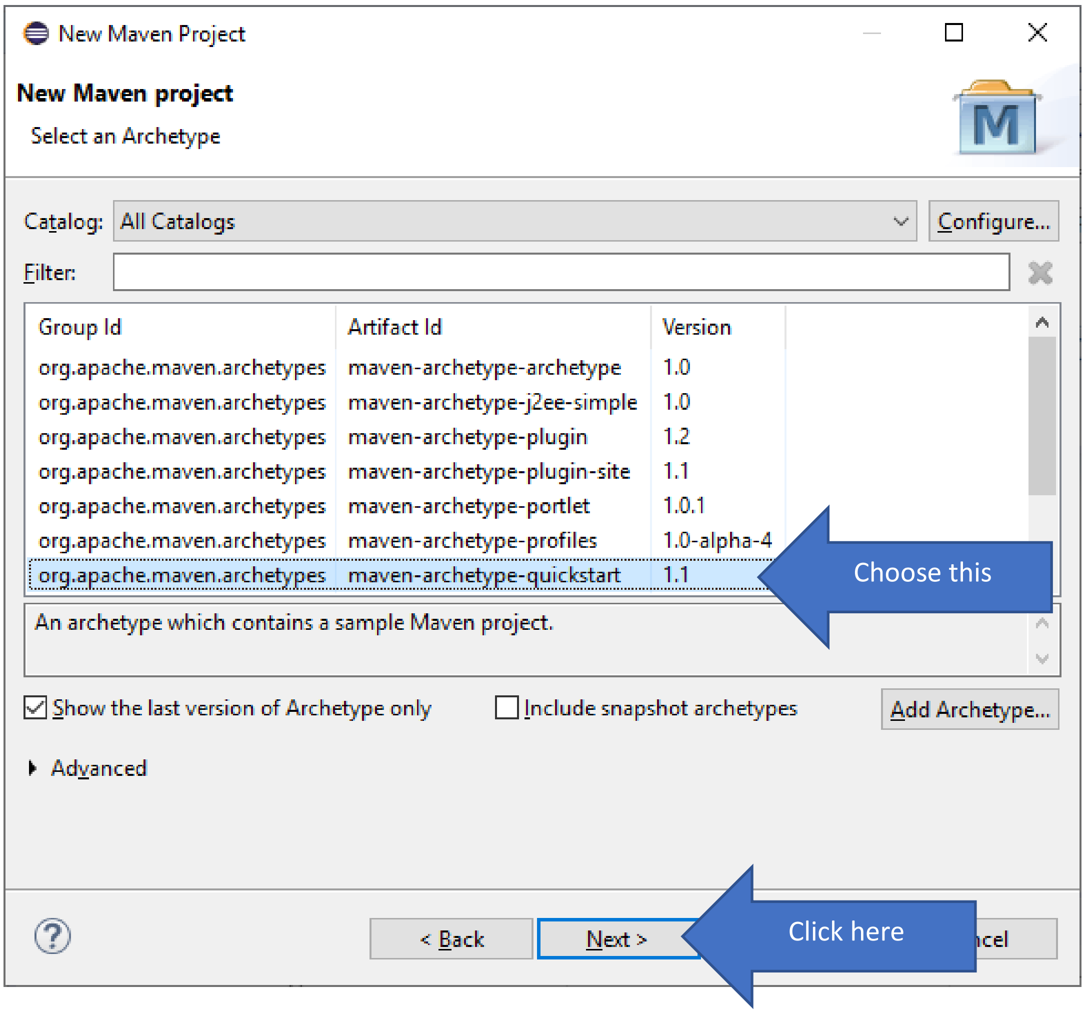
Once you have done the steps above, you are ready to create a new Maven project. In Eclipse, click **File – New – Other …** and then choose **Maven project** (as below) and click **Next**.



Choose **Next** on the next screen too (below).


New Maven Project
New Maven project
Select project name and location
Create a simple project (skip archetype selection)
Use default Workspace location
Location:
Browse...
Add project(s) to working set
Working set:
More...
Advanced
< Back
Next >
Click here

On the next screen, select **maven-archetype-quickstart 1.1** and click **Next**.


New Maven Project

New Maven project

Select an Archetype

Catalog: All Catalogs

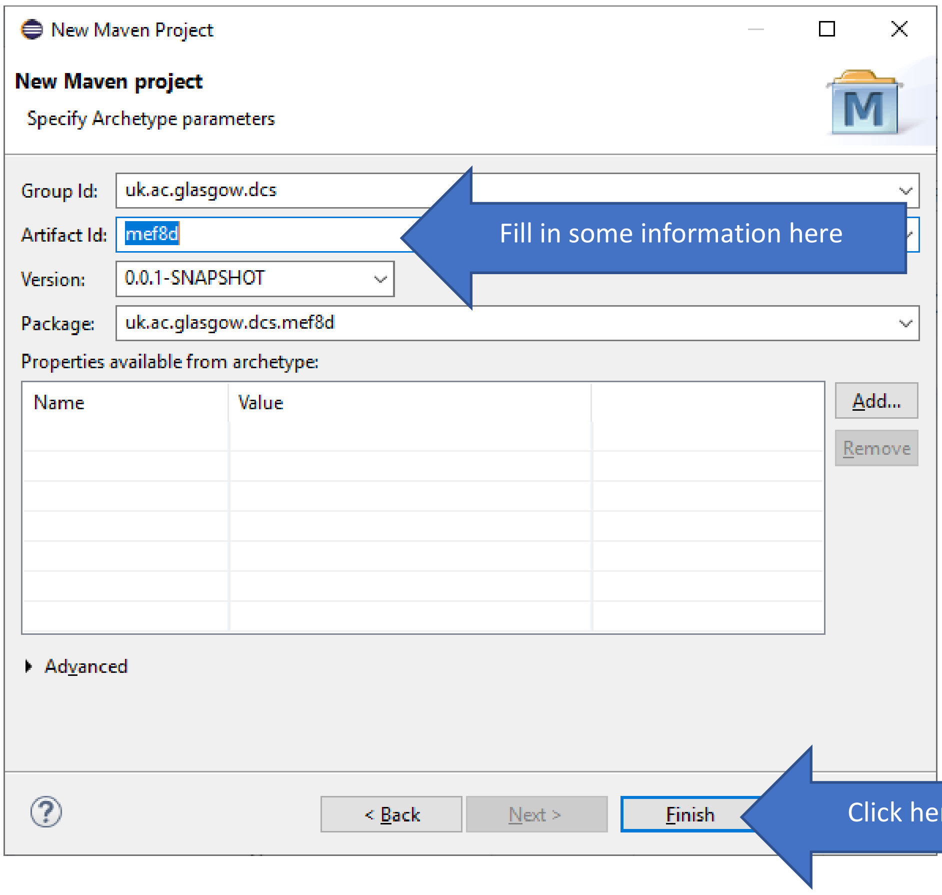
Filter:

Group Id	Artifact Id	Version
org.apache.maven.archetypes	maven-archetype-archetype	1.0
org.apache.maven.archetypes	maven-archetype-j2ee-simple	1.0
org.apache.maven.archetypes	maven-archetype-plugin	1.2
org.apache.maven.archetypes	maven-archetype-plugin-site	1.1
org.apache.maven.archetypes	maven-archetype-portlet	1.0.1
org.apache.maven.archetypes	maven-archetype-profiles	1.0-alpha-4
org.apache.maven.archetypes	maven-archetype-quickstart	1.1

An archetype which contains a sample Maven project.

[✓] Show the last version of Archetype only   [ ] Include snapshot archetypes

On the next screen, give your project whatever coordinates you want and press **Finish**.


New Maven Project

New Maven project

Specify Archetype parameters

Group Id: uk.ac.glasgow.dcs

Artifact Id: mef8d

Version: 0.0.1-SNAPSHOT

Package: uk.ac.glasgow.dcs.mef8d

Properties available from archetype:

Name	Value

The final step is to configure the **pom.xml** correctly for the current environment, as the default version refers to an older version of Java and JUnit. You should open the **pom.xml** file in your newly created Maven project and edit it so that its content is similar to the following (obviously, your **groupId** and so on will be different):

```xml
<project xmlns="http://maven.apache.org/POM/4.0.0"
	xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
	xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/xsd/maven-4.0.0.xsd">
	<modelVersion>4.0.0</modelVersion>

	<groupId>uk.ac.glasgow.dcs</groupId>
	<artifactId>mef8d</artifactId>
	<version>0.0.1-SNAPSHOT</version>

	<properties>
		<maven.compiler.release>11</maven.compiler.release>
	</properties>

	<build>
		<pluginManagement>
			<plugins>
				<plugin>
					<groupId>org.apache.maven.plugins</groupId>
					<artifactId>maven-compiler-plugin</artifactId>
					<version>3.8.1</version>
				</plugin>
			</plugins>
		</pluginManagement>
	</build>

	<dependencies>
		<dependency>
			<groupId>junit</groupId>
			<artifactId>junit</artifactId>
			<version>4.12</version>
			<scope>test</scope>
		</dependency>
	</dependencies>

</project>
```

Save the **pom.xml**, right click on the project, choose **Maven – Update project** to be sure that your changes to the POM are reflected in Eclipse.

*Note that if you are on a lab machine, you might need to do the standard project configuration hack to remove **–release** from the Java compiler at this point.*

Task 2: Building your project

You can run any of the Maven phases using Eclipse by right clicking on the **pom.xml** and choosing **Run as – Maven build** with the appropriate target. For example, you can use this to compile the project, to clean it, or to run the tests.

Task 3: Adding dependencies

One of the big advantages of using Maven as a build tool is that it makes it very easy to add external library dependencies to the code. For example, we will add the Apache Commons Collections library to our project; this library provides extended collections classes and methods beyond what the built-in java.util.collections package gives.

To add a dependency, you need to find the coordinates of the target package. The details of the package are at https://mvnrepository.com/artifact/org.apache.commons/commons-collections4, which indicates that the necessary fragment to add to **pom.xml** is as follows:

```
<!-- https://mvnrepository.com/artifact/org.apache.commons/commons-collections4 -->

<dependency>

    <groupId>org.apache.commons</groupId>

    <artifactId>commons-collections4</artifactId>

    <version>4.4</version>

</dependency>
```

Add this fragment to the **dependencies** section of your **pom.xml** and update the Maven project as before. You should now be able to write code in your project that makes use of classes such as the fragments shown in the Commons Collections user guide at https://commons.apache.org/proper/commons-collections/userguide.html. Experiment with what you can do with the new class, and also try adding other libraries if you want to do so.

Task 4: Explore

The full documentation of Maven is at https://maven.apache.org/users/index.html, and there is some more documentation of the Eclipse Maven plugin at https://www.eclipse.org/m2e/. Try doing different tasks with Maven and see how it all works!