Maven Workshop
Lab 1

1 Lab setup

Make sure you have the following items installed

- Latest version of JDK 8 / 11 (note: labs are tested with JDK 8 but should work on JDK 11 with no or minimal changes)
- Eclipse Enterprise Edition for Java (or a suitable alternative IDE for Enterprise Java)
- Latest version of Maven
- A suitable text editor (Notepad ++)
- A utility to extract zip files

In each of the main lab folders, there are two subfolders: `changes` and `final`. The `changes` subfolder just holds the source code files for the lab, while the `final` subfolder holds the complete Eclipse project starting from its project root folder. We will use the code from the `changes` subfolder to build up our applications from scratch and you can always fall back on the complete Eclipse project if you encounter any errors while building up the application.

2 Generating and using a JAR

The source code for this lab is found in `demo-jar/changes` folder.

Switch to Java SE perspective.

Create a new Java project: `SimpleFirstProject`
Create a new package: `com.workshop.operations`

In this package, create two classes:

```
StringOperations.java
MainProgram.java
```

Run `MainProgram` as normal to verify that it works (Run As -> Java Application).

Select project and Export as a Runnable JAR file.
Select a suitable destination folder and name the JAR as: `StringOperations.jar`
For the Launch configuration, select `MainProgram – SimpleFirstProject`.
Click Finish.

In a command prompt or shell terminal, navigate to the destination folder and run the JAR with:

```
java -jar StringOperations.jar
```

Using 7-z to open the JAR, verify that it contains the compiled classes from the project as well as a manifest (MANIFEST.MF) which indicates the main class to run. You can also unzip the JAR to view its contents.

Create another Java project: `SimpleSecondProject`
Create a new package: `com.workshop.user`

In this package, create a class: `UserProgram.java`

Notice that it has a syntax error as the StringOperations class is not on its project build path.
To solve this problem, we need to add the JAR that we generated earlier to its build path.

In `SimpleSecondProject`, create a new folder `lib`. Copy and paste `StringOperations.jar` into this folder.

On the project, select Properties -> Java Build Path
Right click Libraries Tab, select Add JARs. Select the JAR file that you just pasted into lib. Click Apply and Close.

Notice that you can expand the JAR file to see its contents in the Referenced Libraries entry in the Package Explorer.

The syntax error in `UserProgram` should disappear and you should be able to run it successfully.

In this simple example, `StringOperations.jar` is a dependency of `SimpleSecondProject`

3 Using JARs from external libraries (Log4J and JUnit)

The source code for this lab is found in `demo-log4j-junit/changes` folder.

Switch to Java SE perspective.

Create a new Java project: `SimpleLoggingProject`

Create a new package: `com.workshop.operations`
In this package, create a class: `BasicLoggingDemo`

In the project, create a folder `properties` and create the file `log4j.properties` in it.

Create a new package: `com.workshop.test`
In this package, create two classes:
`TestJunit`
`TestRunner`

Notice that there are syntax errors again in both classes as the required classes from the Log4J and JUnit library are not on the build class path yet.

In the project, create a new folder `lib`.

Download the binary distribution for Log4j (either zip for Windows or tar.gz for Linux) from this link and unzip into a folder. Use 7-Zip to unzip as this is faster than the built-in Windows zip extractor
https://logging.apache.org/log4j/1.2/download.html

Copy and paste `log4j-1.2.17.jar` into the `lib` folder. There is a copy of this in the `changes` folder as well if the download is too slow.

Download the JAR files for JUnit from this link below:
https://github.com/junit-team/junit4/wiki/Download-and-Install

Click on the links for `junit.jar` and `hamcrest-core.jar` to redirect to the Maven Central Repository search to download them. Copy and paste these two JAR files into the `lib` folder.
There is a copy of these two files in the `changes` folder as well if the download is too slow.

On the project, select Properties -> Java Build Path
Right click Libraries Tab, select Add JARs. Select all the 3 JAR files that you just pasted into `lib`. Click Apply and Close.

Notice that you can expand any of these JAR files to see its contents in the Referenced Libraries entry in the Package Explorer.

To run `BasicLoggingDemo`, we need to place the `log4.properties` file onto the runtime classpath during execution.

Right-click on the project and select Run As -> Run Configurations.
Double click on the Java application entry
In the Name field, type: `Run with properties`
In the Main tab, click on Search for the Main class: entry and select BasicLoggingDemo
Go to the Classpath tab and select SimpleLogging Project. Click on Advanced button.
Select the Add Folders option and click on OK button.
Browse for the folder `properties` which contains the `log4j.properties` file
Select that folder and click on OK, Apply then Run button.

You should now be able to see the log output from `BasicLoggingDemo`

```
From BasicLoggingDemo
[main] DEBUG com.workshop.operations.BasicLoggingDemo  - Hello this
is a debug message
  [main] INFO   com.workshop.operations.BasicLoggingDemo   - Hello
this is an info message
```

To see the JUnit test in action, go to `TestRunner` and do Run As -> Java Application
Try changing the strings being compared in `TestJUnit` to see how an error is flagged.