Worksheet 1: Getting Started (not assessed)

Key Objectives:

1) Install java & Maven
2) Learn to create and run a Java Maven Project
3) Learn to create and run an ASTRA Maven Project

Part 1: Setup

a) Start by installing a JDK on your machine. I recommend using an older Java version (e.g. Java 8 or 11). If you are using the new Apple Silicon chips, you should use the Azul JDKs (https://www.azul.com/downloads/?package=jdk)

b) Now install maven (basic instructions here: https://maven.apache.org/install.html).
Windows Guide: https://mkyong.com/maven/how-to-install-maven-in-windows/
MacOS Guide: https://mkyong.com/maven/install-maven-on-mac-osx/

Alternative: https://www.baeldung.com/install-maven-on-windows-linux-mac

---

Part 2: Create a Java Maven Project

In this part of the worksheet, we will create the Hello World project using Maven.

a) Create the folder structure: Create a folder called “hello” (this is the project root folder & the name should be the same as the project name) and create the following sub-folder structure: “src”->”main”->”java”. The “java” folder is the place where you put your Java code.

b) Create a pom.xml file in the project root folder and copy the xml below into it:

```
<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>examples</groupId>
    <artifactId>hello</artifactId>
    <version>0.1.0</version>
</project>
```

This is a basic pom file for a project. The <artifactId> property is the name of the project. By convention, we use the same name as we used for the project root folder. This is not required, but is helpful when trying to identify the different projects you have created.

The <groupId> property is used to identify related projects. Here we call it “examples”. Normally, you use an inverted domain name (e.g. developing for “astralanguage.com” would require a groupId “com.astralanguage”).

c) Now create a file called Hello.java in the src/main/java. The contents should be:

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

This is the standard “Hello World” program

d) To compile the code, open a terminal (command prompt). Go to the project root folder (where the pom.xml file is located). Type:

```
$mvn compile
```

e) Running the code. Maven is a build system. In its basic form, support for running code is not provided. To run the code, you should run the command:

```
$mvn package
```

This creates a file called hello-0.1.0.jar in the target folder. The “0.1.0” comes from the <version> property and the “hello” comes from the <artifactId> property.

To run the program, type:

```
$java -cp target/hello-0.1.0.jar Hello
```

This is quite inconvenient, and becomes more inconvenient as the projects you write become more complex. Luckily, somebody has created a plugin for Maven (an extension) that allows you to run a java program using maven. To do this, modify the pom.xml file to look like the xml below:

```
<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>examples</groupId>
    <artifactId>hello</artifactId>
    <version>0.1.0</version>

    <build>
        <plugins>
            <plugin>
                <groupId>org.codehaus.mojo</groupId>
                <artifactId>exec-maven-plugin</artifactId>
                <version>1.6.0</version>
                <configuration>
                    <mainClass>Hello</mainClass>
                </configuration>
            </plugin>
        </plugins>
    </build>
</project>
```

Notice the <mainClass> property matches the name of the Java class we want to run.

```
$mvn exec:java
```

The “exec” part comes from the name of the plugin (*exec*-maven-plugin). The java part is a goal associated with the plugin that can be used to run the Java program. Plugins can have multiple goals (https://www.mojohaus.org/exec-maven-plugin/usage.html).

Part 3: Create an ASTRA Maven Project

The final part of the worksheet involves creating an ASTRA maven project (and writing your first ASTRA program).

a) Create the folder structure: Create a folder called “astra-hello” (again, this is the project root folder & the name should be the same as the project name) and create the following sub-folder structure: “src”->”main”->”astra”. The “astra” folder is the place where you put your ASTRA code.

b) Create a pom.xml file in the project root folder and copy the xml below into it:

```
<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>examples</groupId>
    <artifactId>astra-hello</artifactId>
    <version>0.1.0</version>

    <parent>
        <groupId>com.astralanguage</groupId>
        <artifactId>astra-base</artifactId>
        <version>1.3.2</version>
        <relativePath></relativePath>
    </parent>
</project>
```

Notice the introduction of the <parent> section. This is a pre-written maven project that contains all the necessary configuration for running an ASTRA program. The project is available through Maven Central (https://search.maven.org/artifact/com.astralanguage/astra-interpreter). It includes a plugin developed to support compiling and running of ASTRA programs (similar to the exec-maven-plugin we saw in the previous part of the worksheet). The source XML for the project can be found here: https://gitlab.com/astra-language/astra-core/-/blob/master/astra-base/pom.xml.

c) Create a file called Hello.astra in the src->main->astra folder. Copy the code below into it:

```
agent Hello {
    module Console console;

    rule +!main(list args) {
        console.println("Hello, World! ");
    }
}
```

d) To compile the code, use the “mvn compile” option you used before (the ASTRA plugin is integrated with this phase of the build cycle). Once you run this command, look in the “target” folder, you should see a “gen” subfolder that contains a file called “Hello.java” – this code is generated based on the ASTRA code you wrote in part (c).

e) To run the ASTRA program, modify the pom.xml as follows:

```
<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>examples</groupId>
    <artifactId>astra-hello</artifactId>
    <version>0.1.0</version>

    <properties>
        <astra.main>Main</astra.main>
    </properties>

    <parent>
        <groupId>com.astralanguage</groupId>
        <artifactId>astra-base</artifactId>
        <version>1.3.2</version>
        <relativePath></relativePath>
    </parent>
</project>
```

The <astra.main> property is used to identify which ASTRA program you want to run. An instance of this agent is created with the default name "main" when you run the program. To change the default name, you can use the <astra.name> property.

To run the program, type:

```
$mvn astra:deploy
```

You should see "Hello, World!" printed out to the console. It will be mixed in with the Maven output.