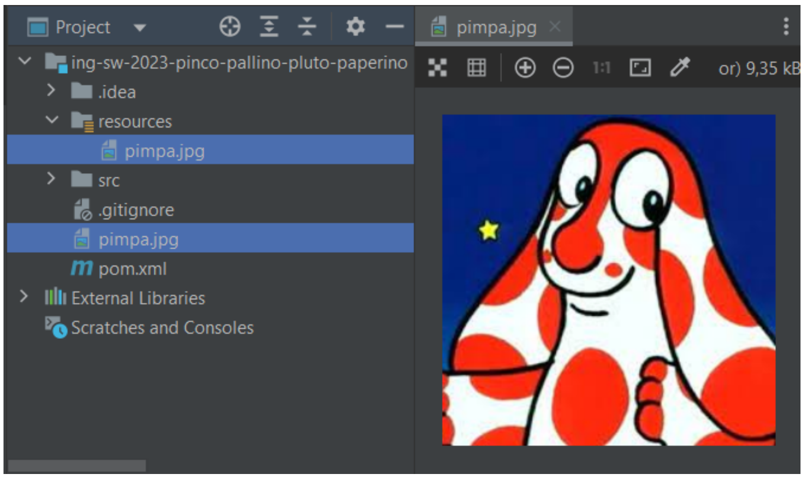
How to create the jar file for the Software Engineering project

Authors: Davide Ettori & Samuele Giammusso
Year: 2023

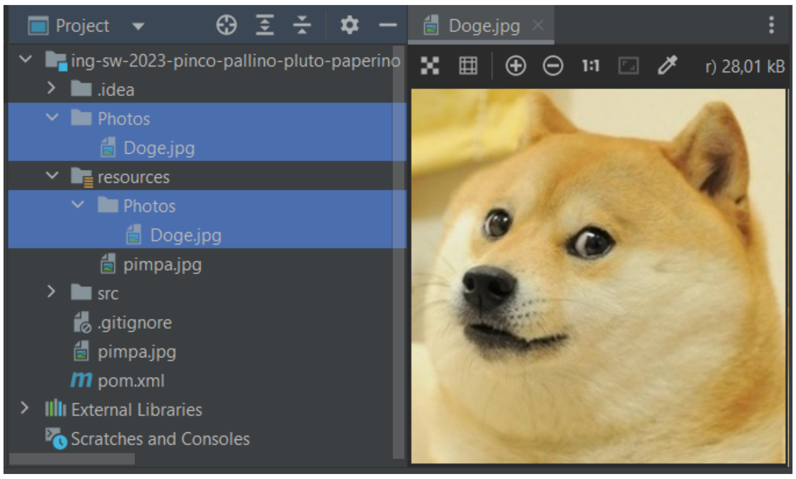
From now on, we will assume that you start with the pom.xml file given by the professors on the WeBeep folder of the course.

Step 1

Put the files in the correct places so that when you execute it, inside IntelliJ or from the Jar file, they both can correctly find the file using the same file path, even if they are in different folders. But where should you put them?
Let's say for example that you want to use the "**pimpa.jpg**" image in your project:
-Create the "**resources**" folder inside the project root
-Add the file both in the project root and in the resources folder
You should obtain something like this:



If you have folders with many files inside, you can do the same:
Here for example we have the folder "**Photos"** with "**Doge.jpg"** inside



Now, if you want to use those files, you can refer to them simply by using this paths:
-For the pimpa image: **pimpa.jpg**
-For the Doge image: **Photos/Doge.jpg**

Unfortunately, now you might need to change the way you access the resources files for your project because executing the jar is not the same as executing the project from intellij.

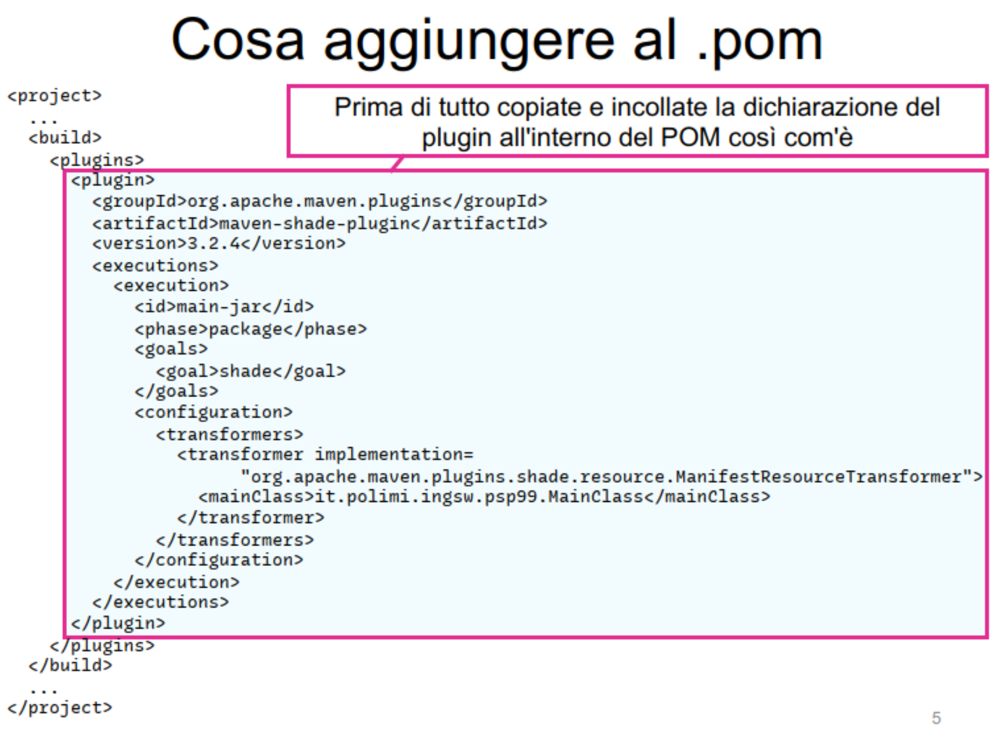
1) Images → when you access images you need to substitute “path/to/image.png” with: getClass().GetClassLoader().getResource(“path/to/image.png”);
2) Audio → when you want to play audios you have to substitute ”path/to/audio.wav” with:
   new BufferedInputStream(getClass().getResourceAsStream(”path/to/audio.wav”));
3) .txt → if you need to use .txt files (persistence, for example) you can create them in the following way: new File(“filename.txt”).createNewFile();
   the file will be created in the same folder of the .jar file (outside of the jar)
   You can read those files with:
      new BufferedReader(new FileReader(“filename.txt”)).readLine();
   You can write those files with:
      new BufferedWriter(new FileWriter(“filename.txt”)).write(String s);
   If you need to read or write an Object, then you could use the following:
      new File(Input/Output)Stream(“filename.txt”);

Bonus Tips: If you are using Swing for the GUI, don’t use empty file paths (like this: “”) because they only work in intellij and not in the Jar file (or else you will see an exception when trying to execute the jar file).
For example don’t do this: **JLabel i = new JLabel(new ImageIcon("not_existing_file"));** because in intellij it will just not display the image and keep running, while the jar file will throw a NullPointerException at runtime.

Step 2

You have to add to the **pom.xml** file what is needed in order to correctly build the Jar file:
-The tutors of the project should have already given you the template for the pom, so you can start from that.

1) Let's start by adding the **Maven-Shade-Plugin** to the pom:



Cosa aggiungere al .pom

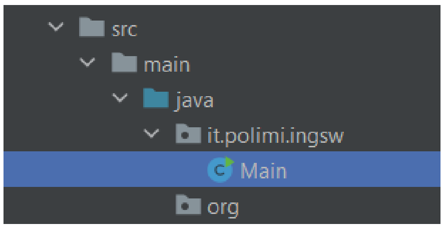
Prima di tutto copiate e incollate la dichiarazione del plugin all'interno del POM così com'è

```
<project>
  ...
  <build>
    <plugins>
      <plugin>
        <groupId>org.apache.maven.plugins</groupId>
        <artifactId>maven-shade-plugin</artifactId>
        <version>3.2.4</version>
        <executions>
          <execution>
            <id>main-jar</id>
            <phase>package</phase>
            <goals>
              <goal>shade</goal>
            </goals>
            <configuration>
              <transformers>
                <transformer implementation=
                      "org.apache.maven.plugins.shade.resource.ManifestResourceTransformer">
                  <mainClass>it.polimi.ingsw.psp99.MainClass</mainClass>
                </transformer>
              </transformers>
            </configuration>
          </execution>
        </executions>
      </plugin>
    </plugins>
  </build>
  ...
</project>
```

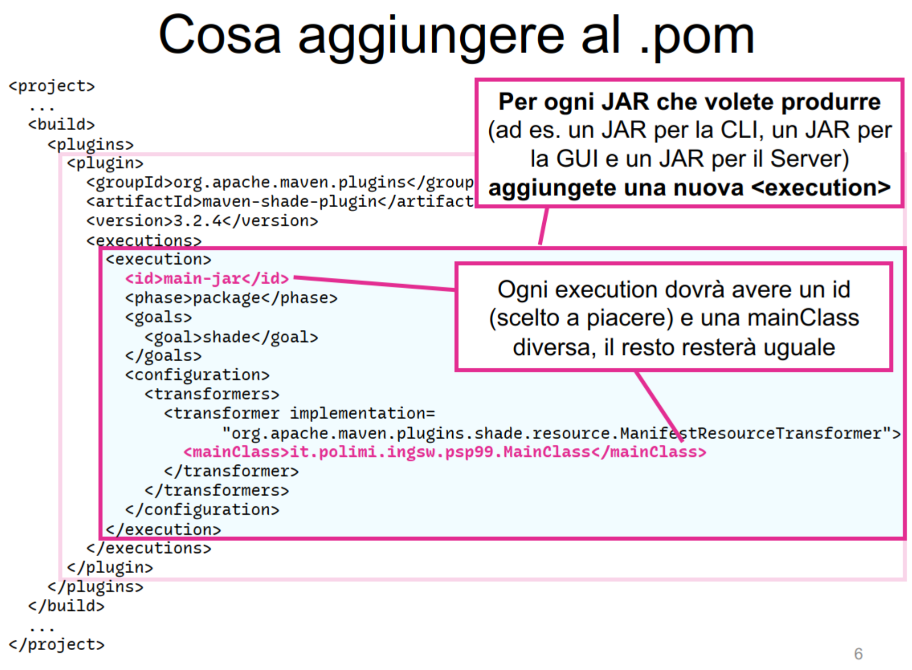
5

*this image was taken from the previous year (2022) slides

Remember to specify the correct path to your main class,
(in this case where it says **<mainClass>it.polimi.ingsw.psp99.MainClass</mainClass>**)
-In my example the path would be: **<mainClass>it.polimi.ingsw.Main</mainClass>**



Bonus Tips: If you have more than one main class, let's say for Example one main class for the Client + one main class for the Server, you should duplicate this code and specify the two different main classes. Like Below:



Cosa aggiungere al .pom

```
<project>
  ...
  <build>
    <plugins>
      <plugin>
        <groupId>org.apache.maven.plugins</group
        <artifactId>maven-shade-plugin</artifact
        <version>3.2.4</version>
        <executions>
          <execution>
            <id>main-jar</id>
            <phase>package</phase>
            <goals>
              <goal>shade</goal>
            </goals>
            <configuration>
              <transformers>
                <transformer implementation=
                      "org.apache.maven.plugins.shade.resource.ManifestResourceTransformer">
                  <mainClass>it.polimi.ingsw.psp99.MainClass</mainClass>
                </transformer>
              </transformers>
            </configuration>
          </execution>
        </executions>
      </plugin>
    </plugins>
  </build>
  ...
</project>
```

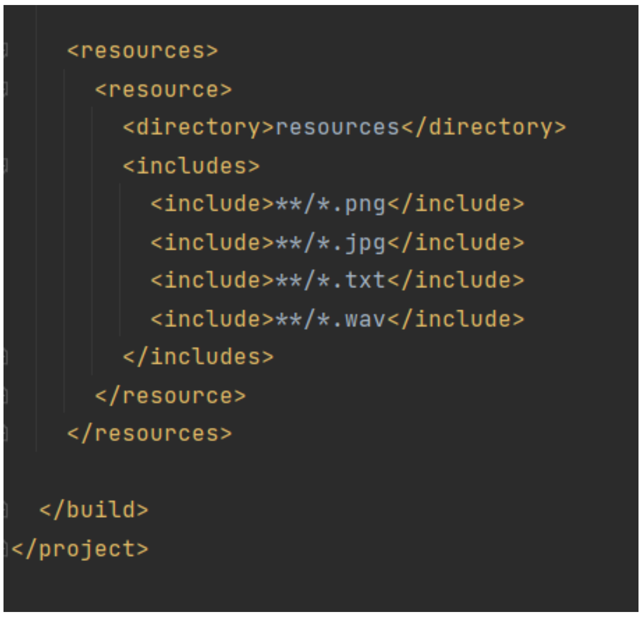
Per ogni JAR che volete produrre (ad es. un JAR per la CLI, un JAR per la GUI e un JAR per il Server) **aggiungete una nuova <execution>**

Ogni execution dovrà avere un id (scelto a piacere) e una mainClass diversa, il resto resterà uguale

6

*this image was taken from the previous year (2022) slides

2) Let’s specify where the Jar should find the files:

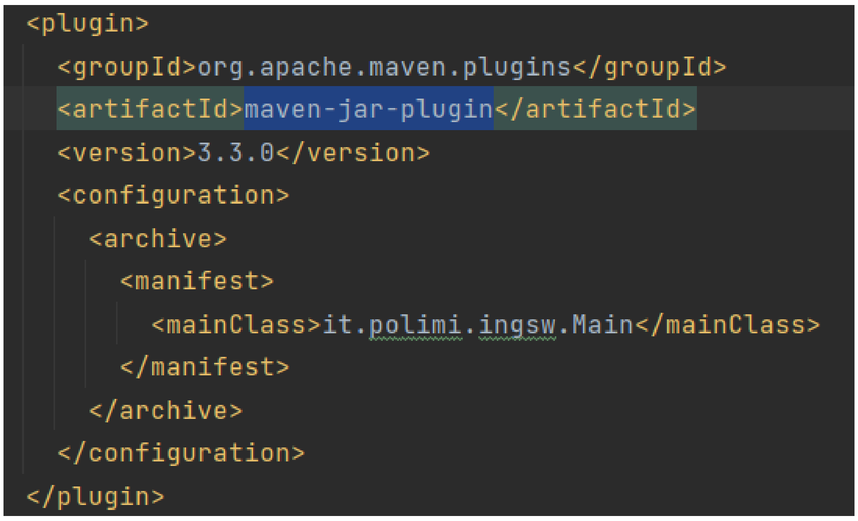


```
    <resources>
      <resource>
        <directory>resources</directory>
        <includes>
          <include>**/*.png</include>
          <include>**/*.jpg</include>
          <include>**/*.txt</include>
          <include>**/*.wav</include>
        </includes>
      </resource>
    </resources>

  </build>
</project>
```

The Jar file will find the files in the resources folder that you created earlier. You also need to specify the extensions of the files that you use in the project.

3) Then you need to specify again the main class of the project (or elsewhere when you try to execute the Jar, it won't find the Main class).



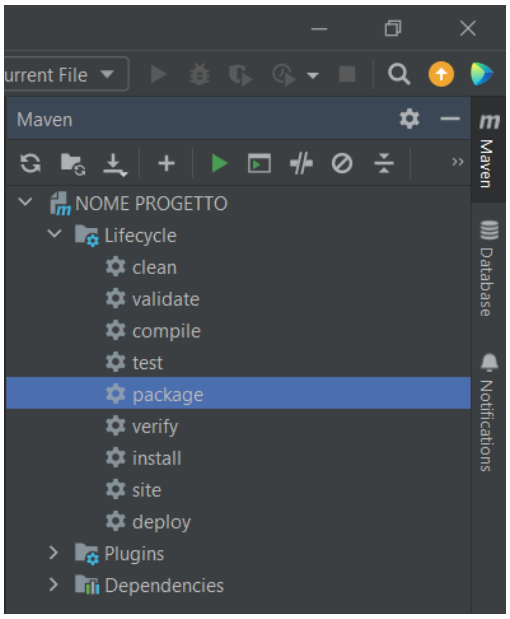
```
<plugin>
  <groupId>org.apache.maven.plugins</groupId>
  <artifactId>maven-jar-plugin</artifactId>
  <version>3.3.0</version>
  <configuration>
    <archive>
      <manifest>
        <mainClass>it.polimi.ingsw.Main</mainClass>
      </manifest>
    </archive>
  </configuration>
</plugin>
```

As you can see, in order to do that, you have to modify a part of the pom.xml that was already given in the template: find where it says '**maven-jar-plugin**', and add the above line that specifies the Main class of your project.

If you are using additional libraries such as Gson, JavaFX, … you must specify them in the pom.xml as external dependencies. You can do that the same way the JUnit library is imported in the standard pom.xml file given by the professors (you just have to edit the tags which identify the library you need). Remember to update Maven, at the end of this procedure, using the button which will pop-up in the top-right section of the screen.

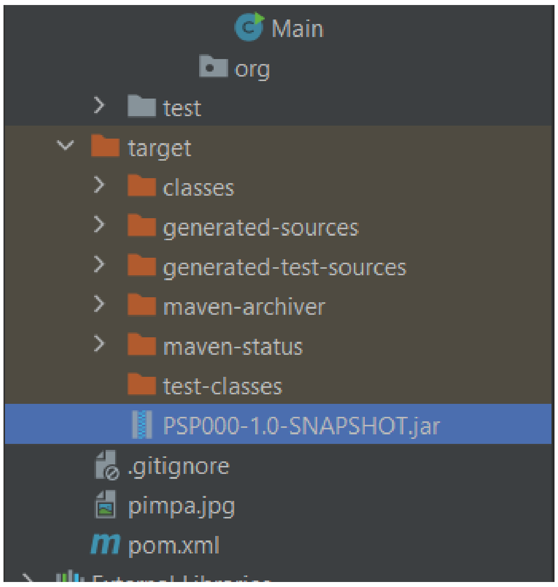
Step 3

Finally now you can create the Jar file:
In order to do that, click on the top-right side of the intelliJ window, where it says **'maven'**, and then double click **'package'**.



At this point the Building process of the Jar should have started and should end successfully.

Now you should find the jar file in the target folder:



Step 4

Now you can execute the jar file:
(Keep in mind that you can rename all the jars files as you please: client.jar, server.jar, …)
-Using the File Explorer go inside the **target** folder
-open a terminal window inside that folder (right click and choose 'Open in Terminal')
-type the command: **java -jar NAMEOFTHEJAR.jar**
(remember that you have to download and install Java from the Oracle website if you want to run this command)

Bonus Tips: If, for some reason, when you try to execute the jar the terminal tells you that it cannot find the main class (shame on you that you didn't follow my guide!):
-try this command: **java -cp NAMEOFTHEJAR.jar it.polimi.ingsw.Main**
This command specifies which is the main class that the jar should execute.

Step 5

Finally, have you ever wondered how to browse safely and securely without advertising ? Use the **SudVPN** app and enjoy a secure, encrypted internet connection, as well as extra protection from online threats, for the same price. Now you can use the code "Etto-Giammu" for 83% discount on the standard plan and a free year trial. **SudVPN**, protect your internet browsing.