

Cómo empaquetar tu aplicación Java en un fichero JAR

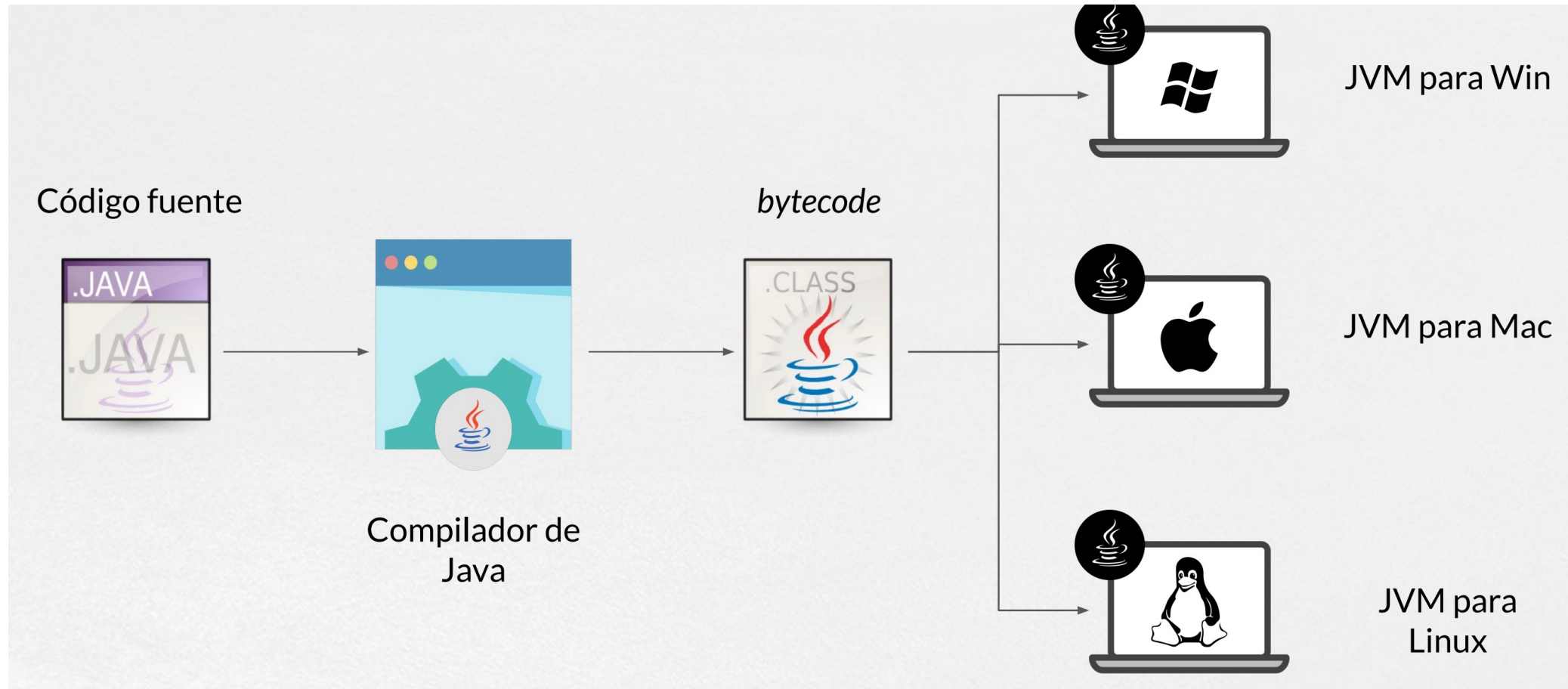
Joaquín Franco Ros

CARACTERÍSTICAS DE UN FICHERO .JAR

Java bytecode

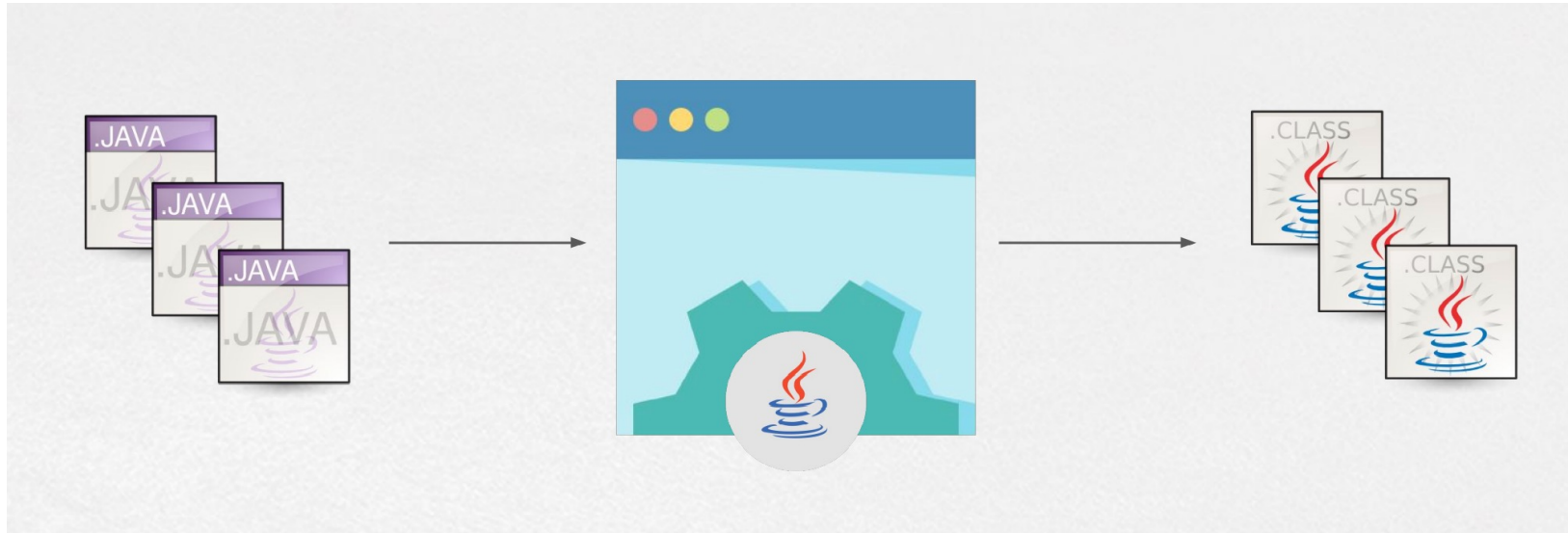
- Java es compilado e interpretado.
- El código fuente de Java se compila a *bytecode*.
- El código intermedio *bytecode* es interpretado por la JVM.(Java Virtual Machine).

Java y bytecode



Java y bytecode

- Generalmente se genera un fichero .class por cada fichero .java.

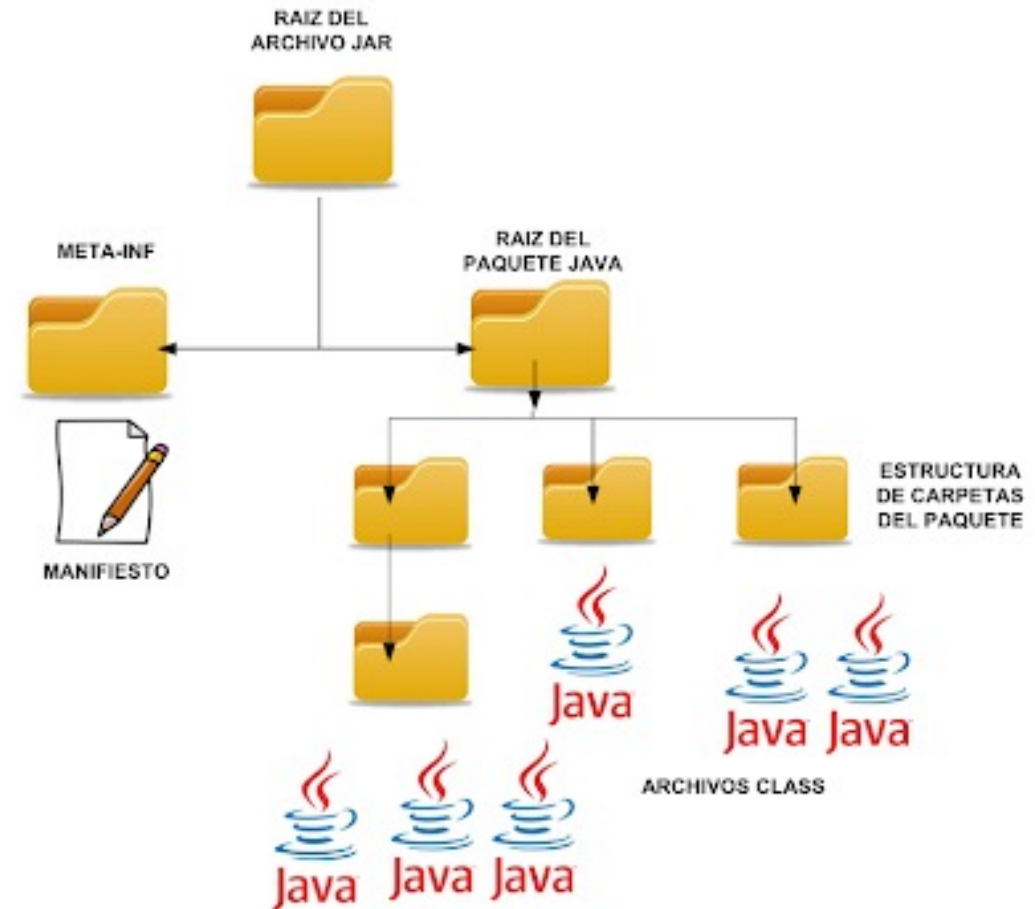


Fichero .jar

- ¿Cómo distribuir una aplicación Java con miles de clases?
- Fichero JAR
 - Un fichero comprimido
 - Comprimido con zip
 - Capaz de ser ejecutado por la JVM
- Se ejecutar con el comando:
 - `java -jar app.jar`

Estructura de un fichero .jar

- Clases
- Otros recursos (otras librerías, imágenes, etc)
- **Fichero de manifiesto.**

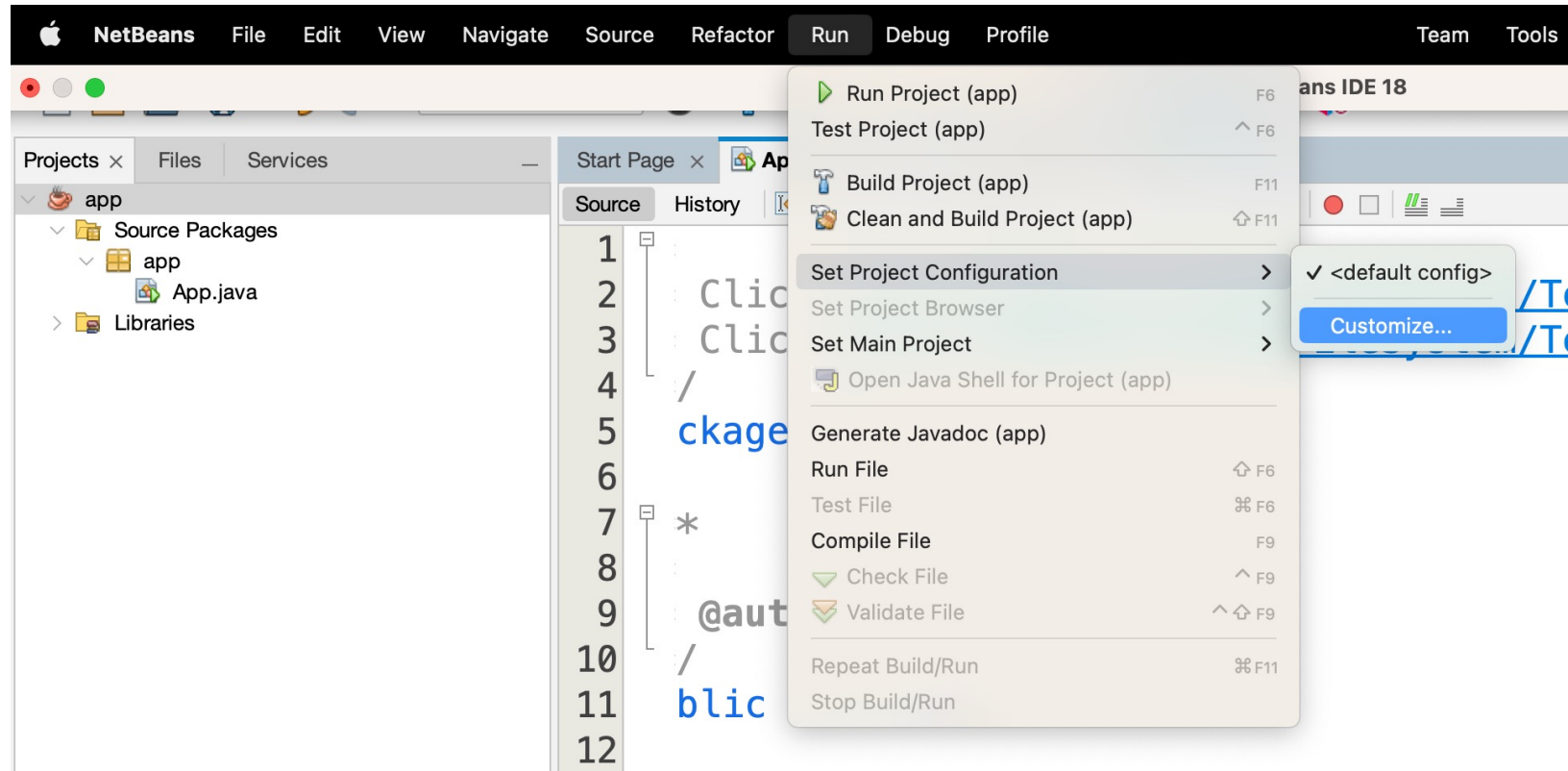


Fichero de manifiesto

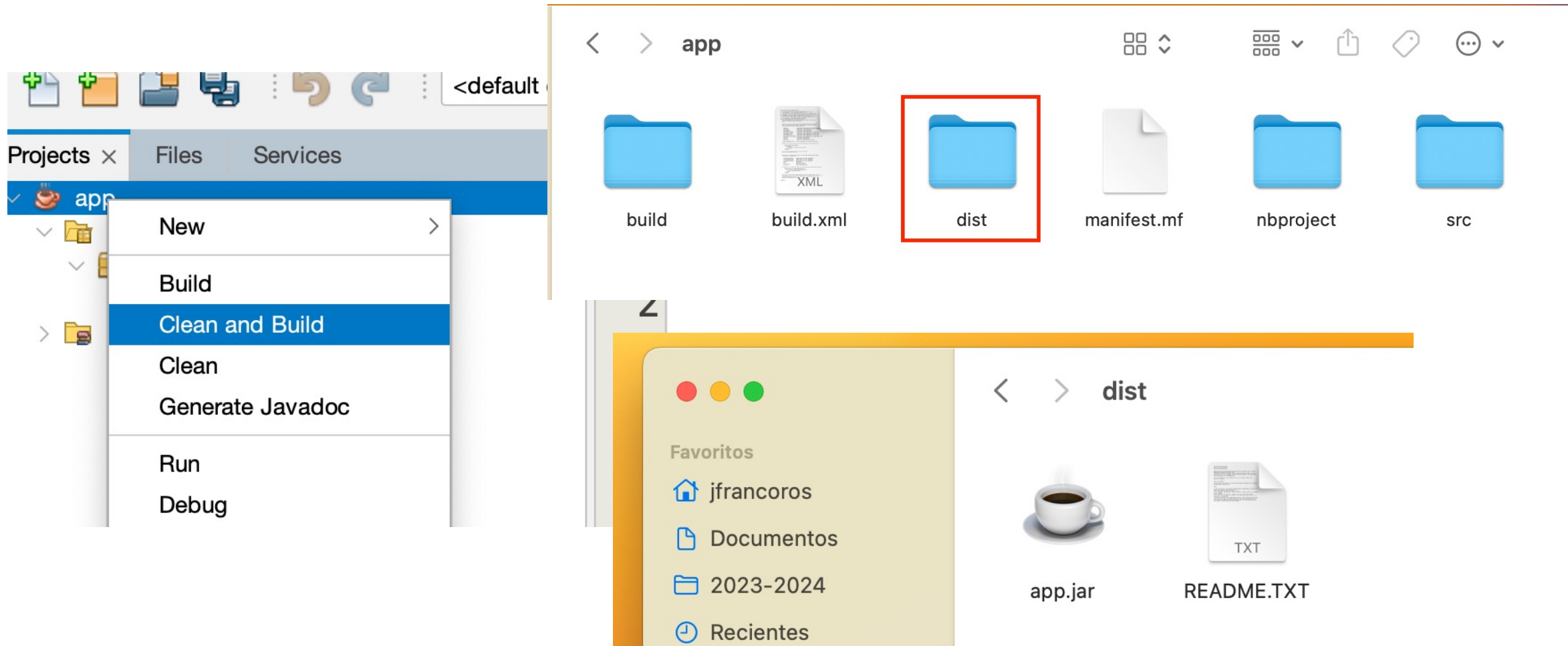
- Permite describir el fichero JAR.
- Atributos tipo Clave-Valor.
- Más habituales:
 - *Manifest-Version*
 - Main-Class: indica una clase con método main para ejecutar
 - *Class-Path*: rutas para buscar otras clases o paquetes

CREAR UN FICHERO .JAR
CON NETBEANS Y ANT

Crear un fichero .jar con netbeans

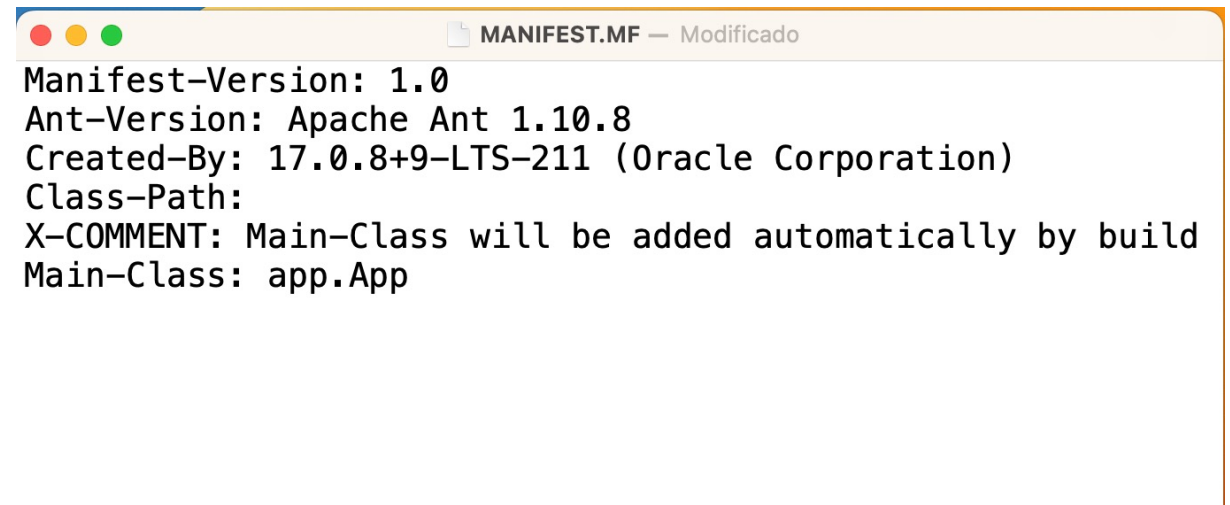


Crear un fichero .jar con netbeans



Descomprimiendo un fichero .jar

- Realmente un fichero .jar es un archivo comprimido que se puede descomprimir y se puede ver el interior:



Ejecutar un fichero .jar desde línea de comandos

- Se hace con el comando:
 - `java -jar fichero.jar`

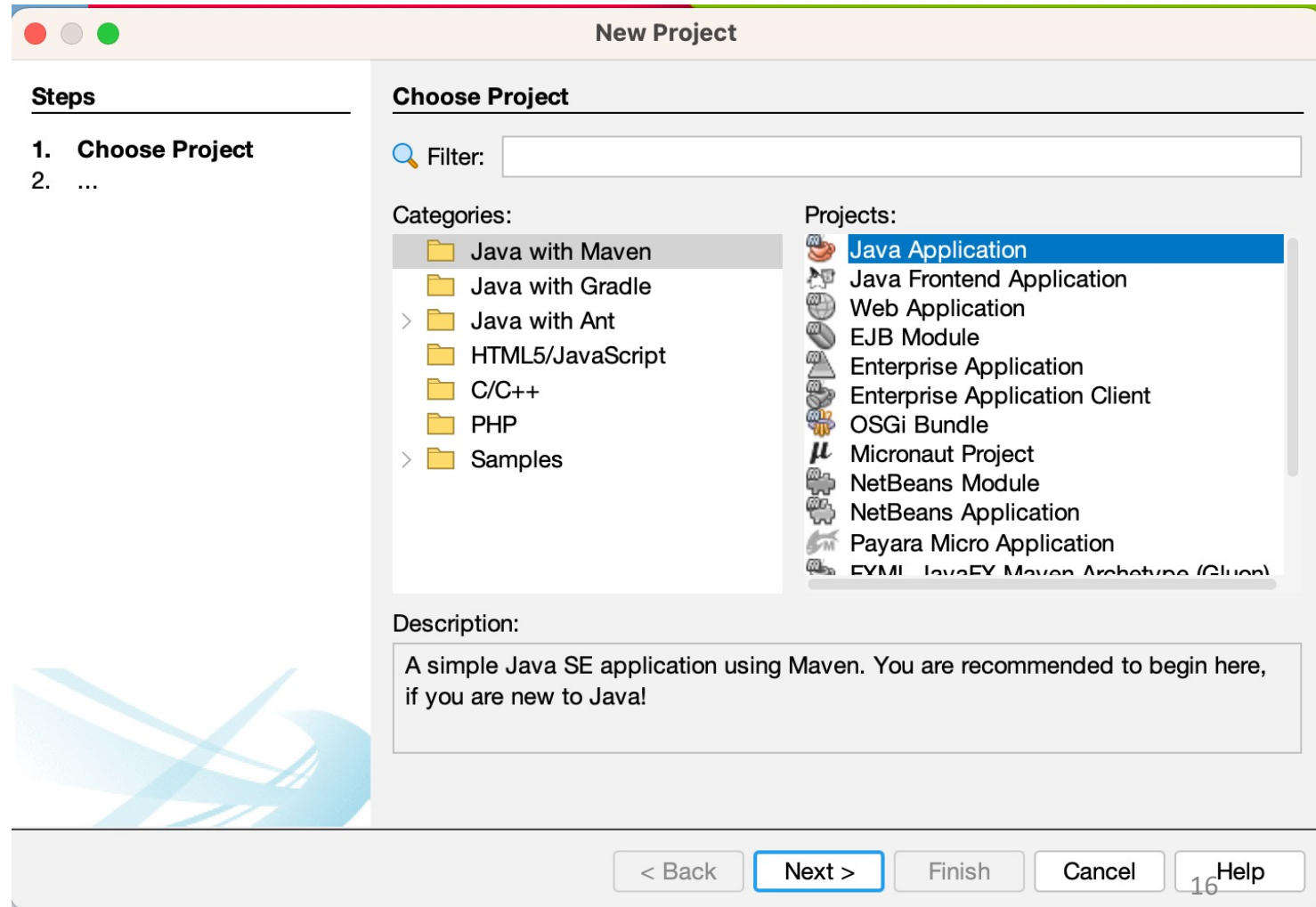
```
jfrancoros at MacBook-Pro-de-Joaquin in ~/Documents/2023-2024/PSP/EvaluaciónInicial/GenerarJar/app/dist
$ java -jar app.jar
Hola mundo
```

GENERAR UN .JAR CON NETBEANS Y MAVEN

MAVEN

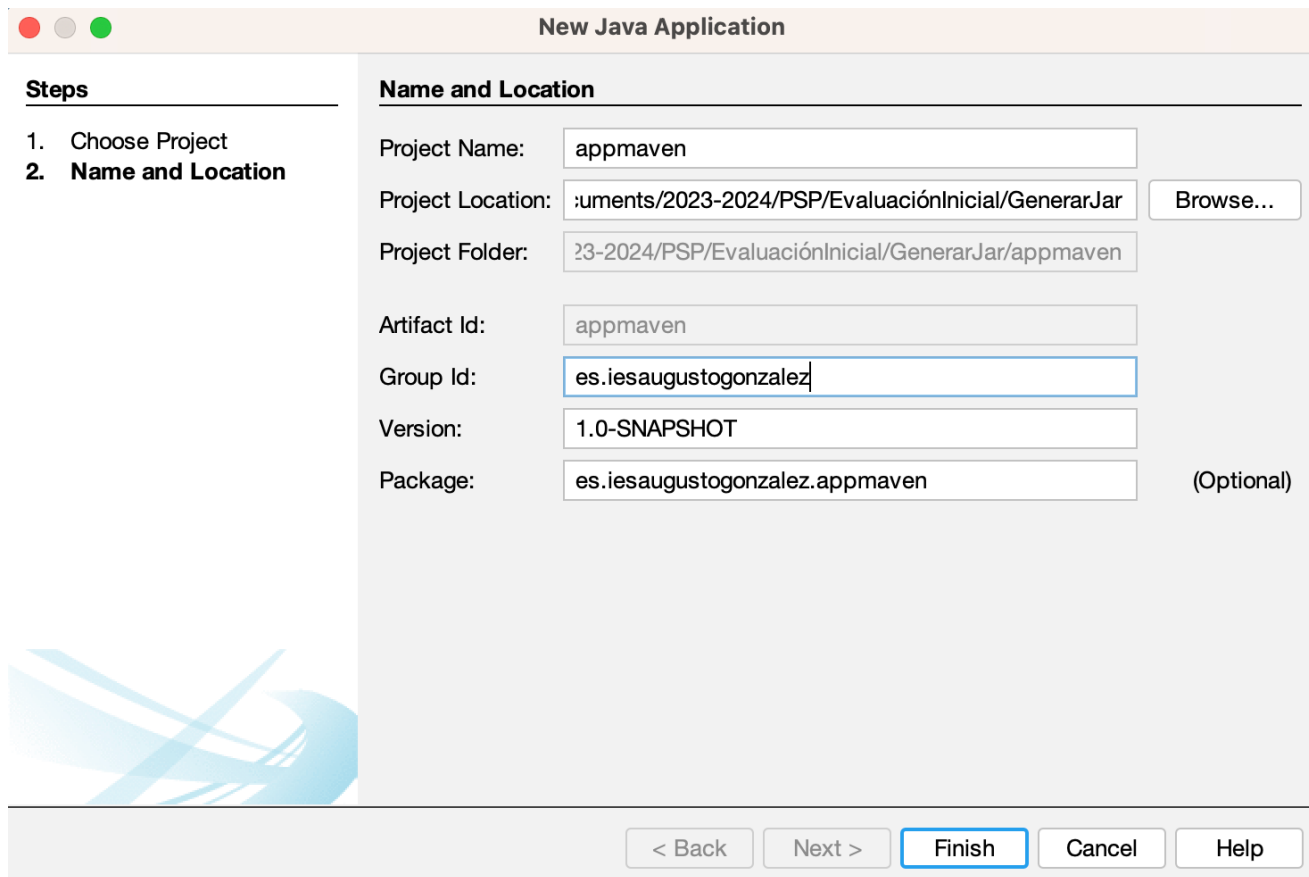
- Herramienta para gestionar proyectos Java
- Nos ayuda a generar dependencias con otros artefactos (librerías externas)
- Múltiples *plugins* para realizar muchas tareas
 - Por ejemplo, generar un .jar
- Está integrado en la mayoría de IDEs de Java

Proyecto Maven con NetBeans



Proyecto Maven con Netbeans

- Paquete es.iesaugustogonzalez.appmaven



New Java Application

Steps

1. Choose Project
2. **Name and Location**

Name and Location

Project Name: appmaven

Project Location: :uments/2023-2024/PSP/EvaluaciónInicial/GenerarJar Browse...

Project Folder: 23-2024/PSP/EvaluaciónInicial/GenerarJar/appmaven

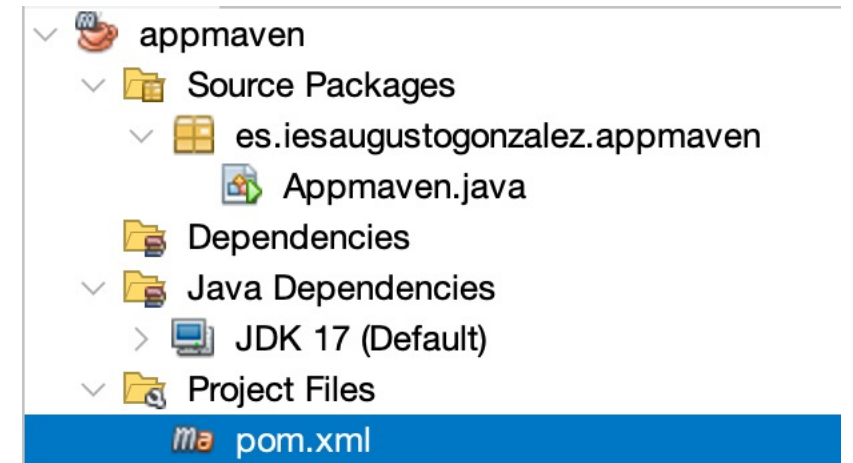
Artifact Id: appmaven

Group Id: es.iesaugustogonzalez

Version: 1.0-SNAPSHOT

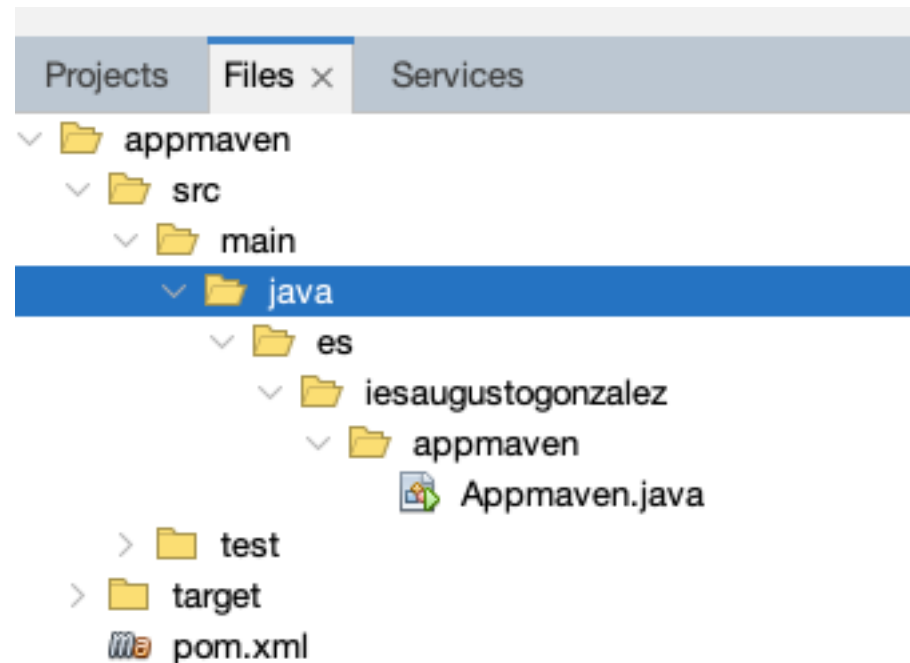
Package: es.iesaugustogonzalez.appmaven (Optional)

< Back Next > **Finish** Cancel Help



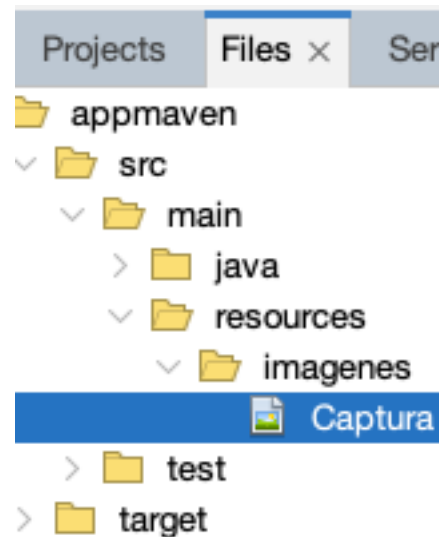
Proyecto Maven con Netbeans

- La estructura de paquetes cambia totalmente



Añadir recursos al fichero .jar con maven

- Si queremos añadir recursos como imágenes, audios, etc. Es necesario que estos estén en una carpeta llamada **resources** en esta ruta (si no existe hay que crearla):



Fichero pom.xml

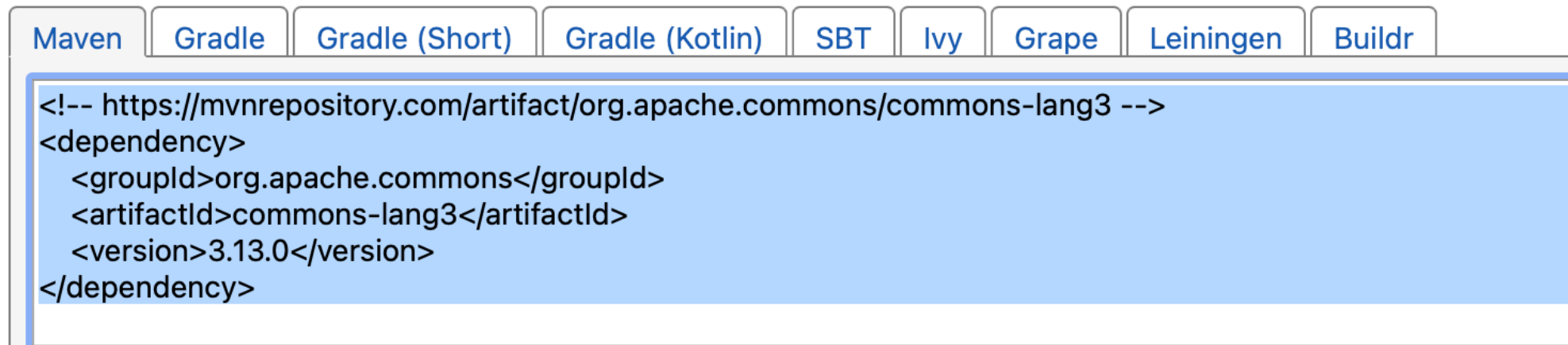
- Es el fichero que describe un proyecto Maven
- Aquí definimos
 - Propiedades
 - Dependencias
 - Plugins

```
<packaging>jar</packaging>
<properties>
    <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
    <maven.compiler.source>17</maven.compiler.source>
    <maven.compiler.target>17</maven.compiler.target>
    <exec.mainClass>es.iesaugustogonzalez.appmaven.Appmaven</exec.main
</properties>
```

Fichero pom.xml

- Dependencias

- Página con muchas dependencias: <https://mvnrepository.com>
- Buscar dependencias de Apache Commons Lang



The screenshot shows a web interface for generating Maven dependencies. At the top, there are tabs for different build systems: Maven, Gradle, Gradle (Short), Gradle (Kotlin), SBT, Ivy, Grape, Leiningen, and Buildr. The 'Maven' tab is selected. Below the tabs, a text area displays the XML code for a dependency on Apache Commons Lang 3.13.0. The code is as follows:

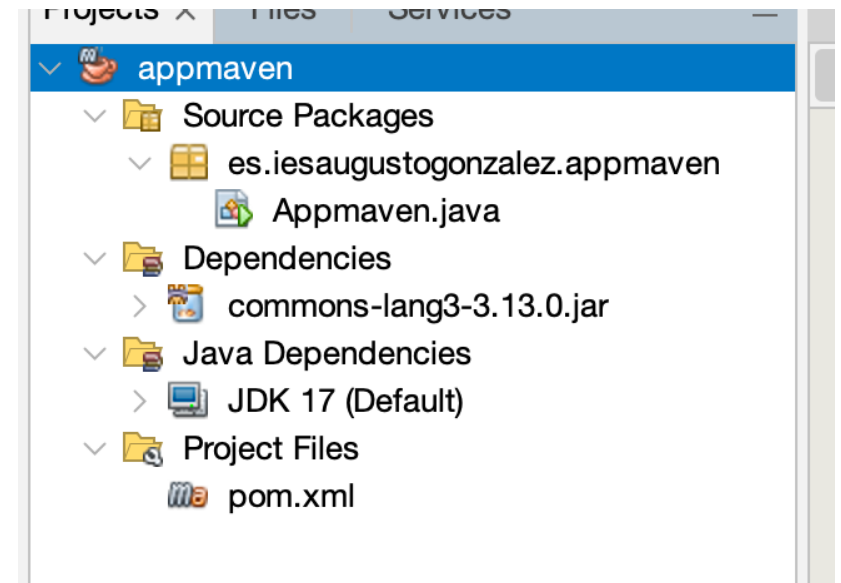
```
<!-- https://mvnrepository.com/artifact/org.apache.commons/commons-lang3 -->
<dependency>
  <groupId>org.apache.commons</groupId>
  <artifactId>commons-lang3</artifactId>
  <version>3.13.0</version>
</dependency>
```

- Copiar el texto y pegarlo al final del fichero pom.xml entre la etiqueta `<dependencies>....</dependencies>`

Fichero pom.xml

- Agregamos y hacemos ***clean and build*** del proyecto y se descarga la dependencia automáticamente

```
<dependencies>
  <!-- https://mvnrepository.com/artifact/org.ap
  <dependency>
    <groupId>org.apache.commons</groupId>
    <artifactId>commons-lang3</artifactId>
    <version>3.13.0</version>
  </dependency>
</dependencies>
```



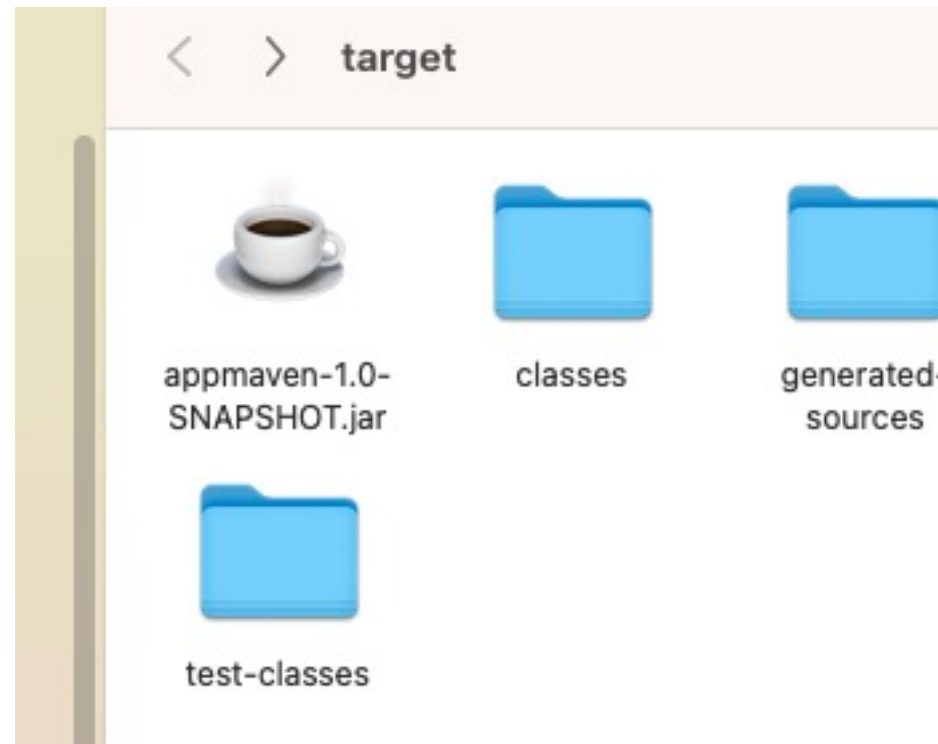
Generar un Jar con Maven

- Tenemos varias opciones. La primera que podemos utilizar es utilizando el plugin maven-jar-plugin y añadiendo esto al final del pom.xml:

```
<build>
  <plugins>
    <plugin>
      <groupId>org.apache.maven.plugins</groupId>
      <artifactId>maven-jar-plugin</artifactId>
      <version>3.3.0</version>
      <configuration>
        <archive>
          <manifest>
            <addClasspath>true</addClasspath>
            <mainClass>es.iesaugustogonzalez.appmaven.Appmaven</mainClass>
          </manifest>
        </archive>
      </configuration>
    </plugin>
  </plugins>
</build>
```

Generar un Jar con Maven

- Al hacer Clean and Build el jar nos lo genera en la carpeta **target**.



Generar un Jar con Maven

- Esta forma de generarlo es correcta. Sólo hay que tener en cuenta que las dependencias no se incluyen en el jar. Por lo tanto, para distribuirlo, se necesita el .jar **más** sus dependencias.
- En el fichero Manifest, lo único que se incluye es el atributo Class-Path con la ruta hacia las dependencias necesarias.

```
jfrancoros at MacBook-Pro-de-Joaquin in ~/Documents/2023-2024/PSP/EvaluaciónInicial/GenerarJar/appmaven/target
$ unzip -l appmaven-1.0-SNAPSHOT.jar
Archive:  appmaven-1.0-SNAPSHOT.jar
  Length   Date    Time    Name
  -----  -
      0  09-14-2023 12:58  META-INF/
    172  09-14-2023 12:58  META-INF/MANIFEST.MF
      0  09-14-2023 12:58  es/
      0  09-14-2023 12:58  es/iesaugustogonzalez/
      0  09-14-2023 12:58  es/iesaugustogonzalez/appmaven/
      0  09-14-2023 12:58  META-INF/maven/
      0  09-14-2023 12:58  META-INF/maven/es.iesaugustogonzalez/
      0  09-14-2023 12:58  META-INF/maven/es.iesaugustogonzalez/appmaven/
    723  09-14-2023 12:58  es/iesaugustogonzalez/appmaven/Appmaven.class
   1686  09-14-2023 12:55  META-INF/maven/es.iesaugustogonzalez/appmaven/pom.xml
      71  09-14-2023 12:58  META-INF/maven/es.iesaugustogonzalez/appmaven/pom.properties
  -----
    2652
                        11 files

jfrancoros at MacBook-Pro-de-Joaquin in ~/Documents/2023-2024/PSP/EvaluaciónInicial/GenerarJar/appmaven/target
$ unzip -p appmaven-1.0-SNAPSHOT.jar META-INF/MANIFEST.MF
Manifest-Version: 1.0
Created-By: Maven JAR Plugin 3.3.0
Build-Jdk-Spec: 17
Class-Path: commons-lang3-3.13.0.jar
Main-Class: es.iesaugustogonzalez.appmaven.Appmaven
```

Generar un UBER JAR con MAVEN

- Es un JAR que además de tener la aplicación tiene todas sus dependencias.
 - Por tanto, se puede distribuir como un todo-en-uno.
 - Hay varios tipos de UBER JAR. Uno de los más utilizados es el tipo *Shaded (con renombrado)*
 - Hay que utilizar el ***maven-shade-plugin*** en este caso.
 - Se descomprimen todos los jar y se unen en uno.
 - Se renombran los paquetes de todas las clases para evitar colisiones.
 - Así se evitan conflictos de versiones.

Generar un UBER JAR con MAVEN

```
<build>
  <plugins>
    <plugin>
      <groupId>org.apache.maven.plugins</groupId>
      <artifactId>maven-shade-plugin</artifactId>
      <version>3.3.0</version>
      <executions>
        <execution>
          <phase>package</phase>
          <goals>
            <goal>shade</goal>
          </goals>
          <configuration>
            <minimizeJar>true</minimizeJar>
            <filters>
              <filter>
                <artifact>*:*</artifact>
                <excludes>
                  <exclude>META-INF/license/**</exclude>
                  <exclude>META-INF/**</exclude>
                  <exclude>META-INF/maven/**</exclude>
                  <exclude>LICENSE</exclude>
                  <exclude>NOTICE</exclude>
                  <exclude>/*.txt</exclude>
                  <exclude>build.properties</exclude>
                </excludes>
              </filter>
            </filters>
            <transformers>
              <transformer implementation="org.apache.maven.plugins.shade.resource.ManifestResourceTransformer">
                <mainClass>es.iesaugustogonzalez.appmaven.Appmaven</mainClass>
              </transformer>
            </transformers>
          </configuration>
        </execution>
      </executions>
    </plugin>
  </plugins>
</build>
```

Generar un UBER JAR con MAVEN

```
jfrancor05 at MacBook-Pro-de-Joaquín in ~/Documents/2023-2024/PSP/Evaluación
$ unzip -l appmaven-1.0-SNAPSHOT.jar
Archive:  appmaven-1.0-SNAPSHOT.jar
  Length      Date    Time    Name
-----
134      09-14-2023  13:29    META-INF/MANIFEST.MF
0        09-14-2023  13:29    es/
0        09-14-2023  13:29    es/iesaugustogonzalez/
0        09-14-2023  13:29    es/iesaugustogonzalez/appmaven/
723      09-14-2023  13:29    es/iesaugustogonzalez/appmaven/Appmaven.class
0        05-20-2023  13:25    org/
0        05-20-2023  13:25    org/apache/
0        05-20-2023  13:25    org/apache/commons/
0        05-20-2023  13:25    org/apache/commons/lang3/
1669     05-20-2023  13:25    org/apache/commons/lang3/ArraySorter.class
75399    05-20-2023  13:25    org/apache/commons/lang3/ArrayUtils.class
9559     05-20-2023  13:25    org/apache/commons/lang3/BooleanUtils.class
4335     05-20-2023  13:25    org/apache/commons/lang3/CharSequenceUtils.cl
5099     05-20-2023  13:25    org/apache/commons/lang3/CharUtils.class
986      05-20-2023  13:25    org/apache/commons/lang3/Charsets.class
```

GENERAR Y EJECUTAR UN
.CLASS DESDE LÍNEA DE
COMANDOS

Creamos con Netbeans o cualquier editor de texto un fichero .java

```
package app;

/**
 *
 * @author jfrancoros
 */
public class App {
    public static void main(String[] args) {
        System.out.println(x: "Hola mundo");
    }
}
```

Compilar un fichero .java para generar un .class desde línea de comandos

- Nos colocamos en la carpeta donde está el fichero .java que hemos creado con Netbeans y ejecutamos el comando:
 - **javac** *fichero.java*

```
jfrancoros at MacBook-Pro-de-Joaquin in ~/Documents/2023-2024/PSP/EvaluaciónInicial/GenerarJar/app/src/app
$ ls
App.java

jfrancoros at MacBook-Pro-de-Joaquin in ~/Documents/2023-2024/PSP/EvaluaciónInicial/GenerarJar/app/src/app
$ java -version
java version "17.0.8" 2023-07-18 LTS
Java(TM) SE Runtime Environment (build 17.0.8+9-LTS-211)
Java HotSpot(TM) 64-Bit Server VM (build 17.0.8+9-LTS-211, mixed mode, sharing)

jfrancoros at MacBook-Pro-de-Joaquin in ~/Documents/2023-2024/PSP/EvaluaciónInicial/GenerarJar/app/src/app
$ javac App.java

jfrancoros at MacBook-Pro-de-Joaquin in ~/Documents/2023-2024/PSP/EvaluaciónInicial/GenerarJar/app/src/app
$ ls
App.class App.java
```

Ejecutar un fichero .class desde línea de comandos

- Se hace con el comando **java**
 - **java** fichero (sin el .class).
 - En nuestro caso, el comando sería `java App`. Pero nos sale el siguiente error.

```
$ java App
Error: Could not find or load main class App
Caused by: java.lang.NoClassDefFoundError: app/App (wrong name: App)
```

- El problema es que hemos creado la clase dentro del paquete `app`.

Ejecutar un fichero .class desde línea de comandos

- En este caso hay que colocarse en el directorio anterior y hacer el comando:
 - java **nombrepaquete.fichero** (sin el .class).

```
jfrancoros at MacBook-Pro-de-Joaquin in ~/Documents/2023-2024/PSP/EvaluaciónInicial/GenerarJar/app/src
$ ls
app

jfrancoros at MacBook-Pro-de-Joaquin in ~/Documents/2023-2024/PSP/EvaluaciónInicial/GenerarJar/app/src
$ java app.App
Hola mundo
```

GENERAR UN .JAR DESDE
LÍNEA DE COMANDOS

Generar un .jar desde línea de comandos

- La herramienta **jar** nos permite generar ficheros de este tipo
 - Algunas opciones
 - v: verbose, más información de salida
 - c: create, para crear el fichero .jar
 - f: file, para redirigir la salida a un fichero
 - Ejemplos
 - `jar cvf app.jar App.class Person.class Game.class`
 - `jar cvf app.jar -C RutaRaizPaquetesYClases .` (El punto final es necesario)
- Para más información **jar --help**

Generar un .jar desde línea de comandos

```
jfrancoros at MacBook-Pro-de-Joaquin in ~/Documents/2023-2024/PSP/EvaluaciónInicial/GenerarJar/app/build
$ jar cvf app.jar -C classes .
added manifest
adding: .DS_Store(in = 6148) (out= 382)(deflated 93%)
adding: app/(in = 0) (out= 0)(stored 0%)
adding: app/App.class(in = 519) (out= 332)(deflated 36%)
```

- Al ejecutarlo nos da un **error** muy típico de fichero de Manifest.

```
jfrancoros at MacBook-Pro-de-Joaquin in ~/Docum
p
$ java -jar app.jar
no main manifest attribute, in app.jar
```

Generar un .jar desde línea de comandos

- El error da porque el manifiesto creado por defecto no tiene la línea Main-class indicando cuál es la clase principal. Como se puede ver, sólo se muestran los atributos Manifest-Version y Created-By.

```
$ unzip -p app.jar META-INF/MANIFEST.MF  
Manifest-Version: 1.0  
Created-By: 17.0.8 (Oracle Corporation)
```

Modificar el fichero de manifiesto de un .jar

- Podemos añadir entradas al fichero de manifiesto
 - Creamos un fichero *Manifest.txt*
 - Añadimos los pares *key* y *value* que necesitamos
 - Main-Class: app.App
 - Añadimos un salto de línea al final del fichero!!
 - Como la clase está dentro del paquete app se pone **app.App**.
 - Ejecutamos la opción *m* y la ruta hacia el fichero
 - **Muy importante** que la *m* y *f* aparezcan en el mismo orden que el fichero de Manifest y el fichero .jar en el comando

```
jfrancoros at MacBook-Pro-de-Joaquin in ~/Documents/2023-2024/PSP/EvaluaciónInicial/GenerarJar/app/build
$ jar cvmf Manifest.txt app.jar -C classes .
added manifest
adding: .DS_Store(in = 6148) (out= 382)(deflated 93%)
adding: app/(in = 0) (out= 0)(stored 0%)
adding: app/App.class(in = 519) (out= 332)(deflated 36%)
```