Tema 1B – Uso y creación de librerías

Unai Aguilera Irazabal

unai.aguilera@deusto.es

Grado en Ingeniería Informática

Facultad de Ingeniería – Universidad de Deusto

Reflexiona

- ¿Cómo organizas tu programa Java?
- ¿Se te ocurre alguna manera de reutilizar código y funcionalidad entre distintos programas?
- ¿Cómo distribuirías tu programa Java una vez que lo has terminado?
  - ¿Y sus dependencias?

Librería en Java

- Además de la funcionalidad del API de Java podemos incluir librerías en nuestros programas.
  - En Java las librerías externas se suelen encontrar empaquetadas en ficheros *.jar* (son ficheros *.zip* que contienen los *.class* resultado de la compilación, así como otros recursos necesarios).
- Un concepto importante en Java es el *classpath*. Es una propiedad que indica al compilador y a la JVM dónde debe buscar las clases necesarias para ejecutar un programa.
  - Por defecto incluye la librería de Java (API) y el directorio actual.
- Para utilizar una librería externa en Java debemos:
- Indicarle al compilador que la use al compilar (si existe más de un fichero *.jar* las rutas se separan con ; en Windows y con : en Linux).

```
javac -cp dir1;dir2;dir3;... <fichero(s) a compilar>
```

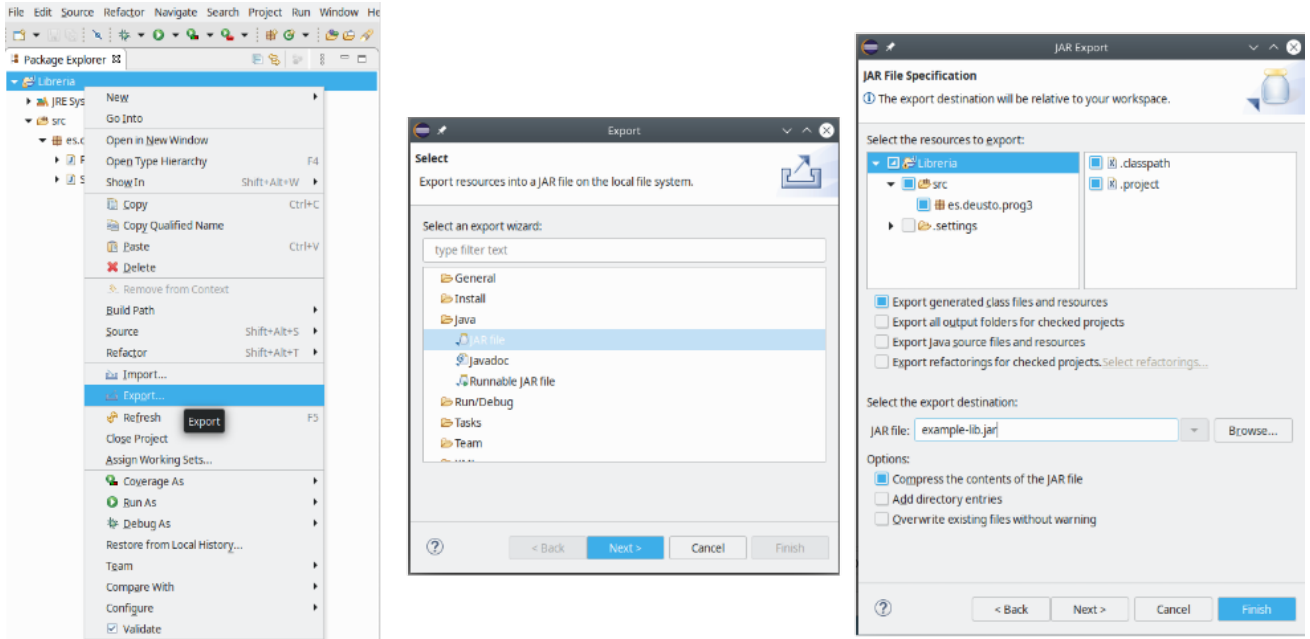
- Cuando se lanza el programa debemos hacer lo mismo con la JVM.

```
java -cp dir1;dir2;dir3;... <clase principal del programa>
```

- Se debe incluir también el directorio actual en este caso porque el argumento *-cp* sobrescribe el CLASSPATH por defecto y se dejaría de tener en cuenta el directorio actual al buscar clases.

Creando nuestras propias librerías

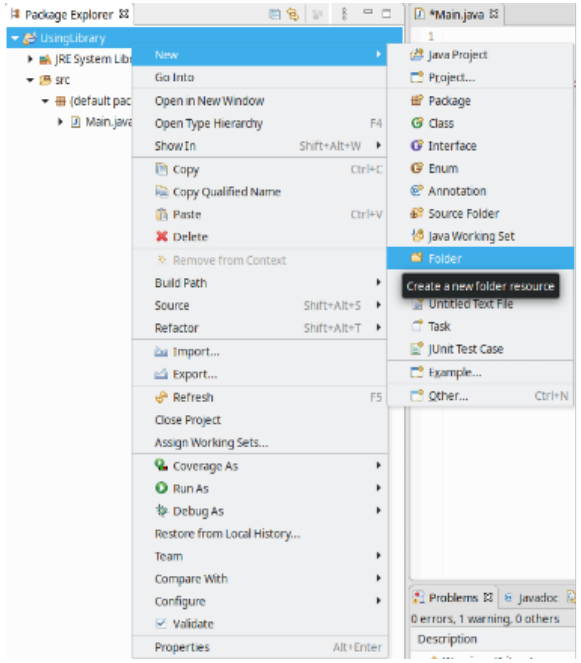
- Si utilizamos Eclipse podemos exportar un proyecto como .jar directamente seleccionando la opción disponible en un proyecto Java.

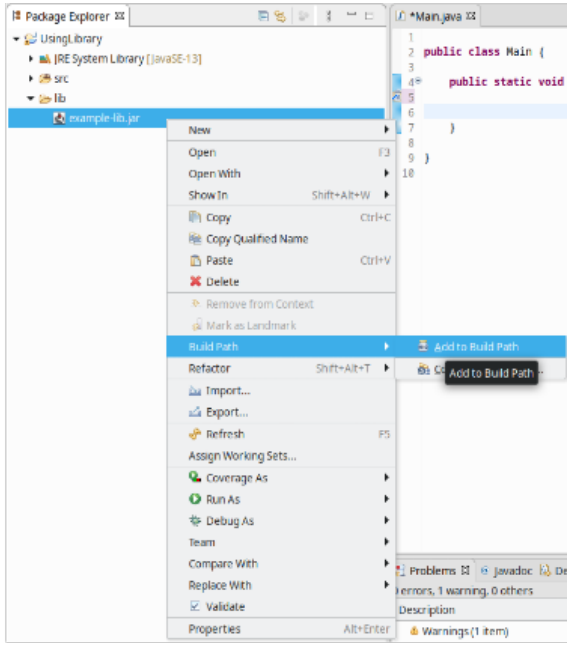


- En el directorio *external_library* se explica el procedimiento para crear una librería en la línea de comandos.

Usando librerías externas

- Si queremos incluir una librería externa para usarla en nuestro programa (pueden ser uno o varios ficheros .jar).
  - Creamos una carpeta *lib* en el proyecto y copiamos ahí los .jar que constituyen la librería externa.
  - Añadimos todos los .jar necesarios al Build path del proyecto. Esto hace que Eclipse use estas librerías al compilar y lanzar el programa (deben aparecen luego en Referenced Libraries).





Práctica 1B – Parte I

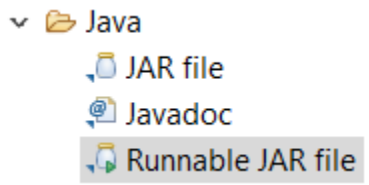
- Crea y configura un proyecto en Eclipse que contenga el código fuente correspondiente a la Práctica 1B en ALUD.
  - Debes respetar la estructura de paquetes y subpaquetes de dicho código.
- Usa la funcionalidad de Eclipse para exportar un JAR como resultado de la compilación del proyecto.
- Recuerda que, en este caso, ese .JAR no es un programa completo, sino que contiene funcionalidad que puede ser utilizada en otros programas.
  - Cuando en Java decimos que hacemos uso de una librería estamos haciendo uso de clases (*.class*) que han sido compilados por otras personas pero que no son programas completos, aunque están listos para ser utilizados.
- Puedes abrir el JAR generado como un compresor (ZIP) para ver su contenido.

Práctica 1B – Parte II

- Ahora que ya has creado un proyecto para desarrollar y generar tu propia librería debes crear otro proyecto con el programa principal, que haga uso de dicha librería generada.
- Para ello, crea un nuevo proyecto que contenga como programa principal la clase *EjemploLibrary* contenida en el repositorio de código en el directorio *tema1*
- Lanza el programa tal cual y comprueba el error que se produce debido a que falta la librería.
- Añade al proyecto correctamente la librería que has exportando anteriormente para que sea utilizada en este proyecto.
- Comprueba que ahora el programa se ejecuta correctamente ya que dispone de todas las partes necesarias.
- Comprueba el contenido del fichero JAR generado abriéndolo como un .zip
  - Verás que contiene los .class resultado de la compilación y una carpeta META-INF con un fichero MANIFEST.MF que contendrá distinta información según el tipo de fichero JAR y su configuración.

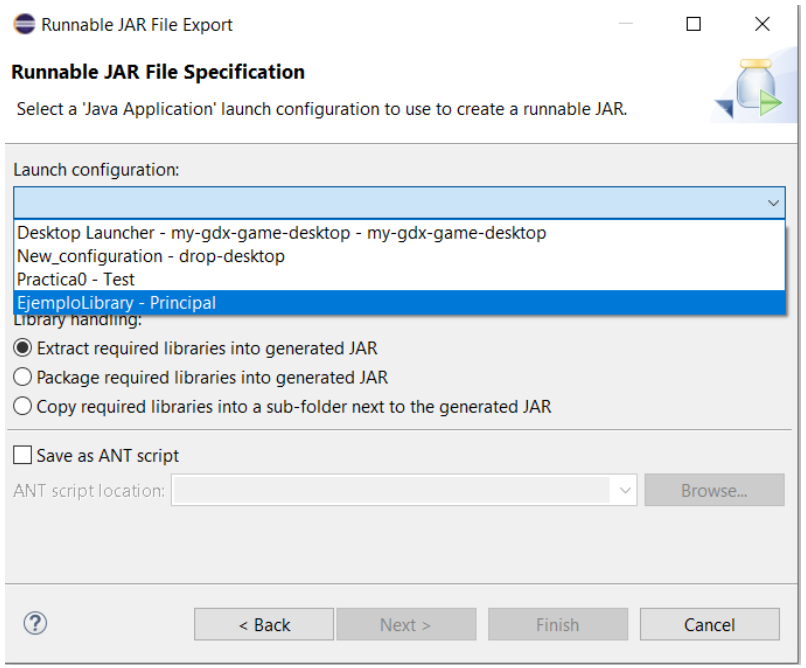
Generando JAR ejecutable (I)

- Los programas completos Java también pueden ser distribuidos como ficheros JAR que contienen todas las clases y recursos necesarios para su ejecución.
  - Podemos aplicar el procedimiento anterior para crear un JAR a partir de cualquier proyecto en Eclipse.
- Como el proyecto que estamos exportando tiene un programa principal, tenemos la opción de hacer que el JAR sea ejecutable (*Runnable*)
  - Para ello seleccionaremos la opción "*Runnable JAR file*" al exportar el proyecto.



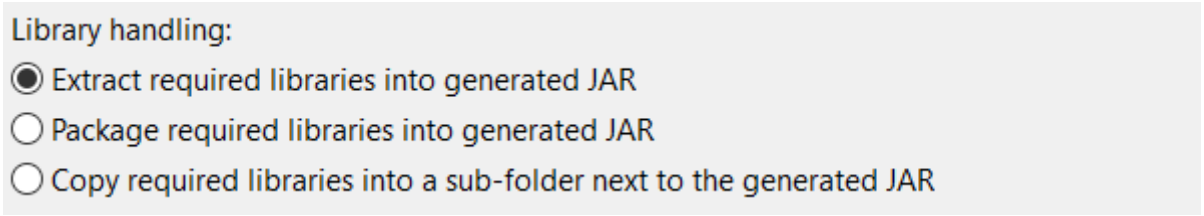
Generando JAR ejecutable (II)

- Indicaremos qué configuración de lanzamiento de las disponibles en nuestro se usará para lanzar el *main*
  - Esto configura en el JAR qué clase contiene el *main* que inicia el programa.



Generando JAR ejecutable (III)

- Además, podemos indicar que incluya las librerías necesarias dentro del JAR
  - Esto evita que tengamos que distribuir varios ficheros distintos y que sea más fácil lanzar el proyecto.
  - Puede ocasionar problemas de licencias de distribución o conflictos de versiones.



Práctica 1B – Parte III

- Continuando con la práctica anterior, exporta el proyecto que contiene al programa principal en un JAR ejecutable.
  - Selecciona correctamente qué configuración de Eclipse lanza el proyecto.
  - Indica que la librería se incluya dentro del JAR generado.
- El JAR resultante se puede lanzar haciendo
  - **Directamente haciendo doble-clic.** Esto no siempre funciona ya que depende de la configuración de Windows para abrir ficheros JAR.
    - Además, si el programa es en línea de comandos, la salida puede no verse.
  - **En línea de comandos.** Busca el fichero JAR resultado de la exportación (suponemos que se llama *ejemplo.jar*) y ejecuta `java -jar ejemplo.jar`
- Abre el JAR resultante como un ZIP para ver su estructura interna.
  - Verás que contiene las clases, tanto principal como las de la librería que se han extraído dentro y la configuración de cuál es la clase principal y el CLASSPATH a usar en el MANIFEST.MF

```
Manifest-Version: 1.0
Main-Class: EjemploLibrary
Class-Path: .
```

Práctica 1B – Parte IV

- Ahora vamos a comprobar a generar un JAR ejecutable que no incluya directamente las librerías necesarias.
- Genera el JAR de nuevo utilizando la opción correspondiente para no incluir las librerías, sino que estas se copien junto con el fichero exportado.
  - Lanza de nuevo el JAR usando `java -jar ejemplo.jar`
  - El programa funciona porque el JAR principal ha sido configurado en su MANIFEST.MF para indicar el CLASSPATH por defecto.

```
Manifest-Version: 1.0
Main-Class: EjemploLibrary
Class-Path: . ejemplo_lib/example-lib.jar
```

  - En este caso, el CLASSPATH carga el directorio actual y el directorio *ejemplo_lib*, generado automáticamente al exportar las librerías del proyecto.