# Introducción a la herramienta de desarrollo Eclipse

## *Introducción*

En este laboratorio se va a presentar la herramienta de desarrollo de software Eclipse. Se trata de una plataforma de software de código abierto independiente de plataforma que permite desarrollar aplicaciones en diferentes lenguajes de programación con el consiguiente ahorro de tiempo y esfuerzo.

## *Objetivos*

Los objetivos de este laboratorio son:

- Tener una primera toma de contacto con un entorno de desarrollo visual Java, que en concreto, es el que se va a utilizar durante los laboratorios y la práctica de la asignatura.
- Mostrar las ventajas del uso de estas herramientas: generación automática de código y facilidades de compilación, ejecución, depuración de programas.
- Mostrar las facilidades para consulta on- line de documentación sobre Java.
- Mostrar las posibilidades de generación automática de documentación para nuestra aplicación, extraída de los comentarios escritos en los programas fuente.
- Obtener un programa ¿ejecutable? para poder lanzar la aplicación desde fuera de la herramienta de desarrollo.

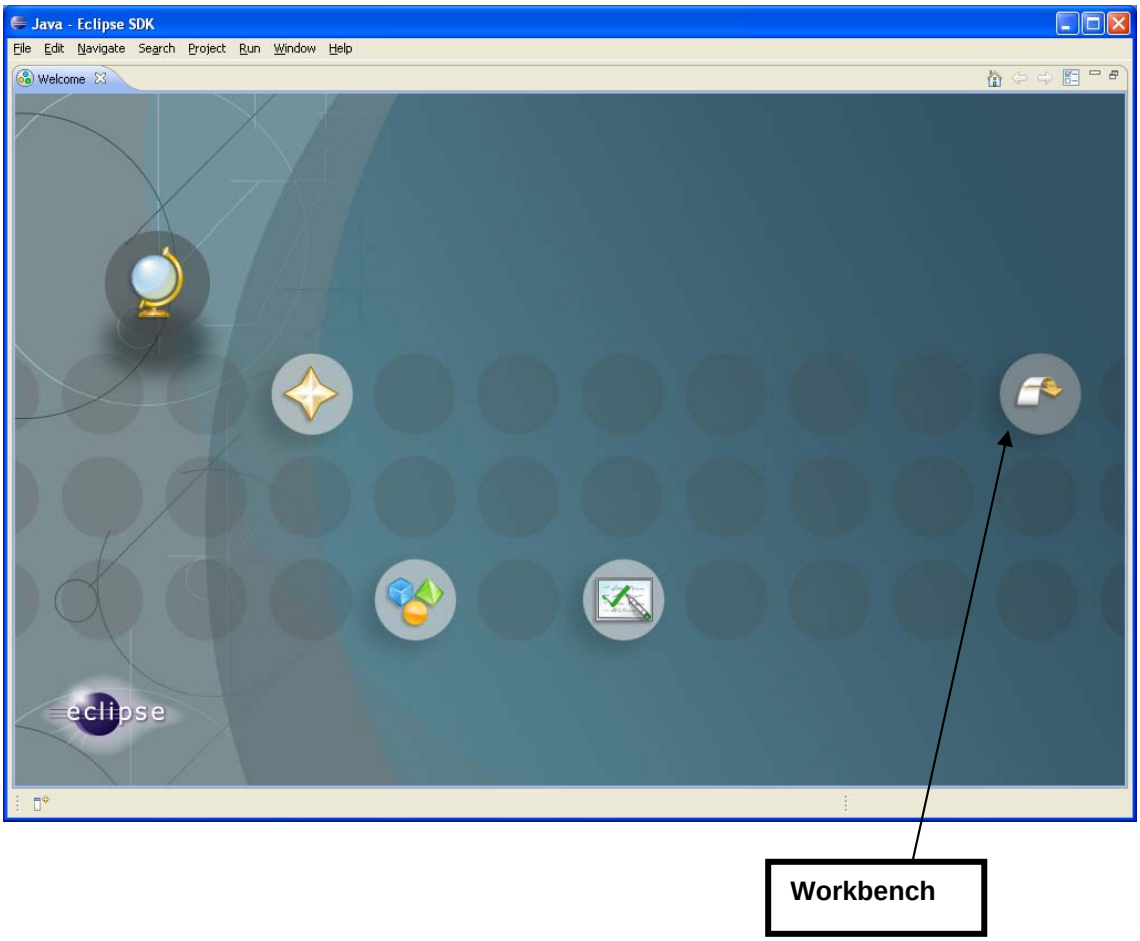
Para ello realizaremos una implementación de un ejercicio clásico como es el de calcular el número Fibonacci de un número. A continuación se crear xxxx. Por último, veremos cómo crear documentación usando Javadoc y cómo crear un fichero .jar.

## *Procedimiento a seguir*

A continuación se describen los pasos a realizar partiendo de la situación en la que se encuentra todo el software necesario instalado en la máquina. En el **anexo 1** se explica cómo instalar Eclipse.

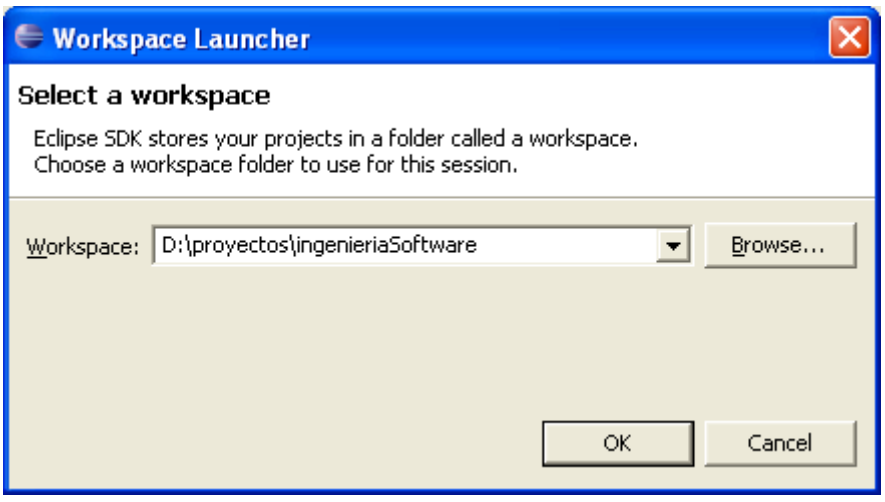
### Abrir la herramienta de trabajo Eclipse

A continuación se puede ver al pantalla de inicio de Eclipse.




**Crear un nuevo espacio de trabajo** (workspace). Desde la pantalla anterior pincharemos sobre el icono correspondiente a Workbench.

Nos aparece una ventana para seleccionar el espacio de trabajo. En la ventana que aparece, elegiremos uno de los Workspace que ya estén creados o escribiremos el nombre de uno nuevo. En nuestro caso vamos a crear un nuevo espacio de trabajo por lo que escribiremos la ubicación en la que se va a guardar:

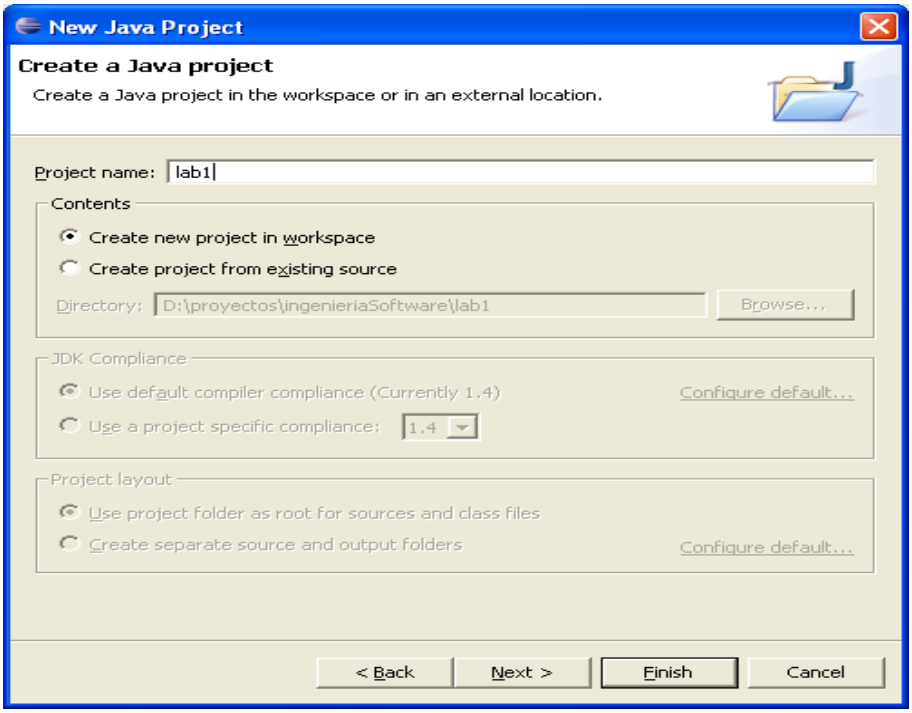


Pinchamos en OK. En ese momento se volverá a abrir Eclipse y tendremos nuestro espacio de trabajo vacío. En el espacio de trabajo crearemos los diferentes proyectos que vayamos a realizar. Podremos tener diferentes proyectos en el mismo espacio de trabajo creado.

**Crear proyecto**

Pincharemos en File / New / Project y seleccionaremos 'Java Project' en la ventana que se abre.

Escribimos el nombre del proyecto y seleccionamos la opción 'Create new project in workspace'.



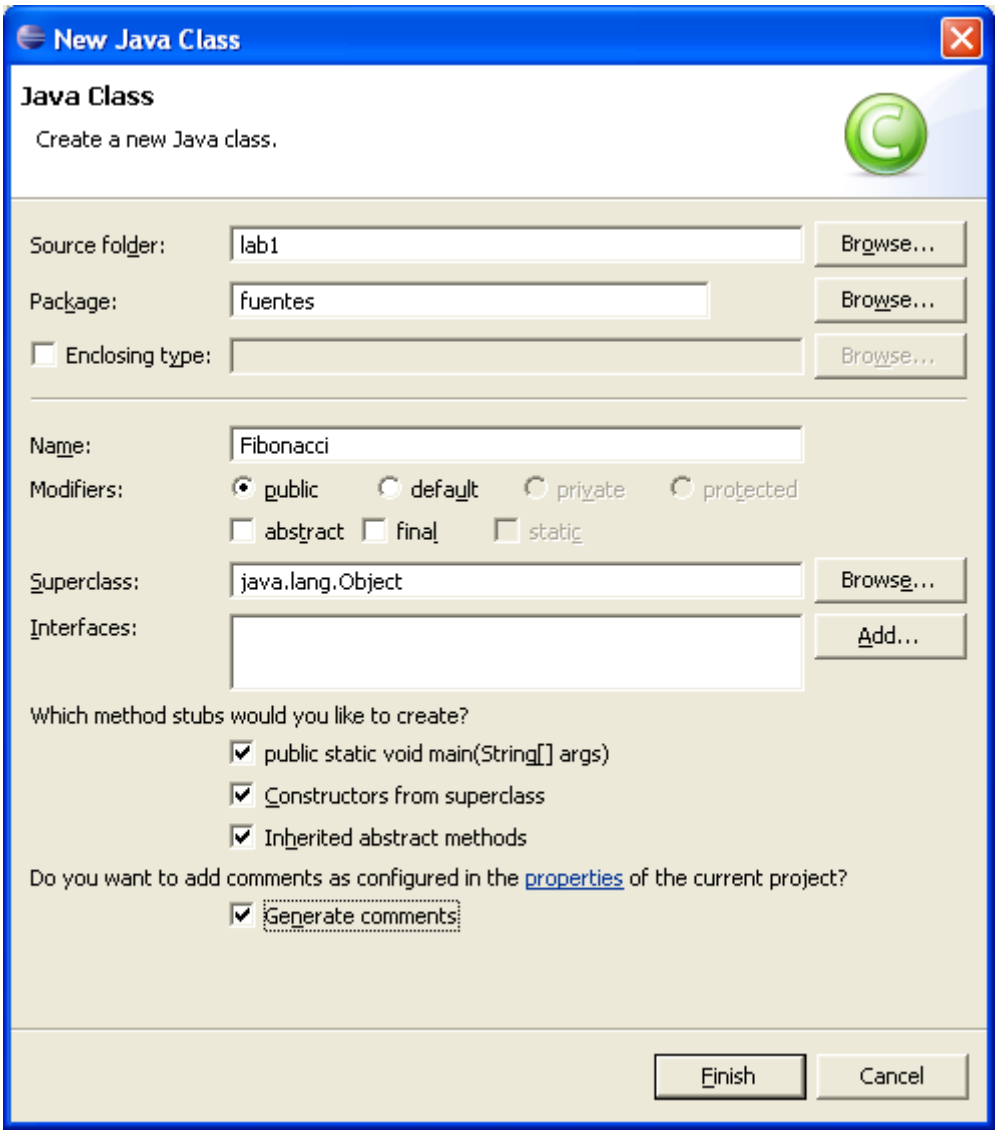

Después pulsamos el botón 'finish'.

Vemos que aparece nuestro proyecto creado, con la subcarpeta src para las fuentes. Podemos observar que en sistema de ficheros se ha creado una carpeta con el nombre del proyecto dentro del workspace. Dentro de esta carpeta aparece una subcarpeta que se llama src donde se ubicarán los ficheros fuente.

Ya tenemos el proyecto creado.

**Crear clase java**

Los ficheros java los vamos a guardar dentro de la carpeta 'src'. Para crear nuestra primera clase java pincharemos en File / new / Class y rellenamos los datos como se muestran a continuación:




Pulsamos 'Finish' y veremos, en el explorador de la izquierda que nos ha creado un nuevo fichero debajo del paquete 'fuentes', como le hemos especificado.

### Introducir el código

Copiamos el siguiente código en la clase generada:

```
public static void main(String[] args) {
    int numero = 4;
    System.out.println("El número de Fibonacci es: "+ calcFibonacci(numero));

}

public static int calcFibonacci(int n){
    int ant = 0;
    int sig = 1;
    for (int j=0; j <n; j++){
        int tmp = sig;
        sig = sig + ant;
        ant = tmp;
    }
    return ant;
}
```

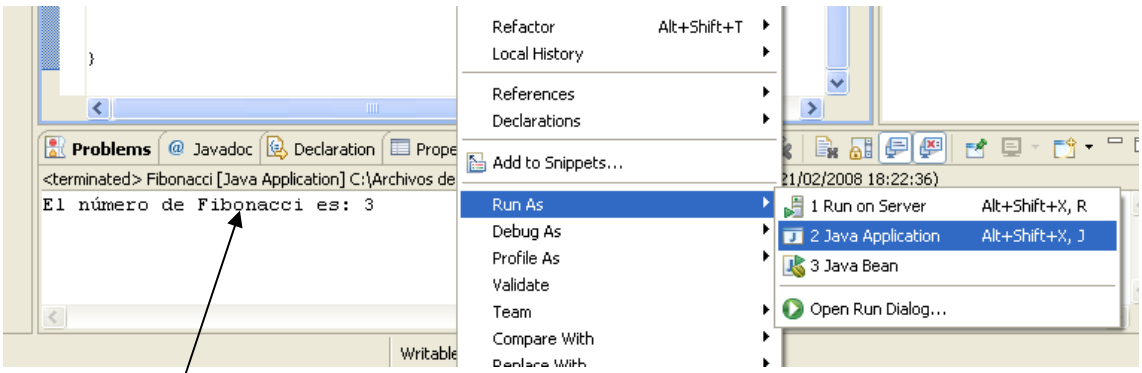
### Compilar la aplicación

Para compilar la clase es suficiente con hacer `ctrl. + s` (para salvar el código modificado) y `ctrl. + b`. Si estuviera seleccionada la opción "`/Project/Build automaticaly`" sería suficiente con pulsar `ctrl. + s` ya que cada vez que detectara que se ha modificado una clase la compilaría de manera automática.

Podremos ver en la carpeta 'classes' (carpeta donde le hemos dicho que deje las clases compiladas) que ha generado una estructura de carpetas que coincide con la estructura del paquete en el que se encuentra la clase java y dentro está el fichero compilado con extensión .class

### Ejecutar la aplicación

Utilizaremos la opción '`Run`' que encontraremos pulsando con el botón derecho del ratón sobre la clase java, dentro de la opción "`Run as/Java aplication`". Otra posibilidad es pulsar sobre el icono [ ] que aparece en la barra de herramientas de la parte superior.




Resultado de la ejecución

**Ejecutar la aplicación fuera de Eclipse, invocando directamente a la máquina virtual Java.**

Los pasos a realizar son los siguientes:

1) Lanzar un intérprete de comandos (Inicio => Ejecutar => cmd)

2) Asegurarse de que la herramienta JAVA se puede ejecutar en dicho intérprete de comandos. Escribir el comando "`java –version`" y ver si lo encuentra o no.

Si no lo encuentra hay que modificar la variable de entorno PATH del sistema. El PATH se puede cambiar en el PANEL DE CONTROL => Sistema => Seleccionar solapa Opciones Avanzadas => Variables de Entorno. Hay que añadir o editar la variable de entorno de usuario PATH para que contenga el directorio donde se encuentra java.exe.

3) Posicionarse en el directorio del CLASSPATH (en nuestro caso en misclases) (con el comando cd del sistema operativo) donde se encuentran todos los ficheros .class, y ejecutar la clase laboratorio1.Fibonacci

```
java laboratorio1.Fibonacci
```

**NOTAS**:

Si ejecutamos la versión de Fibonacci que recibe la entrada como parámetro lo haremos de esta forma: java laboratorio1.Fibonacci 5 (por ejemplo)

Es importante que estemos situados justo en la carpeta que contiene el nombre de la fuente que contiene la clase.

**Consultar la documentación de las API de Java.**

Comprobar que efectivamente el método "parseInt" de la clase "Integer" puede lanzar una excepción "NumberFormatException". Para consultar la documentación de las API de Java, se hará directamente en Internet, que a veces es hasta más rápido. Para ello buscar en Google directamente (poniendo "API" y el nombre de la clase, por ejemplo "API Integer"). En el paquete "java.lang", encontraremos la clase "Integer" y dentro de la misma podremos ver que existe un método "parseInt" que puede lanzar una excepción "NumberFormatException". La documentación completa de Java puede ser accedida también a través de Google (buscando API Java) o accediendo a http://docs.oracle.com/javase/7/docs/api/, para acceder a la API de JDK Standard Edition 7.

**Generar documentación de manera automática.** Para ello pulsaremos en la opción del menú "`Project=>Generate Javadoc`". La documentación se generará dependiendo de las opciones seleccionadas en las propiedades del proyecto: "`Project => Properties => Java Compiler =>Javadoc`". El destino de la

documentación generada se lo especificaremos en "`Project => Properties =>Javadoc Location`". Podríamos elegir la carpeta misdocs creada inicialmente. Cuando termine, podremos comprobar que en el subdirectorio *misdocs* aparecen varios ficheros .html con la documentación de las clases que hemos creado. Puede verse que se encuentra en el mismo formato que la documentación previamente consultada de las librerías de Java, esto es, el formato de la documentación de Java JDK. Aunque mucha de esta documentación se ha generado de manera automática, es conveniente añadir comentarios Javadoc a las clases fuente de Java para que sean incluidos en la documentación. Por ejemplo, comprobar lo que sucede si se añade el siguiente comentario Javadoc inmediatamente antes del método public static int calcFibonacci(int n) en la clase Fibonacci:

```
/**
Método para calcular el número fibonacci de un número (menor que 47)
* @param n Número del que quiere calcularse el fibonacci
* @return Fibonacci de n
*/
```

y se genera de nuevo la documentación con Javadoc. Ver cómo quedaría la documentación.

**Generar un fichero .jar ejecutable (en realidad ejecutable por la máquina virtual Java).**

Vamos a crear un fichero jar que nos permita distribuir la aplicación de forma que nos permita ejecutarla de forma autónoma. El fichero se creará con ayuda de Eclipse.

### A. Usando el comando JAR

Para ello hay que:

1) Lanzar un intérprete de comandos (Menú Inicio del Sistema Operativo => Ejecutar => cmd)
2) Asegurarse de que la herramienta JAR se puede ejecutar en dicho intérprete de comandos. Escribir el comando jar y ver si lo encuentra o no. Si no lo encuentra hay que modificar la variable de entorno PATH del sistema. El PATH se puede cambiar en el PANEL DE CONTROL => Sistema => Seleccionar solapa 'Opciones Avanzadas' => Variables de Entorno. Ahí hay que añadir o editar la variable de entorno de usuario PATH para que contenga el directorio donde se encuentra jar.exe (carpeta bin dentro de la instalación del jdk). Una vez modificado el PATH, volver a lanzar un intérprete de comandos y comprobar que se ejecuta JAR.
3) Posicionarse en el directorio del CLASSPATH (en nuestro caso en *misclases*) (con el comando cd del sistema operativo) donde se encuentran todos los ficheros .class, crear con un editor de textos un fichero que diga cuál es la clase principal de la aplicación (por ejemplo con el comando edit). Crearemos un fichero (por ejemplo miManifiesto.mf) cuyo contenido sea:

```
Manifest-Version: 1.0
```

```
Main-Class: laboratorio1.Principal
```

**NOTA**: terminad la línea anterior con el carácter RETURN y GUARDAD UN ESPACIO EN BLANCO después de Main-Class:

Y desde el directorio (misclases), crear el fichero ejecutable (por ejemplo lab1.jar) con el siguiente comando:

```
jar cvfm lab1.jar miManifiesto.mf laboratorio1\*
```

El comando anterior crea el fichero lab1.jar que incluye de manera comprimida todos los ficheros .class que se encuentran en el paquete laboratorio1, y que contiene así mismo información de cuál será la clase principal se va a ejecutar.

Dicho fichero ejecutable se podrá ejecutar "directamente" con el siguiente comando:

```
java -jar lab1.jar
```

### B. Usando Eclipse

Antes de crear el JAR tenemos que crear un archivo “Manifest” en el que especificaremos cuál es la clase principal que queremos ejecutar (clase que contiene el método estático ‘main’).

Para crear el archivo haremos bajo el proyecto File => New => File Introduciremos el nombre Manifest.mf y presionaremos Finish.

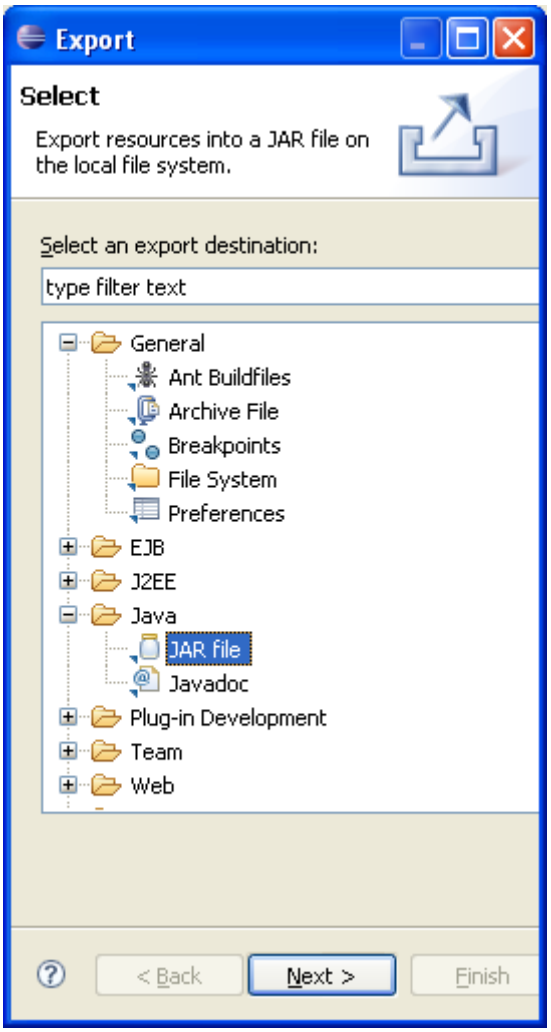
Introduciremos el siguiente contenido:

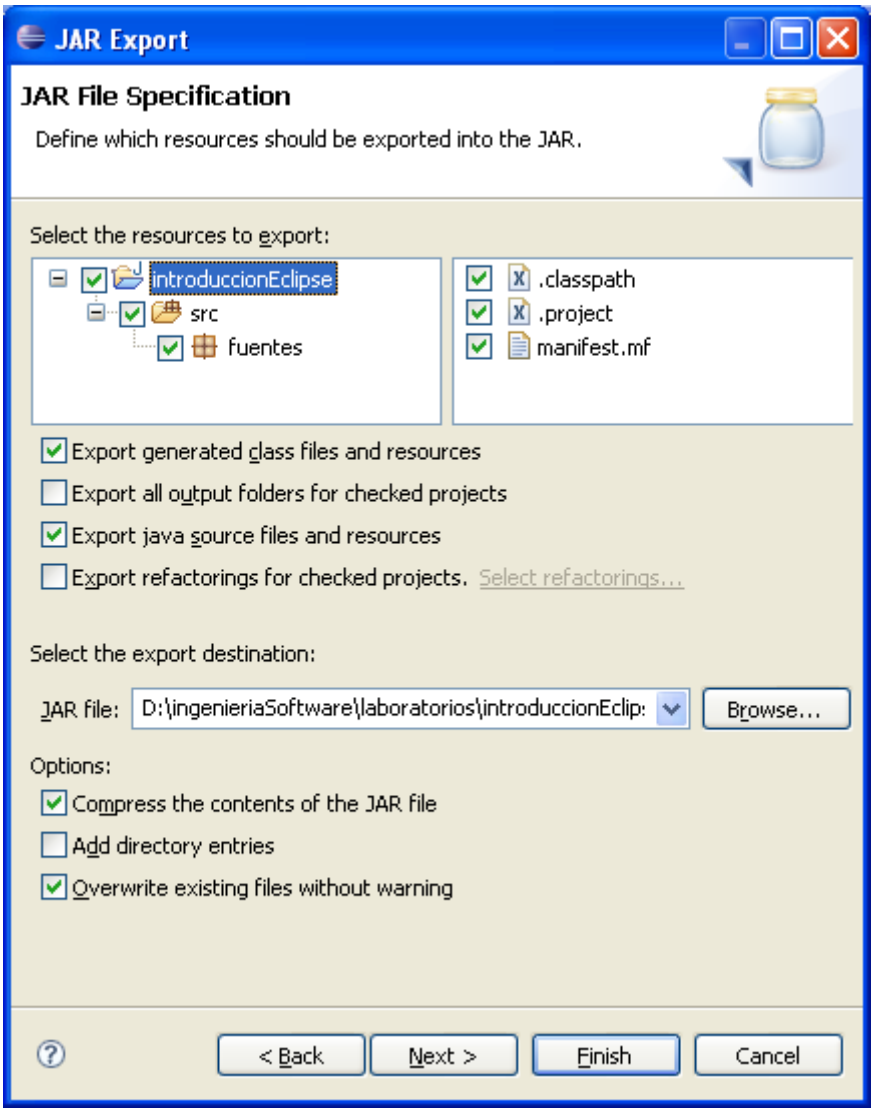
```
Manifest-Version: 1.0

Main-Class: laboratorio1.Principal
```

Para ello hay que:

1) Seleccionar File => Export
2) Elegimos ‘Jar’ en las opciones ofrecidas. A continuación se puede ver la pantalla.


Export
Select
Export resources into a JAR file on the local file system.
Select an export destination:
type filter text
General
Ant Buildfiles
Archive File
Breakpoints
File System
Preferences
EJB
J2EE
Java
JAR file
Javadoc
Plug-in Development
Team
Web
< Back
Next >
Finish




Pulsamos Next. Estaríamos en la pantalla de 'Packaging Options'. Dejamos las dos opciones que aparecen seleccionadas y volvemos a pulsar Next. En esta nueva pantalla 'JAR Manifest Specification' deberemos elegir el fichero manifest que hemos seleccionado.


Y pulsamos el botón Finish. Ya nos ha creado el fichero jar.

En JAR Options => Indicar el nombre del fichero jar (JAR File) (por ejemplo, lab1.jar), seleccionar que incluya un fichero de manifiesto (Include Manifest File) e indicar en la caja de texto de Main Class, la clase a ejecutar (en nuestro caso laboratorio1.Principal)

Una vez creado e fichero se podrá ejecutar "directamente" con el siguiente comando:

```
java -jar lab1.jar
```

**Reutilización de las clases creadas en este proyecto en otros proyectos.** Después de salvar todo el proyecto, vamos a crear un nuevo proyecto como se ha hecho anteriormente, pero indicando para el mismo otro paquete Java diferente, por ejemplo Laboratorio1Nuevo. Para que la clase Fibonacci del paquete Laboratorio1 sea accesible a las clases del nuevo proyecto hay que modificar el CLASSPATH de este segundo proyecto. En Eclipse se hace desde: "`Project / Properties`". Dentro de la opción "Java Build Path" escoged la solapa '`Add External JARs…`' y buscar el jar creado anreriormente. No hay que olvidar que, dentro de las clases que quieran usar la clase Fibonacci hay que añadir la sentencia import correspondiente: import Laboratorio1.Fibonacci;

**Salir de la herramienta una vez que acabemos el laboratorio eliminando la carpeta creada al principio, que contiene nuestros proyectos, junto con los .java generados, las clases como resultado, .class , y los proyectos, el Workspace y los ficheros con la documentación en formato html.**