

JAVA DESDE CONOSLA AVANZADO

UD2 Entornos de desarrollo

ÍNDICE

1. INTRODUCCIÓN	2
2. PLANTEAMIENTO DEL EJERCICIO	2
3. SOLUCIÓN PROPUESTA	2
3.1. IDENTIFICANDO ACTORES O ELEMENTOS INVOLUCRADOS	2
3.2. DESCRIPCIÓN DE LAS CLASES	3
3.3. USO DEL TRADUCTOR/COMPILADOR	7
4. EMPAQUETAR .CLASS EN FICHERO .JAR	8
4.1. CREAR EL CHERO MANIFEST.MF	8
4.2. CREAR EL CHERO SEMAFORO.JAR	9
ACTIVIDAD. EJECUCIÓN DESDE ECLIPSE Y DESDE LA CONSOLA	10

1. Introducción

Este documento nos servirá de apoyo para poder entender, a través de un ejemplo, algunos de los conceptos que hemos estado viendo hasta el momento, tales como:

- Lenguaje de alto nivel. Un acercamiento al lenguaje Java.
- Código fuente y código máquina.
- Traductor. En este caso se trata del compilador de Java.
- Código interpretable o bytecode. Intérprete.
- Conceptos asociados con la programación orientada a objetos.

No forma parte de este ejercicio, entender el código Java en su totalidad. Dichos contenidos serán tratados en otros módulos del ciclo.

2. Planteamiento del ejercicio

Un profesor de autoescuela pretende enseñar a un estudiante la combinación de colores entre los que puede ir cambiando un semáforo. En el aula existe un ordenador que simula el comportamiento del semáforo, al que podemos consultar el color que tiene en cada momento.

En la secuencia de ejecución, el profesor preguntará al estudiante el color que tiene el semáforo. Para responder, el estudiante obtiene el resultado del ordenador.

3. Solución propuesta

Cuando damos solución a un problema propuesto mediante un programa, éste puede ser implementado de muy diversas formas. Cualquiera de ellas será igualmente válida siempre y cuando cumpla los requisitos que se le piden (funcionales, de rendimiento, disponibilidad ...).

A continuación se propone una de las posibles soluciones.

.

3.1. Identificando actores o elementos involucrados

De acuerdo a lo solicitado en el enunciado, es posible plantear el problema haciendo uso de tres actores (profesor, estudiante, ordenador). Para cada uno de ellos se desarrollará una clase Java.

Se crearán cinco ficheros de código fuente, que una vez compilados nos proporcionarán otros tantos ficheros en código intermedio (bytecode de Java).

Para la ejecución del programa, el intérprete de Java hará uso de los ficheros bytecode creados, junto a otras funciones pertenecientes a las bibliotecas de Java.

Cuando queremos implementar una solución, se crea un “proyecto”, cuyos contenidos quedan almacenados en el sistema de archivos del ordenador en un directorio (semaforo en nuestro caso).

Tendremos que crear los directorios semáforo y dentro de él los directorios clases y principal con el comando mkdir:

```
mkdir semaforo → crea el directorio
```

```
cd semaforo → entrar al directorio
```

```
mkdir principal → crea el directorio
```

```
mkdir clases → crea el directorio
```

Algunas órdenes que os ayudarán en la creación de los directorios son:

```
mkdir nombredirectorio → crea un directorio con el nombre nombredirectorio
```

```
cd nombredirectorio → entra a un directorio con el nombre nombredirectorio
```

```
cd.. → salimos del directorio y vamos al superior o anterior
```

```
ls → lista los archivos y directorios del directorio en el que nos encontremos
```

A partir de aquí, se pueden ver subdirectorios, que organizan los diferentes archivos del proyecto en paquetes (paquetes clases y principal para este ejemplo).

Por último, cada paquete puede contener un conjunto de ficheros, donde se implementan las clases. Podrá ser una o varias por fichero (típicamente un fichero .java implementa una clase).

```
root@debian:/home/debian/semaforo# ls ./principal/
ClaseColor.java Prueba.java
root@debian:/home/debian/semaforo# ls ./clases/
Estudiante.java Ordenador.java Persona.java Profesor.java
root@debian:/home/debian/semaforo#
```

3.2. Descripción de las clases

<i>Clase ClaseColor (ClaseColor.java)</i>
<pre>package principal; import clases.Profesor; //Clase color el profesor pregunta a un alumno por un color entre (rojo, amarillo y verde) Public class ClaseColor{ public static void main(String[] args) { Profesor teacher = new Profesor(); String color = teacher.preguntacolor(); System.out.println("La respuesta recibida es:" + color);</pre>

```
    }  
}
```

Consideraciones de la clase ClaseColor:

- La clase ClaseColor está contenida en el fichero ClaseColor.java y se incluye en un paquete llamado principal.
- Como parte del código de la clase, se va a utilizar la clase Profesor que forma parte del paquete clases.
- El inicio del programa se lleva a cabo en esta clase, puesto que incluye la función main.
- De la clase profesor se crea un objeto llamado teacher.
- Desde la clase ClaseColor se envía un evento a la clase Profesor (a través del objeto teacher), para que esta última ejecute el método preguntacolor.
- También utilizamos funciones proporcionadas en las librerías de Java, como println que se encuentra en el paquete System.out

Clase Ordenador (Ordenador.java)

```
package clases;  
  
import java.util.Random;  
  
public class Ordenador {  
  
    public Ordenador() {}  
  
    public String color(){  
  
        Random randomGenerator = new Random();  
  
        int randomInt = randomGenerator.nextInt(3);  
  
        if(randomInt == 0){  
  
            return "rojo";  
  
        } else if(randomInt == 1) {  
  
            return "amarillo";  
  
        }else{  
  
            return "verde";  
  
        }  
  
    }  
  
}
```

Consideraciones de la clase ordenador:

- La clase Ordenador utiliza la función Random, incluida en la biblioteca java.util.Random para obtener un número aleatorio entre 0 y 2.
- El método color() devuelve una de las siguientes cadenas de caracteres ("rojo", "amarillo", "verde"), instrucción return.
-

Clase Persona (Persona.java)
<pre>package clases; // Clase utilizada para ser herencia de Estudiante y Profesor public class Persona { // Metodos de clase. Edad y nombre int edad; String nombre; }</pre>

Consideraciones de la clase persona:

- La clase Persona no va a ser instanciada directamente en el programa (crear un objeto de la clase). Se trata de una clase padre, que va a ser utilizada por las clases Estudiante y Profesor para heredar sus características.
- En esta clase se definen las variables de clase int edad y nombre.

Clase estudiante (estudiante.java)
<pre>package clases; public class Estudiante extends Persona{ // Incluye un método de clase que se une a los heredados int curso; public Estudiante() { edad=25; nombre = "Luis"; curso = 1; } }</pre>

```
}

public void presentarse(){

    System.out.println("Soy " + nombre + " Alumno de " + Integer.toString(curso) + " y tengo
    una edad de: " + Integer.toString(edad));

}

public String preguntacolor(){

    presentarse();

    Ordenador mipc = new Ordenador();

    return mipc.color();

}

}
```

Consideraciones de la clase Estudiante:

- Esta clase hereda de la clase Persona, cláusula extends.
- En la clase Estudiante se pueden utilizar las propiedades y métodos definidos en la clase padre.
- Además puede incluir otros métodos y propiedades propias como es *curso*.
- Si hubiéramos sobreescribo una de las propiedades o métodos del padre, en la propia clase, ésta utilizaría sus propios métodos y propiedades en lugar de los heredados.

Clase Profesor (Profesor.java)

```
package clases;

public class Profesor extends Persona{

    public Profesor() {}    // Constructor

    // Hace la pregunta al estudiante sobre el color

    public String preguntacolor(){

        Estudiante alumno = new Estudiante();

        String colorRec = alumno.preguntacolor();

        return colorRec;

    }

}
```

3.3. Uso del traductor/compilador

Como se indicaba anteriormente, cuando se crea un nuevo proyecto Java todos sus recursos "cuelgan" de un determinado directorio en el sistema de archivos ("/home/debian/semaforo" en nuestro caso).

A partir de aquí, podemos ir distribuyendo los ficheros en diferentes paquetes, que corresponderán a subdirectorios del directorio proyecto ("/home/debian/semaforo"). Así hemos incluido los ficheros Ordenador.java, Persona.java, Profesor.java y Estudiante.java en el paquete clases. Además de distribuir los paquetes en el directorio correspondiente, es necesario indicarlo en el código (ver instrucción package).

Para utilizar cada una de estas clases en otros ficheros .java deberemos importarlos (instrucción import). Si dos ficheros forman parte del mismo paquete, no es necesario que sean importados.

```
root@debian:/home/debian/semaforo# ls
clases principal
```

El fichero de inicio del programa ClaseColor.java (incluye la función main), y forma parte del paquete principal.

Primero deberemos compilar las clases que están ubicadas en la carpeta "clases" y que son referenciadas por la clase ClaseColor.

IMPORTANTE: Para que el compilador funcione tal y como se ha descrito en este punto, será necesario que se hayan metido las variables de entorno tal y como se ha indicado en otras prácticas ya realizadas.

Para ello, estando en la carpeta de "semaforo", ejecutar:

```
javac clases/*.java.
```

Esto invocará al compilador de Java para analizar léxica, sintáctica y semánticamente los ficheros, y generar los códigos objeto (bytecode, en formato .class).

Una vez realizado el comando, si compila correctamente no sacará nada por pantalla, pero al ingresar a la carpeta podremos ver los ficheros .class:

```
root@debian:/home/debian/semaforo# javac clases/*.java
root@debian:/home/debian/semaforo# cd clases
root@debian:/home/debian/semaforo/clases# ls
Estudiante.class Ordenador.class Persona.class Profesor.class
Estudiante.java Ordenador.java Persona.java Profesor.java
root@debian:/home/debian/semaforo/clases#
```

Una vez compiladas las clases necesarias, procedemos a compilar la clase principal, ClaseColor, ubicada en la carpeta "principal". Para ello, desde la carpeta "semaforo", ejecutar:

```
javac principal/ClaseColor.java
```



```
root@debian:/home/debian/semaforo# javac principal/ClaseColor.java
root@debian:/home/debian/semaforo#
```

Como resultado, se obtendrá un fichero .class, que se podrá invocar con la JVM de Java:

```
root@debian:/home/debian/semaforo# ls ./principal/
ClaseColor.class  ClaseColor.java
```

Una vez disponibles todos los ficheros bytecode, procedemos a la ejecución del programa desde un terminal. Vamos al directorio de inicio del proyecto y ejecutamos el intérprete java:

```
java principal/ClaseColor
```

El resultado por pantalla será:

```
root@debian:/home/debian/semaforo# java principal/ClaseColor
Soy Luis Alumno de 1 y tengo una edad de: 25
La respuesta recibida es:verde
```

4. Empaquetar .class en fichero .jar

Java nos permite empaquetar todos los ficheros (**.class, junto a otros recursos utilizados**) del proyecto en uno único con extensión .jar (Java ARchives).

Además el formato del fichero .jar es comprimido, por lo que ocupa menos espacio que la información original. A continuación se indica el procedimiento para llevarlo a cabo.

Los ficheros .jar tienen dos finalidades:

1. Permitir crear **paquetes** de archivos .class, es decir, librerías que podré utilizar en otros proyectos.
2. Crear un **ejecutable** en el que se aglutina, de forma comprimida los recursos necesarios para poder ejecutar un proyecto.

4.1. Crear el fichero MANIFEST.MF

En la carpeta de nuestro proyecto, "semaforo", crear un directorio META-INF (¡las mayúsculas son importantes!) y dentro un fichero MANIFEST.MF. Este fichero indica, entre otras cosas, cuál será la clase principal. A menudo el fichero MANIFEST.MF contiene una única línea:

```
Main-Class: principal.ClaseColor
```

donde se indica que ClaseColor.class (del paquete principal) es la clase que contiene el método main.

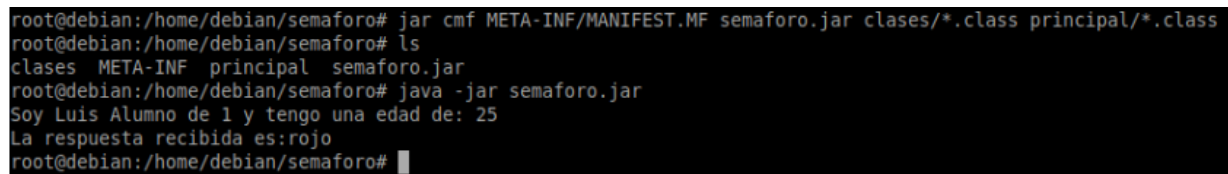
Nota: el fichero MANIFEST.MF se puede crear con cualquier editor de texto. Hay que introducir un fin de línea (enter) tras la última línea con texto.

4.2. Crear el fichero semaforo.jar

El paquete JDK proporciona una serie de ejecutables localizados su ruta /bin ("/usr/java/jdk-17.0.1/bin" en nuestra distribución). Ya se ha trabajado con algunos de ellos como el compilador de java (javac) y con el intérprete (java) en prácticas anteriores. Otro de los programas a nuestra disposición es jar, que permite empacar todos los recursos de un proyectos Java en un fichero con extensión .jar.

Desde consola del sistema, ejecutar el programa jar con los siguientes parámetros:

```
jar cmf META-INF/MANIFEST.MF semaforo.jar clases/*.class principal/*.class
```



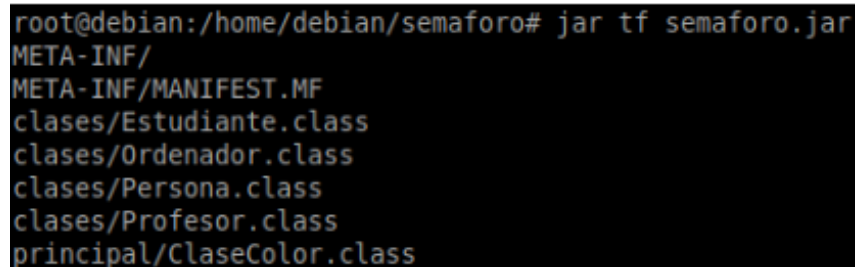
```
root@debian:/home/debian/semaforo# jar cmf META-INF/MANIFEST.MF semaforo.jar clases/*.class principal/*.class
root@debian:/home/debian/semaforo# ls
clases META-INF principal semaforo.jar
root@debian:/home/debian/semaforo# java -jar semaforo.jar
Soy Luis Alumno de 1 y tengo una edad de: 25
La respuesta recibida es:rojo
root@debian:/home/debian/semaforo#
```

Observa en la imagen una ejecución del proyecto utilizando el fichero .jar obtenido:

```
java -jar semaforo.jar
```

El programa jar también nos permite ver el contenido de proyecto comprimidos .jar:

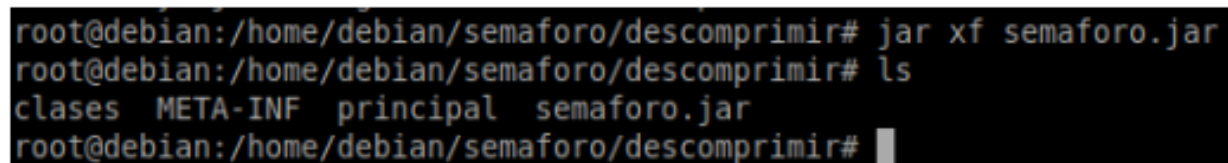
```
jar tf semaforo.jar
```



```
root@debian:/home/debian/semaforo# jar tf semaforo.jar
META-INF/
META-INF/MANIFEST.MF
clases/Estudiante.class
clases/Ordenador.class
clases/Persona.class
clases/Profesor.class
principal/ClaseColor.class
```

Si deseamos descomprimir la información del fichero .jar, para obtener sus ficheros originales. Mejor crear una carpeta, depositar el jar dentro, y descomprimir:

```
jar xf semaforo.jar
```



```
root@debian:/home/debian/semaforo/descomprimir# jar xf semaforo.jar
root@debian:/home/debian/semaforo/descomprimir# ls
clases META-INF principal semaforo.jar
root@debian:/home/debian/semaforo/descomprimir#
```

Actividad. Ejecución desde consola

1. Crea el árbol de directorios necesario (semaforo, principal, clases).
2. Crea las clases en un editor de textos cada una en el directorio que corresponda.
3. Compila y ejecuta el programa.
4. Crea el fichero MANIFEST.MF
5. Exporta el proyecto
6. Compártelo con un compañero.
7. Descomprímelo en una nueva carpeta y comprueba que funciona el programa de tu compañero