ENTORNOS DE DESARROLLO

DESARROLLO JAVA

JAVA DESDE CONSOLA AVANZADO

Alberto Martínez Pérez

1º CFGS DESARROLLO DE APLICACIONES WEB (daw)

ÍNDICE

[1. CREACIÓN DEL ÁRBOL DE DIRECTORIOS 3](#_Toc119042141)

[2. CREACIÓN DE LAS CLASES 4](#_Toc119042142)

[3. COMPILACIÓN Y EJECUCIÓN DEL PROGRAMA 8](#_Toc119042143)

[4. CREACIÓN DEL FICHERO MANIFEST.MF 9](#_Toc119042144)

[5. EXPORTACIÓN 10](#_Toc119042145)

ÍNDICE DE FIGURAS

[*Fig. 1 Terminal de Ubuntu. Comandos de creación de directorios. 3*](file:///C:\Users\alber\Desktop\UD2%20-%20E6%20-%20ACTIVIDADES%20REFLEXIÓN%20(II).docx#_Toc119040075)

[*Fig. 2 Directorio principal (seleccionado) y subdirectorios en su interior. 3*](file:///C:\Users\alber\Desktop\UD2%20-%20E6%20-%20ACTIVIDADES%20REFLEXIÓN%20(II).docx#_Toc119040076)

[*Fig. 3 Clase principal. ClaseColor.java 4*](file:///C:\Users\alber\Desktop\UD2%20-%20E6%20-%20ACTIVIDADES%20REFLEXIÓN%20(II).docx#_Toc119040077)

[*Fig. 4 Archivo ClaseColor.java 4*](file:///C:\Users\alber\Desktop\UD2%20-%20E6%20-%20ACTIVIDADES%20REFLEXIÓN%20(II).docx#_Toc119040078)

[*Fig. 5 Clase Ordenador.java 5*](file:///C:\Users\alber\Desktop\UD2%20-%20E6%20-%20ACTIVIDADES%20REFLEXIÓN%20(II).docx#_Toc119040079)

[*Fig. 6 Clase Persona.java 5*](file:///C:\Users\alber\Desktop\UD2%20-%20E6%20-%20ACTIVIDADES%20REFLEXIÓN%20(II).docx#_Toc119040080)

[*Fig. 7 Clase Estudiante.java 6*](file:///C:\Users\alber\Desktop\UD2%20-%20E6%20-%20ACTIVIDADES%20REFLEXIÓN%20(II).docx#_Toc119040081)

[*Fig. 8 Clase Profesor.java 6*](file:///C:\Users\alber\Desktop\UD2%20-%20E6%20-%20ACTIVIDADES%20REFLEXIÓN%20(II).docx#_Toc119040082)

[*Fig. 9 Subdirectorio clases con las cuatro clases creadas. 7*](file:///C:\Users\alber\Desktop\UD2%20-%20E6%20-%20ACTIVIDADES%20REFLEXIÓN%20(II).docx#_Toc119040083)

[*Fig. 10 Terminal de Ubuntu. Compilación de clases. 8*](file:///C:\Users\alber\Desktop\UD2%20-%20E6%20-%20ACTIVIDADES%20REFLEXIÓN%20(II).docx#_Toc119040084)

[*Fig. 11 Compilación clase principal. 8*](file:///C:\Users\alber\Desktop\UD2%20-%20E6%20-%20ACTIVIDADES%20REFLEXIÓN%20(II).docx#_Toc119040085)

[*Fig. 12 Ejecución del programa. 8*](file:///C:\Users\alber\Desktop\UD2%20-%20E6%20-%20ACTIVIDADES%20REFLEXIÓN%20(II).docx#_Toc119040086)

[*Fig. 13 Creación del directorio META-INF. 9*](file:///C:\Users\alber\Desktop\UD2%20-%20E6%20-%20ACTIVIDADES%20REFLEXIÓN%20(II).docx#_Toc119040087)

[*Fig. 14 Archivo MANIFEST.MF 9*](file:///C:\Users\alber\Desktop\UD2%20-%20E6%20-%20ACTIVIDADES%20REFLEXIÓN%20(II).docx#_Toc119040088)

[*Fig. 15 Comando para generar el archivo semaforo.jar. 10*](file:///C:\Users\alber\Desktop\UD2%20-%20E6%20-%20ACTIVIDADES%20REFLEXIÓN%20(II).docx#_Toc119040089)

[*Fig. 16 Archivo semaforo.jar. 10*](file:///C:\Users\alber\Desktop\UD2%20-%20E6%20-%20ACTIVIDADES%20REFLEXIÓN%20(II).docx#_Toc119040090)

[*Fig. 17 Ejecución de semaforo.jar. 10*](file:///C:\Users\alber\Desktop\UD2%20-%20E6%20-%20ACTIVIDADES%20REFLEXIÓN%20(II).docx#_Toc119040091)

[*Fig. 18 Visualización de los archivos contenidos en semaforo.jar. 11*](file:///C:\Users\alber\Desktop\UD2%20-%20E6%20-%20ACTIVIDADES%20REFLEXIÓN%20(II).docx#_Toc119040092)

[*Fig. 19 Comando de descompresión (abajo) y resultado (arriba). 11*](file:///C:\Users\alber\Desktop\UD2%20-%20E6%20-%20ACTIVIDADES%20REFLEXIÓN%20(II).docx#_Toc119040093)

# CREACIÓN DEL ÁRBOL DE DIRECTORIOS

Vamos a crear una serie de directorios necesarios para organizar nuestro programa.

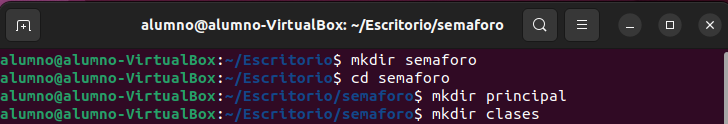
Para crear un directorio a través de la terminal de Ubuntu vamos a usar el comando mkdir nombreDirectorio (fig. 1).

Fig. Terminal de Ubuntu. Comandos de creación de directorios.

El primero de los comandos nos va a generar un directorio principal, una vez creado haremos un salto al interior de este directorio y crearemos dos subdirectorios, uno para la clase principal y otro para el resto de clases.

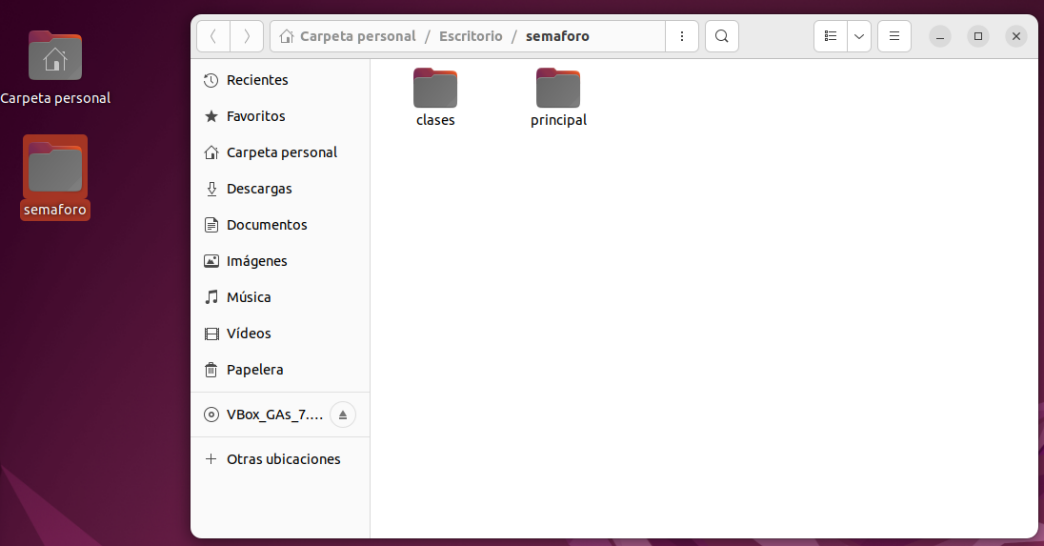


Fig. Directorio principal (seleccionado) y subdirectorios en su interior.

# CREACIÓN DE LAS CLASES

Vamos a empezar creando la clase principal, para ello haremos uso de algún procesador de textos, por ejemplo, nano o gedit.

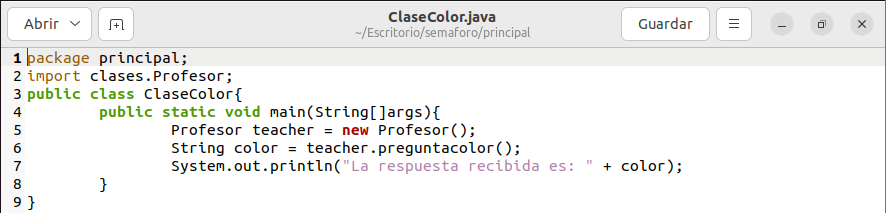
Comenzaremos con la clase principal, para ello realizaremos un salto hacia el directorio principal y, una vez ahí, escribiremos el código que se ve en la fig. 3. Guardamos el archivo como ClaseColor.java (fig. 4).

Fig. Clase principal. ClaseColor.java

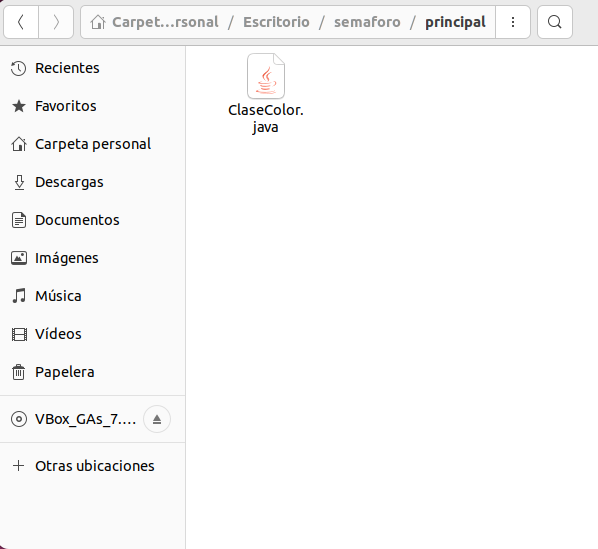


Fig. Archivo ClaseColor.java

Esta clase principal contiene el método main y, por tanto, va a ser el punto de inicio de nuestro programa. Además, podemos ver que se va a importar la clase Profesor (import clases.Profesor) por lo que vamos a tener que tener al menos una clase Profesor dentro del subdirectorio clases.

Dentro de este método main vamos a crear un nuevo objeto profesor de nombre teacher y vamos a guardar un color dentro de un atributo String color a través de la llamada a un método propio de la clase Profesor. Por último, se imprimirá un mensaje con el texto “La respuesta recibida es:” seguido del color asignado en el atributo String color.

Ahora vamos a realizar las otras clases con las que trabajará el programa, es decir, las clases Ordenador, Persona, Estudiante y Profesor.

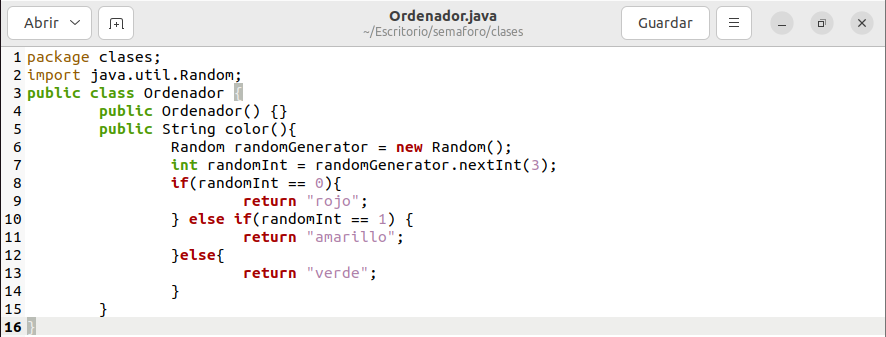
* Ordenador (fig. 5).

Fig. Clase Ordenador.java

Esta clase va a utilizar la función Random del paquete java.util el cual no está integrado dentro de los paquetes básicos de la JDK por lo que vamos a necesitar importarlo desde la API.

Dentro de esta clase tenemos dos métodos, un constructor que no tiene atributos propios y una función que asigna un color aleatorio en función de un valor conseguido con la función Random, estos valores van del 0 al 2, si es 0 es “rojo”, si es 1 es “amarillo” y si es 2 es “verde”.

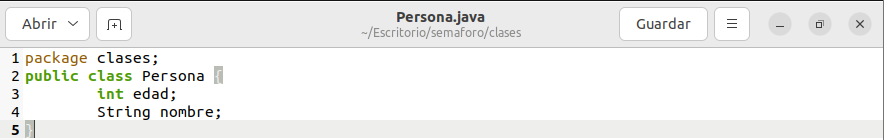
* Persona (fig. 6).

Fig. Clase Persona.java

La clase Persona no va a tener ningún método en su interior va a funcionar como una clase padre de la que heredarán sus atributos (int edad y String nombre) las clases hijas Estudiante y Profesor.

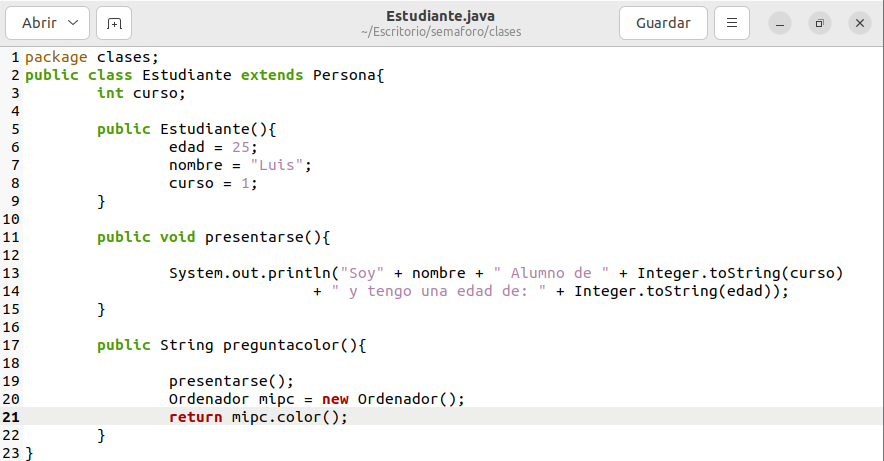
* Estudiante (fig. 7).

Fig. Clase Estudiante.java

Esta clase como se ha comentado antes es una clase hija de la clase Persona, por tanto, deberá tener la cláusula extends a continuación del nombre de la clase.

Al heredar de Persona no vamos a tener que volver a crear las variables int edad y String nombre para utilizarlas en el constructor Estudiante, simplemente iniciaremos las variables. Eso sí, toda variable que no se haya creado en Persona (es decir, todo atributo que no sea de Persona) debe crearse antes del constructor, es el caso del atributo int curso.

Esta clase además tiene una función de tipo void que genera un mensaje donde se imprimirán los diferentes valores que se hayan asignado a las variables.

Por último, tiene una función que genera un nuevo objeto Ordenador (en este caso mipc) y solicita la función propia de la clase Ordenador para asignar un color y que se explicó más arriba.

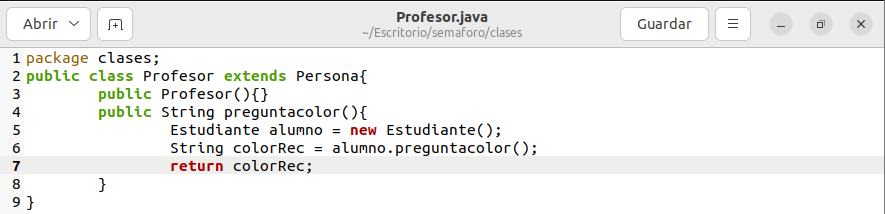
* Profesor (fig. 8).

Fig. Clase Profesor.java

Es la última clase, vuelve a heredar de Persona para su constructor (aunque en este caso no iniciaremos ninguna variable) y, además, tiene un método pregunta color que generará un nuevo objeto Estudiante, de nombre alumno, solicitará una función propia de Estudiante para guardar un color en una variable String colorRec y devolverá ese valor.

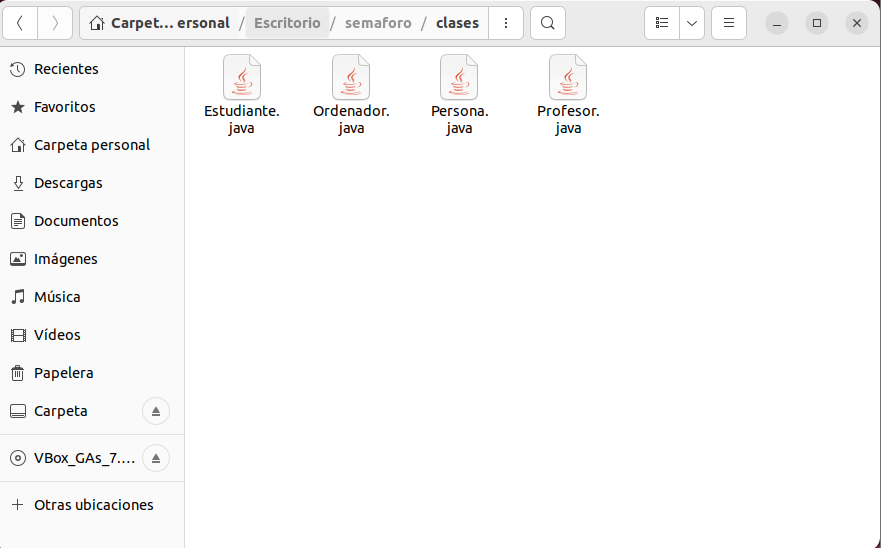
Si hemos hecho todo bien deberíamos tener 4 archivos .java dentro del subdirectorio clases (fig. 9).

Fig. Subdirectorio clases con las cuatro clases creadas.

# COMPILACIÓN Y EJECUCIÓN DEL PROGRAMA

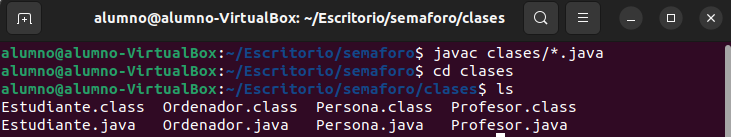
Lo primero que vamos a hacer es compilar todos los .java del subdirectorio “Clases” para ello abriremos una terminal que partirá del directorio principal (semáforo) y vamos a utilizar el comando javac nombreDirectorio/\*.java (en este caso javac clases/\*.java). Además, podemos usar el comando ls para ver cuántos archivos tenemos en esta carpeta (fig. 10).

Fig. Terminal de Ubuntu. Compilación de clases.

Comando 1: Compilación de todos los archivos .java en el directorio clases. Comando 2: Salto al directorio clases. Comando 3: Listado de archivos.

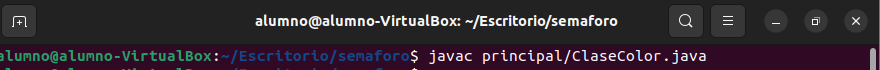
También vamos a realizar una compilación del método main (fig. 11).

Fig. Compilación clase principal.

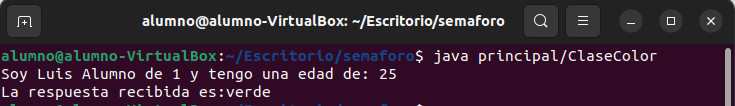
Para ejecutar el código vamos a usar el comando javac/nombreDirectorio/nombreClasePrincipal (fig. 12).

Fig. Ejecución del programa.

# CREACIÓN DEL FICHERO MANIFEST.MF

Vamos a empaquetar todos los .class en un único fichero .jar y para ello lo primero que tendremos que crear es un fichero MANIFEST.MF.

Para ello lo primero será crear un directorio META-INF (fig. 13) dentro del directorio semáforo y dentro de este habrá que crear un fichero MANIFEST.MF, este fichero va a servir para indicar cuál es la clase principal por lo que sólo va a contender una línea, en nuestros caso Main-Class: principal.ClaseColor (fig. 14).

Fig. Creación del directorio META-INF.



Fig. Archivo MANIFEST.MF

# EXPORTACIÓN

Por último, vamos a exportar el programa. Para facilitar esto, vamos a crear un archivo con extensión .jar. Para ello habrá que seguir una serie de pasos introducir el siguiente comando (fig. 15):

Fig. Comando para generar el archivo semaforo.jar.

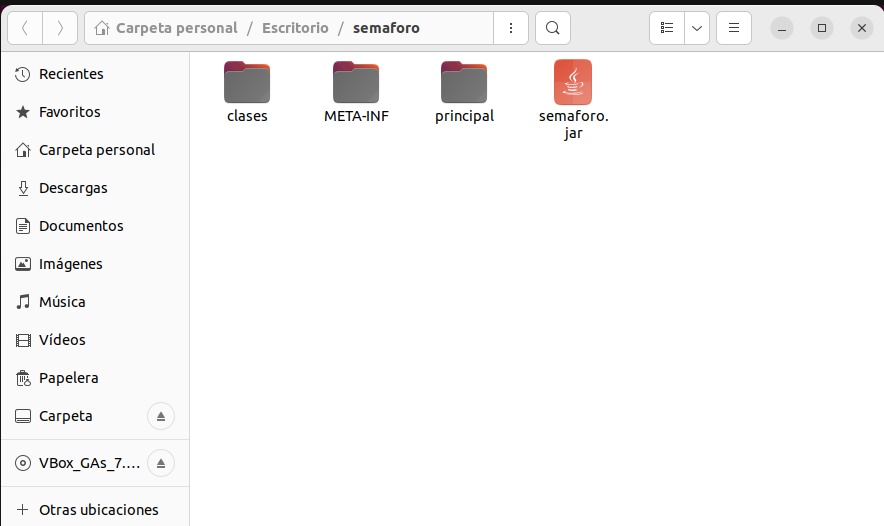
Con esto conseguimos como se ha dicho un archivo .jar que se encontrará en el directorio principal (fig. 16):

Fig. Archivo semaforo.jar.

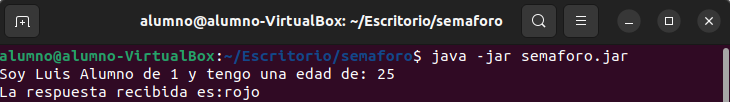
Una vez conseguido el archivo podemos probar su funcionamiento con el comando java -jar nombreArchivoJar.jar (fig. 17):

Fig. Ejecución de semaforo.jar.

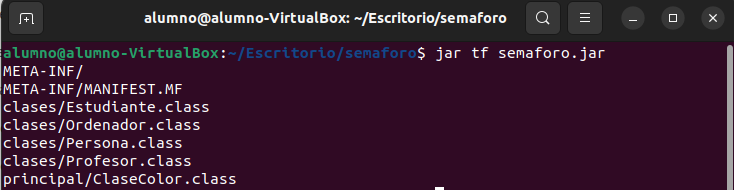
Otras opciones que tenemos con el jar son ver los archivos que se encuentran dentro de él, para ello haremos uso del comando jar tf nombreArchivoJar.jar (fig. 18):

Fig. Visualización de los archivos contenidos en semaforo.jar.

Por último, podemos descomprimir el jar con el comando jar xf semaforo.jar (fig. 19):

Fig. Comando de descompresión (abajo) y resultado (arriba).