**ACTIVIDAD 11**

**Enunciado**

Realiza un programa Java (**Actividad11**) que ejecute el programa que se muestra a continuación (**Ejemplo2**). El programa Actividad11 sólo requiere adaptar el valor que se le asigna a la variable *comando*. En ambos programas se usan las clases Java Runtime y Processpara la gestión de procesos. Incluye los dos programas Java en el paquete por defecto que ofrece el IDE Eclipse.

Compila tanto el programa del enunciado (Ejemplo2) como el nuevo programa (Actividad11) desde la línea de comandos, aportando evidencias de cómo lo has hecho.

Después de que la compilación esté libre de errores, ejecuta el programa Java de nombre Actividad11 desde la línea de comandos, aportando evidencias de tanto su ejecución como del resultado de misma.

import java.io.\*;

public class Ejemplo2 {

public static void main(String[] args){

Runtime r=Runtime.*getRuntime*();

//String comando="ls -al"; //Descomentar la línea para Linux

//String comando="CMD C/ DIR"; //Descomentar la línea para Windows

Process p=null;

try {

p = r.exec (comando);

InputStream is = p.getInputStream();

BufferedReader br = new BufferedReader (new InputStreamReader(is));

String linea;

while((linea = br.readLine()) != null) // lee una linea

System.*out*.println(linea);

br.close();

}

catch (Exception e) {

e.printStackTrace();

}

///// Comprobación: Valor= 0 bien, Valor = - 1 mal

int exitVal;

try {

exitVal=p.waitFor();

System.*out*.println ("Valor de Salida "+exitVal);

} catch (InterruptedException e){

e.printStackTrace();

}

}

}

**Solución:**

Incluir aquí el código del programa Actividad11.java

package application;

public class Actividad11 {

public static void main(String[] args){

Runtime r=Runtime.getRuntime();

String comando="java"; //Descomentar la línea para Linux

//String comando="CMD C/ DIR"; //Descomentar la línea para Windows

try {

String claseAEjecutar = "application/Ejemplo2";

// Crea un proceso para ejecutar el comando

ProcessBuilder builder = new ProcessBuilder(comando, claseAEjecutar);

// Redirige la salida estándar y la salida de error al proceso actual

builder.redirectOutput(ProcessBuilder.Redirect.INHERIT);

builder.redirectError(ProcessBuilder.Redirect.INHERIT);

// Ejecuta el proceso

Process proceso = builder.start();

// Espera a que el proceso termine

int codigoSalida = proceso.waitFor();

}

catch (Exception e) {

System.out.println("error en:"+comando);

e.printStackTrace();

}

}

}

package application;

import java.io.BufferedReader;

import java.io.IOException;

import java.io.InputStream;

import java.io.InputStreamReader;

public class Ejemplo2 {

public static void main(String[] args) {

// Comando para listar archivos en el directorio actual (Linux/Mac)

String comando = "ls -al";

ProcessBuilder processBuilder = new ProcessBuilder(comando.split(" "));

processBuilder.redirectErrorStream(true);

try {

Process proceso = processBuilder.start();

// Lee la salida del proceso

InputStream is = proceso.getInputStream();

BufferedReader br = new BufferedReader(new InputStreamReader(is));

String linea;

while ((linea = br.readLine()) != null) {

System.out.println(linea);

}

// Espera a que el proceso termine y obtiene el código de salida

int exitCode = proceso.waitFor();

System.out.println("Valor de Salida: " + exitCode);

} catch (IOException | InterruptedException e) {

e.printStackTrace();

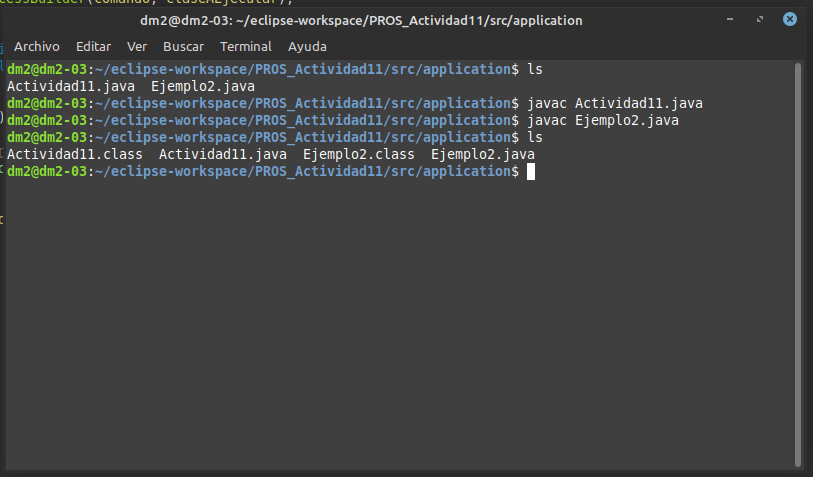
}

}

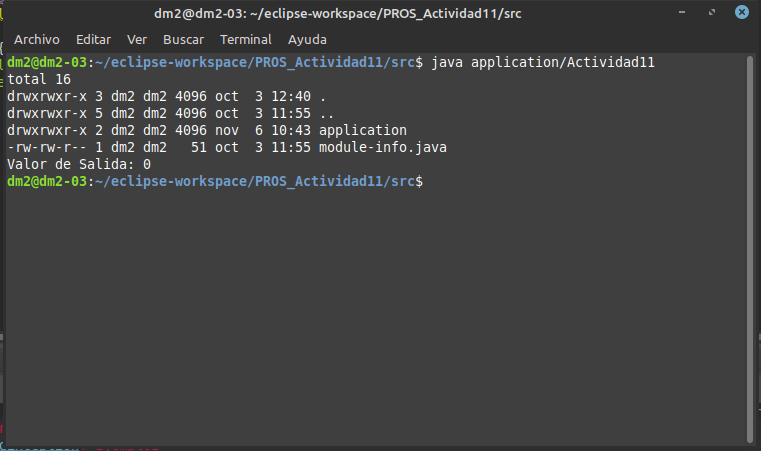
}

**Evidencias de la solución:**

* Compilación desde la línea de comandos de los programas fuente Ejemplo2.java y Actividad11.java:



* Ejecución del programa Actividad11.class desde la línea de comandos:

****

**Resolución de problemas:**

No se han producido errores en la realización del ejercicio.

**CRITERIOS DE CORRECCIÓN**

1. Nivel de ajuste a la nomenclatura de objetos definida en el enunciado de la actividad y nivel de ajuste al almacenamiento de la información requerida en las subcarpetas definidas para la actividad y unidad didáctica correspondientes (15%).
2. Correcto funcionamiento de los programas desarrollados en C (35%).
3. Ajuste a los resultados esperados (35%).
4. Claridad del código C (15%).
5. Explicación detallada de cómo se han resuelto problemas que hayan podido surgir en la realización de la actividad (hasta un 15% adicional sobre la nota obtenida).
6. Avisos o errores de compilación (-15%).
7. Nivel de comentarios en el código fuente (-15%).