Ejercicio 1.- Crea un programa llamado *Ejercicio1.java* que reciba desde los argumentos del main() un nombre y lo lance a su ejecución, con las siguientes características:

- Si el programa finaliza correctamente utiliza *System.exit(1)*, en caso que no se hayan introducido los argumentos correctamente utiliza *System.exit(-1)*.

- Comprueba el valor de salida del proceso que se ejecuta, de tal manera que:

- Si se ejecuta correctamente muestra en pantalla la información que devuelve el programa lanzado (es importante invocar programas que generen algo, como "DIR").

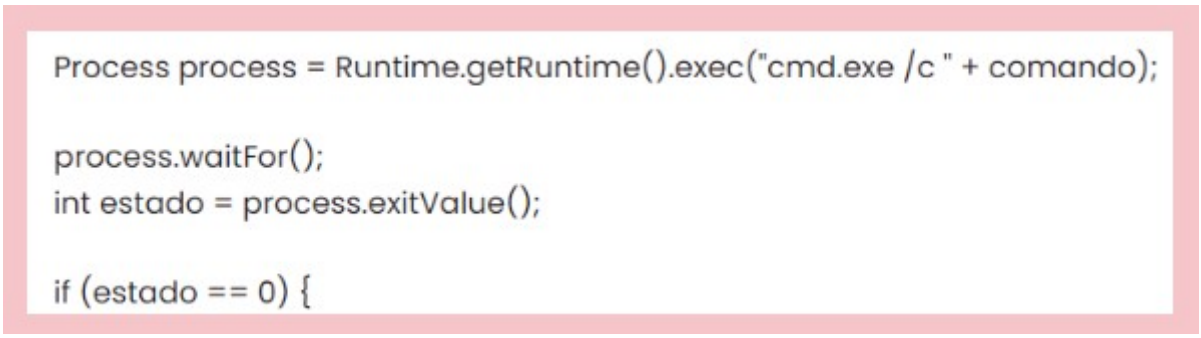
-Si no se ejecuta correctamente el lanzamiento, debe mostrar los errores en un archivo llamado "errores.txt".

********* Lo plantea muy complicado pero a cambio del otro ejemplo aquí introduce waitFor(correctamente) ********************

```
package Ejercicio13;
import java.io.BufferedReader;
import java.io.BufferedWriter;
import java.io.File;
import java.io.FileWriter;
import java.io.IOException;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.util.logging.Level;
import java.util.logging.Logger;

public class Main {


    public static void main(String[] args) {

        String comando = null;
        Process proceso = null;
        String flujo = null;

        if( args.length > 0 ){

            comando = args[0];

            proceso = lanzaProceso(comando);
            //Debemos esperar a que termine el proceso
            try {
                proceso.waitFor();
            } catch (InterruptedException ex) {
                Logger.getLogger(Main.class.getName()).log(Level.SEVERE, null, ex);
            }

            InputStream is = proceso.getInputStream();
            flujo = leeFlujo(is);
            if(flujo!=null)
                System.out.println(flujo);
            else
                System.out.println("Problema al leer el flujo");

            System.exit(1);
        }else{
            try {
                escribeEnFichero("errores.txt", "Comando no encontrado", 2);
            } catch (IOException ex) {
                Logger.getLogger(Main.class.getName()).log(Level.SEVERE, null, ex);
            }

            System.exit(-1);
        }
    }
```

```
    /**
     * Función para leer un flujo de datos
     * @return flujo Flujo de datos leido
     */
    public static String leeFlujo(InputStream is){
        String linea = null;
        String flujo = null;
        BufferedReader br = new BufferedReader(new InputStreamReader(is));
        try {
            linea = br.readLine();
            while(linea!=null){
                flujo = flujo + linea + "\n";
                linea = br.readLine();
            }
        } catch (IOException ex) {
            System.out.println("Error de entrada/salida. " + ex.getMessage());
        }
        try {
            is.close();
            br.close();
        } catch (IOException ex) {
            Logger.getLogger(Main.class.getName()).log(Level.SEVERE, null, ex);
        }
        return flujo;
    }

    /**
     * Método mediante el cual escribimos el texto recibido en un fichero
     * @param nombreFichero Nombre del fichero
     * @param texto Texto que escribiremos en el fichero
     * @param modo Valor 1 para agregar información al fichero, valor 2 para sobreescribir
     * @throws IOException
     */
    public static void escribeEnFichero(String nombreFichero, String texto, int modo) throws IOException{

        File f = new File(nombreFichero);
        //Si el archivo no existe lo creamos
        if (!f.exists()){
            f.createNewFile();
        }
        //modo=1 agrega información | modo=2 sobreescribe
        boolean agrega =(modo==1)?true:false;

        FileWriter fw = new FileWriter(f.getAbsoluteFile(), agrega);
        BufferedWriter bw = new BufferedWriter(fw);

        bw.write(texto);
        bw.newLine();

        if( bw != null ) bw.close();
        if( fw != null ) fw.close();
    }

    /**
     * Función para lanzar un proceso
     * @param cadena Comando a ejecurar
     * @return Proceso
     */
    public static Process lanzaProceso(String cadena){

        Process proceso = null;

        //Lanzamos el proceso
        try {
            proceso = Runtime.getRuntime().exec("cmd /c "+cadena);
        } catch (IOException ex) {
            System.out.println("Error de entrada/salida. " + ex.getMessage());
        } catch (SecurityException ex){
            System.out.println("Error de seguridad. " + ex.getMessage());
        } catch (Exception ex){
            System.out.println("Error. " + ex.getMessage());
        }
        return proceso;
    }
}
```

***IMPORTANTE



Realmente, el método waitFor(); y exitValue(); devuelven el mismo valor:

Por lo que puedes hacer directamente...

int estado = process.waitFor();

```
if (estado == 0) {
...
}
```