RUNTIME y PROCESSBUILDER

EJERCICIO 1. RUNTIME

```
/*1. Realiza un programa en Java que ejecute el comando
correspondiente con
el sistema operativo donde se esté ejecutando (Windows o Linux),
muestre
la dirección IP a través de la consola y muestre su resultado por
pantalla.
*/


import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;

public class Ejercicio1 {

   public static void main(String[] args) {
       String os = System.getProperty("os.name").toLowerCase();
       String command;

       // Determinar el sistema operativo y asignar el comando
adecuado
       if (os.contains("win")) {
           command = "ipconfig";//Windows
       } else if (os.contains("nix") || os.contains("nux") ||
os.contains("aix")) {
           command = "ip addr"; // Comando para sistemas
Linux/Unix
       } else {
           System.out.println("Sistema operativo no compatible.");
           return; // Termina el programa si el SO no es Windows o
Linux
       }

       System.out.println("-> Ejecutando el comando '" + command +
"' para " + System.getProperty("os.name") + "...");

System.out.println("-----------------------------------------------
---");

       try {
           // 2. Ejecutar el comando en el sistema
           Process process = Runtime.getRuntime().exec(command);

           // 3. Leer la salida del comando ejecutado
           BufferedReader reader = new BufferedReader(new
InputStreamReader(process.getInputStream()));
           String linea;
```

RUNTIME y PROCESSBUILDER

```
            // 4. Imprimir cada línea de la salida en la consola
            while ((linea = reader.readLine()) != null) {
                System.out.println(linea);
            }

            int exitCode = process.waitFor();

System.out.println("-----------------------------------------------
---");
            System.out.println("-> El comando finalizo con código
de salida: " + exitCode);

        } catch (IOException | InterruptedException e) {
            System.err.println("Error al ejecutar comando:");
            e.printStackTrace();
        }
    }
}
```

Consola:

-> Ejecutando el comando 'ipconfig' para Windows 11...

Configuraci�n IP de Windows

Adaptador de Ethernet Ethernet:

   Estado de los medios. . . . . : medios desconectados
   Sufijo DNS espec�fico para la conexi�n. . : iesps.local

Adaptador de Ethernet vEthernet (WSL (Hyper-V firewall)):

   Sufijo DNS espec�fico para la conexi�n. . :
   V�nculo: direcci�n IPv6 local. . . : fe80::ce70:3202:c3af:dac4%52
   Direcci�n IPv4. . . . . : 172.30.16.1
   M�scara de subred . . . . . : 255.255.240.0
   Puerta de enlace predeterminada . . . . . :

Adaptador de Ethernet Ethernet 2:

   Sufijo DNS espec�fico para la conexi�n. . :
   V�nculo: direcci�n IPv6 local. . . : fe80::9a8d:31b4:25d0:e1dd%45
   Direcci�n IPv4. . . . . : 192.168.56.1
   M�scara de subred . . . . . : 255.255.255.0
   Puerta de enlace predeterminada . . . . . :

RUNTIME y PROCESSBUILDER

Adaptador de LAN inal�mbrica Conexi�n de �rea local* 1:

   Estado de los medios. . . . . : medios desconectados
   Sufijo DNS espec�fico para la conexi�n. . :

Adaptador de LAN inal�mbrica Conexi�n de �rea local* 10:

   Estado de los medios. . . . . : medios desconectados
   Sufijo DNS espec�fico para la conexi�n. . :

Adaptador de LAN inal�mbrica Wi-Fi:

   Sufijo DNS espec�fico para la conexi�n. . :
   V�nculo: direcci�n IPv6 local. . . : fe80::ce3b:1c15:36cc:917d%16
   Direcci�n IPv4. . . . . : 172.30.4.213
   M�scara de subred . . . . : 255.255.252.0
   Puerta de enlace predeterminada . . . . . : 172.30.4.251

-> El comando finalizo con código de salida: 0

RUNTIME y PROCESSBUILDER

EJERCICIO 2. RUNTIME

```java
/*2. Utiliza el objeto Runtime para obtener información del equipo
donde se
ejecuta el programa. Muestra la información acerca del número de
procesadores disponibles.
*/

public class Ejercicio2 {

   public static void main(String[] args) {
        // 1. Obtener la instancia única del objeto Runtime
        Runtime runtime = Runtime.getRuntime();

        // 2. Usar el metodo para obtener el número de procesadores
disponibles
        int numberOfProcessors = runtime.availableProcessors();

        // 3. Mostrar la información por pantalla de forma clara
        System.out.println("-> Obteniendo información del
sistema...");

System.out.println("----------------------------------------");
        System.out.println("Número de procesadores disponibles: " +
numberOfProcessors);

System.out.println("----------------------------------------");
   }
}
```

Consola:

```
-> Obteniendo información del sistema...
----------------------------------------
Número de procesadores disponibles: 8
----------------------------------------

Process finished with exit code 0
```

Daniel Remiro Laguna

RUNTIME y PROCESSBUILDER

EJERCICIO 1. PROCESSBUILDER

```
/*1. Modificar el ejercicio 1 para utilizar ProcessBuilder y que
guarde el
resultado en un archivo txt. Necesitarás utilizar el metodo
redirectOutput.*/


import java.io.File;
import java.io.IOException;

public class Ejercicio2_1 {

   public static void main(String[] args) {
        String os = System.getProperty("os.name").toLowerCase();
        String command;
        String outputFileName = "ip_result.txt";

        // 1. Determinar el comando según el sistema operativo
        if (os.contains("win")) {
            command = "ipconfig";
        } else if (os.contains("nix") || os.contains("nux") ||
os.contains("aix")) {
            command = "ip";
        } else {
            System.out.println("Sistema operativo no compatible.");
            return;
        }

        System.out.println("-> Ejecutando el comando para " +
System.getProperty("os.name") + "...");
        System.out.println("-> El resultado se guardará en el
archivo: " + outputFileName);

        try {
            // 2. Crear una instancia de ProcessBuilder
            ProcessBuilder processBuilder;
            if (os.contains("linux") || os.contains("nix")) {

                processBuilder = new ProcessBuilder("ip", "addr");
            } else {

                processBuilder = new ProcessBuilder(command);
            }


            // 3. Redirigir la salida estándar del proceso al
archivo especificado
            File outputFile = new File(outputFileName);
```

RUNTIME y PROCESSBUILDER

```
            processBuilder.redirectOutput(outputFile);

            // 4. Iniciar el proceso
            Process process = processBuilder.start();

            // 5. Esperar a que el proceso termine
            int exitCode = process.waitFor();


System.out.println("-----------------------------------------------
---");
            if (exitCode == 0) {
                System.out.println("¡Exito! La información de la
red ha sido guardada en '" + outputFile.getAbsolutePath() + "'.");
            } else {
                System.out.println(" El comando finalizo con un
código de error: " + exitCode);
            }

        } catch (IOException | InterruptedException e) {
            System.err.println("Ocurrio un error al ejecutar el
proceso:");
            e.printStackTrace();
        }
    }
}
```

Consola:

-> Ejecutando el comando para Windows 11...
-> El resultado se guardará en el archivo: ip_result.txt

¡Exito! La información de la red ha sido guardada en
'C:\Users\dremi\Desktop\DAM2A\AAD\MiPrimerJuego\ip_result.txt'.

RUNTIME y PROCESSBUILDER

EJERCICIO 2. PROCESSBUILDER

```
/*2. Utilizando ProcessBuilder, realiza un programa en Java que
admite como
parámetro de entrada el comando a ejecutar en la consola del
sistema
operativo y muestre en pantalla el resultado. (Not all the command
might
work)*/


import java.io.BufferedReader;
import java.io.InputStreamReader;
import java.io.IOException;
import java.util.Arrays;

public class Ejercicio2_2 {

   public static void main(String[] args) {
       // 1. Validar que se ha proporcionado un comando
       if (args.length == 0) {
           System.out.println("Error: Debes proporcionar un
comando a ejecutar.");
           System.out.println("Ejemplo (Windows): java
CommandExecutor ipconfig");
           System.out.println("Ejemplo (Linux):   java
CommandExecutor ls -l");
           return;
       }

       System.out.println("Ejecutando comando: " + String.join(" ",
args));

System.out.println("-----------------------------------------------
---");

       try {
           // 2. Crear una instancia de ProcessBuilder con los
argumentos recibidos
           ProcessBuilder pb = new ProcessBuilder(args);

           pb.redirectErrorStream(true);

           // 3. Iniciar el proceso
           Process process = pb.start();

           // 4. Leer y mostrar la salida del proceso
           BufferedReader reader = new BufferedReader(new
InputStreamReader(process.getInputStream()));
```

RUNTIME y PROCESSBUILDER

```
            String line;
            while ((line = reader.readLine()) != null) {
                System.out.println(line);
            }

            // 5. Esperar a que el proceso termine y mostrar el
código de salida
            int exitCode = process.waitFor();

System.out.println("-----------------------------------------------
---");
            System.out.println("El comando finalizó con código de
salida: " + exitCode);
            if (exitCode != 0) {
                System.out.println("(Un código de salida distinto
de 0 usualmente indica un error)");
            }

        } catch (IOException e) {
            System.err.println("Error al ejecutar el comando.
¿Estás seguro de que el comando '" + args[0] + "' existe?");
        } catch (InterruptedException e) {
            System.err.println("El proceso fue interrumpido.");
            Thread.currentThread().interrupt();
        }
    }
}
```

Consola:

Error: Debes proporcionar un comando a ejecutar.
Ejemplo (Windows): java CommandExecutor ipconfig
Ejemplo (Linux): java CommandExecutor ls -l

Process finished with exit code 0