MEMORIA DE EJERCICIOS DE PROCESOS

Por: María Chaparro Caballero

INDICE

Introducción:

En este documento explicaremos los ejercicios de comunicación de procesos, sus canales y todas las nuevas funciones aprendidas.

Ejercicio 1programación multiprocesos Sincronización y comunicación entre dos procesos

Enunciado:

Desarrollar un programa en Java que cree dos procesos hijos y sincronice la salida estándar del primer proceso con la entrada estándar del segundo proceso.

1. Crear un programa en Java que: o Cree dos procesos hijos.

o El primer proceso debe ejecutar el comando dir o El segundo proceso debe ejecutar el comando findstr /i "d" (que busca en la salida del comando anterior las líneas que contienen la letra "d", de manera no sensible a mayúsculas).

Ayuda para la implementación:

1. Crear el primer proceso: Utilizar ProcessBuilder para ejecutar el comando dir.

2. Redirigir la salida estándar: Usar getInputStream() para obtener la salida estándar del primer proceso.

3. Crear el segundo proceso: Utilizar otro ProcessBuilder para ejecutar findstr /i "d".

4. Redirigir la entrada estándar: Conectar la salida del primer proceso con la entrada del segundo usando getOutputStream().

5. Sincronizar ambos procesos: Usar waitFor() para asegurarse de que el segundo proceso no lea hasta que el primero haya terminado

Clase Ejercicio1comunicacion:

La clase Ejercicio1comunicacion es la clase que contiene el proceso padre (main) en la que desarrollaremos dos subprocesos hijos

Atributos: Ninguno

Explicación:

1. Primero utilizaremos el processBuilder para hacer la "Estructura" de los 2 subprocesos a iniciar.

   1.2 En el primero subproceso indicamos que interactué con la cmd, desde donde "/c" el comando a realizar (dir)

```
// estructuramos los procesos//
ProcessBuilder hijo1 = new ProcessBuilder("cmd","/c","dir");
```

   1.3 La "Estructura" del subproceso 2 a iniciar, en el que indicamos que interactué con la cmd, desde donde "/c" el comando a realizar (findstr) y que busqué los que contengan d

```
ProcessBuilder hijo2= new ProcessBuilder("cmd","/c","findstr", "/i", "d");
```

2. Iniciamos el primer subproceso y dejamos que haga lo suyo

```
//empezamos el primero y dejamos que haga lo suyo//
Process process1=hijo1.start();
```

3. Utilizamos un InputStreamReader (ya que son caracteres lo que vamos a manejar) y metemos en este los datos de salida del primer subproceso Y los almacenamos en un bufferReader

```
//abrimos canal por parte del proceso1//
InputStreamReader nuevo1= new InputStreamReader(process1.getInputStream());
OutputStream o1 = process1.getOutputStream();
//guardamos los datos del 1//
BufferedReader br= new BufferedReader(nuevo1);
```

4. Empezamos el subproceso 2 y inicializamos un nuevo outputStream del proceso 2 para abrir un canal de comunicación entre el subproceso 2 y 1

```
//emepzamos el segundo proceso//
Process process2=hijo2.start();


OutputStream o2= process2.getOutputStream();
```

5. Para hacer el transporte de los datos por el canal usamos un Bufferedwriter que tendrá como argumento un nuevo OutputStreamWriter ya que lo usaremos para sacar los caracteres

```
//abrimos canal por parte del 2 con el 1 y writer porque son caracteres//
BufferedWriter bw= new BufferedWriter(new OutputStreamWriter(o2));
```

6. Con una nueva variable String abrimos un bucle en el que leemos lo del BufferedReader del subproceso 1, se lo copiamos al String que acabamos de hacer y a la vez lo pasamos al BufferedWriter del subproceso 2 para que la salida del subproceso ahora contenga los datos del subproceso 1 mediante el uso del metodo write siempre y cuando el BufferedReader no haya terminado aka sea null

```
String linea;
while((linea=br.readLine())!=null) {
    //leemos y escribimos//
    bw.write(linea);
    bw.newLine();//separador de lineas//


}
```

7. Después del bucle cerramos los BufferedWriter y reader y hacemos un waitFor del subproceso 1 que podría haber terminado o no ya que la siguiente parte es indispensable que haya terminado

```
//cerramos todo para que n
br.close();
bw.close();

//antes de continuar hacem
process1.waitFor();
```

8. Repetimos lo que hemos hecho antes, pero para el subproceso 2, en un BufferedReader metemos los datos de salida del subproceso 2 mediante un nuevo InputStreamReader que será igual a la salida de datos del subproceso 2

```
BufferedReader br2= new BufferedReader(new InputStreamReader(process2.getInputStream()));
```

9. Con otra variable String leemos lo del BufferedReader almacenándolo en el String y ese ahora imprimiéndolo por consola con un system.out y otro que hará de salto de linea, hasta que el contenido del buffer se termine.

```
String linea2;
while((linea2=br2.readLine())!=null) {
        System.out.println(linea2);//imprimimos y salto de linea//
System.out.println();
}
```

10. Usamos un waitFor para comprobar que el segundo proceso ha terminado antes de cerrar el BufferedReader

```
process2.waitFor();
br2.close();
```

Imagen del código completo:

```java
import java.io.*;

public class Ejer1comunicacion {

    public static void main(String[] args) throws IOException, InterruptedException {
        // estructuramos los procesos//
        ProcessBuilder hijo1 = new ProcessBuilder("cmd","/c","dir");
        ProcessBuilder hijo2= new ProcessBuilder("cmd","/c","findstr", "/i", "d");
        //empezamos el primero y dejamos que haga lo suyo//
        Process process1=hijo1.start();

        //abrimos canal por parte del proceso1//
        InputStreamReader nuevo1= new InputStreamReader(process1.getInputStream());
        OutputStream o1 = process1.getOutputStream();
        //guardamos los datos del 1//
        BufferedReader br= new BufferedReader(nuevo1);

        //emepzamos el segundo proceso//
        Process process2=hijo2.start();


        OutputStream o2= process2.getOutputStream();
        //abrimos canal por parte del 2 con el 1 y writer porque son caracteres//
        BufferedWriter bw= new BufferedWriter(new OutputStreamWriter(o2));
        String linea;
        while((linea=br.readLine())!=null) {
            //leemos y escribimos//
            bw.write(linea);
            bw.newLine();//separador de lineas//


        }
        //cerramos todo para que no nos de errores//
        br.close();
        bw.close();

        //antes de continuar hacemos una comprovacion de que el prcess 1 ha terminado lo hacemos a
        process1.waitFor();

        BufferedReader br2= new BufferedReader(new InputStreamReader(process2.getInputStream()));/
        String linea2;
        while((linea2=br2.readLine())!=null) {
            System.out.println(linea2);//imprimimos y salto de linea//
        System.out.println();
        }
        process2.waitFor();//antes de cerrar el buffer comprovamos que el process2 haya terminado/
        br2.close();

    }

}
```

Casos prueba:

1. Probamos a buscar los archivos que contengan d

```
 El volumen de la unidad C no tiene etiqueta.

 El n@mero de serie del volumen es: 3A07-81E6

 Directorio de C:\Users\MariaChaparroCaballe\eclipse-workspace\clase2_10_2024

02/10/2024  09:50    <DIR>          .
02/10/2024  09:50    <DIR>          ..
02/10/2024  09:50    <DIR>          .settings
10/10/2024  08:42    <DIR>          bin
09/10/2024  09:18    <DIR>          src
               5 dirs  1.883.478.441.984 bytes libres
```

2. Probamos a buscar los archivos que contiene p

(La modificación del código)

```
ProcessBuilder hijo1 = new ProcessBuilder("cmd","/c","dir");
ProcessBuilder hijo2= new ProcessBuilder("cmd","/c","findstr", "/i", "p");
//empezamos el primero y dejamos que haga lo suyo//
Process process1=hijo1.start();
```

Resultado:

```
 Directorio de C:\Users\MariaChaparroCaballe\eclipse-workspace\clase2_10_2024

02/10/2024  09:50               393 .classpath
02/10/2024  09:50               390 .project
```

2 comunicación Bidireccional:

Enunciado: Implementar una clase Java que permita la comunicación bidireccional entre un proceso padre y un proceso hijo. El proceso padre enviará texto al proceso hijo, el hijo lo convertirá a mayúsculas y lo devolverá al padre, que lo mostrará en pantalla. Descripción:

1. Clase a implementar: Crear una clase llamada Mayusculas que: o Cree un proceso hijo en Java. o El proceso padre lea líneas de texto desde su entrada estándar (teclado) y las envíe al proceso hijo utilizando la entrada estándar del proceso hijo. o El proceso hijo reciba el texto desde su entrada estándar, lo convierta a mayúsculas, y lo envíe de vuelta al proceso padre a través de su salida estándar. o El proceso padre lea el texto convertido a mayúsculas desde la salida estándar del proceso hijo y lo muestre en su propia salida estándar (pantalla)

Clase Mayuscula:

Es la clase que contiene el proceso padre (main) que llamara a la clase hijo cuando se ejecute y ejecutara el subproceso hijo. En otras palabras, nuestro programa principal

Atributos: ninguno.

Explicación:

- **1.**Abrimos el try catch y usamos System.getProperty("user.dir") para obtener la ruta del directorio de trabajo actual de la aplicación y concatenamos la ruta con el directorio bin.

```
try {
String ruta = System.getProperty("user.dir");
String classpath = ruta + "/bin";
```

2. Hacemos la estructura del subproceso hijo y lo ejecutamos

```
ProcessBuilder hijo= new ProcessBuilder("java","-cp", classpath, "Ejer2Bidireccional.Hijo");
Process processhijo = hijo.start();
```

3. Pedimos un mensaje por la entrada estándar (teclado) y la almacenamos en un bufferedReader que tiene como argumento el inputstream porque son caracteres con lo que nos vamos a mover, y este lo obtiene del teclado

```
System.out.println("Escriba un mensaje, si quiere salir escriba salir");
BufferedReader br= new BufferedReader(new InputStreamReader(System.in));//main//
```

4. Empezamos con los canales de comunicación, para ello usaremos un BufferedWriter con argumentos de un nuevo OutputStreamWriter (tiene que ser writer porque nos movemos con caracteres) y este así mismo tiene como argumento que sea el outputstream del subproceso hijo que, aunque aún está vacío porque no hay datos ya inicializamos el canal que usaremos para sacar los datos. También hacemos un nuevo bufferedreader con el outputstream del subproceso hijo esto es para que nos meta los datos de salida del subproceso en el buffer en otras palabras es la vía de cómo van a volver los datos de nuevo al padre

```
BufferedWriter bw= new BufferedWriter(new OutputStreamWriter(processhijo.getOutputStream()));//co
BufferedReader brfinal= new BufferedReader(new InputStreamReader(processhijo.getInputStream()));/
```

5. Creamos dos variables String, una es “linea” esta es para leer el bufferedreader “br” que es el que guardaba lo que metía el usuario por teclado. La otra “respuesta” es donde almacenaremos la linea que devuelve el proceso de la clase hijo, pero de momento la dejamos inicializada, pero en blanco

```
String linea;
String respuesta="";
```

6.Creamos un bucle, en el que igualamos el “String linea” a la cadena que lea el bufferReader por teclado y mientras que este no sea null y lo contenido en “linea” no sea “salir” nos traspasara lo contenido en “linea” al bufferwriter “bw” mediante el uso de la función “.write()”, añadimos un

flush para forzar el envío y limpiar. Después metemos el contenido de “brfinal” a respuesta, imprimimos respuesta para mmostrar la cadena original ya transformada en el proceso de la clase mayusculas

```
while((linea=br.readLine()) !=null  && !linea.equalsIgnoreCase("salir")) {
    bw.write(linea);
    bw.flush();
    respuesta= brfinal.readLine();
    System.out.println(respuesta);
}
```

7.cerramos los dos primeros buffers (br, bw), nos aseguramos de que el subproceso hijo haya terminado y cerramos el “brfinal” por último cerramos el try y en caso de excepción el catch nos imprime todo el historial de errores

```
        bw.close();
        br.close();
        processhijo.waitFor();
        brfinal.close();
        }catch( Exception a) {
            a.printStackTrace();
        }
    }

}
```

Código completo:

```java
package Ejer2Bidireccional;

import java.io.*;

public class Mayusculas {

    public static void main(String[] args) throws IOException, InterruptedException {
        // TODO Auto-generated method stub
        try {
        String ruta = System.getProperty("user.dir");
        String classpath = ruta + "/bin";

        ProcessBuilder hijo= new ProcessBuilder("java","-cp", classpath, "Ejer2Bidireccional.Hijo");
        Process processhijo = hijo.start();

        System.out.println("Escriba un mensaje, si quiere salir escriba salir");
        BufferedReader br= new BufferedReader(new InputStreamReader(System.in));//main//

        BufferedWriter bw= new BufferedWriter(new OutputStreamWriter(processhijo.getOutputStream()));//c
        BufferedReader brfinal= new BufferedReader(new InputStreamReader(processhijo.getInputStream()));

        String linea;
        String respuesta="";

        while((linea=br.readLine()) !=null  && !linea.equalsIgnoreCase("salir")) {
            bw.write(linea);
            bw.flush();
            respuesta= brfinal.readLine();
            System.out.println(respuesta);
        }

        bw.close();
        br.close();
        processhijo.waitFor();
        brfinal.close();

        }catch( Exception a) {
            a.printStackTrace();
        }
    }

}
```

Clase Hijo:

Clase hijo es la clase que se llama cuando se ejecuta el subproceso, que ejecuta su proceso main, nuestras conexiones son con el main de la clase hijo y el main de la clase Mayusculas.

Atributos: ninguno.

Explicación:

1.Creamos un nuevo bufferedReader “brhijo” para meter los datos por la entrada estándar pero la del proceso main de Mayusculas

y un BufferedWriter de la salida estándar del proceso main de la clase Mayusculas, esta es nuestra forma de establecer comunicación con el proceso main de la clase padre aka la clase “Mayusculas” desde la clase hijo.

```
BufferedReader brhijo = new BufferedReader(new InputStreamReader(System.in));
BufferedWriter bwhijo = new BufferedWriter(new OutputStreamWriter(System.out));
```

2.Creamos una nueva variable String “lineaHijo” y la inicializamos pero la dejamos vacía, esta variable la usaremos para almacenar lo contenido en el bufferedreader anterior

```
String lineahijo="";
```

3.Creamos un bucle while donde metemos el contenido del bufferedreader en la variable “lineahijo” y mientras que no sea null aka haya terminado y no sea la palabra “salir” nos escribirá lo contenido en “lineahijo” en la salida por consola del proceso main de la clase Mayusculas por el uso del método write del bufferedwriter “bwhijo”.

Metemos una nueva línea para que nos empiece en otra línea y forzamoséll envió a la clase padre aka Mayusculas

```
while ((lineahijo=brhijo.readLine()) !=null  && !lineahijo.equalsIgnoreCase("salir")) {

    bwhijo.write(lineahijo.toUpperCase());
    bwhijo.newLine();
    bwhijo.flush();
}
```

4. Cerramos todos los bufferedeader y writer

```
brhijo.close();
bwhijo.close();
```

Foto de código completo:

```
package Ejer2Bidireccional;

import java.io.*;

public class Hijo {

    public static void main(String[] args) throws IOException {
        // TODO Auto-generated method stub
        BufferedReader brhijo = new BufferedReader(new InputStreamReader(System.in));
        BufferedWriter bwhijo = new BufferedWriter(new OutputStreamWriter(System.out));

        String lineahijo="";

        while ((lineahijo=brhijo.readLine()) !=null  && !lineahijo.equalsIgnoreCase("salir")) {

            bwhijo.write(lineahijo.toUpperCase());
            bwhijo.newLine();
            bwhijo.flush();
        }

        brhijo.close();
        bwhijo.close();
    }

}
```