

## Ejercicio de Repaso: Comunicación entre procesos con ProcessBuilder

### Objetivo:

Comprender cómo se comunican los procesos en Java utilizando las clases Process y ProcessBuilder, y cómo se emplean los flujos (InputStream, OutputStream, ErrorStream) para intercambiar información entre ellos.

### Criterios de evaluación R1:

- a) Se han reconocido las características de la programación concurrente y sus ámbitos de aplicación.
- b) Se han identificado las diferencias entre programación paralela y programación distribuida, sus ventajas e inconvenientes.
- c) Se han analizado las características de los procesos y de su ejecución por el sistema operativo.
- d) Se han caracterizado los hilos de ejecución y descrito su relación con los procesos.
- e) Se han utilizado clases para programar aplicaciones que crean subprocesos.
- f) Se han utilizado mecanismos para compartir información con los subprocesos iniciados.
- g) Se han utilizado mecanismos para sincronizar y obtener el valor devuelto por los subprocesos iniciados.
- h) Se han desarrollado aplicaciones que gestionen y utilicen procesos para la ejecución de varias tareas en paralelo.
- i) Se han depurado y documentado las aplicaciones desarrolladas.

### Enunciado:

Se desea crear un pequeño sistema de comunicación entre dos programas Java:

- **Padre.java** → crea un proceso hijo utilizando la clase ProcessBuilder.
- **Hijo.java** → espera recibir una línea de texto desde su entrada estándar, la procesa y devuelve una respuesta.

### Programa Hijo.java

- Debe leer una línea de texto desde la entrada estándar (System.in).
- Debe mostrar en su salida estándar (System.out) el mensaje recibido en mayúsculas y precedido por el texto: 'Mensaje recibido: '.
- Si ocurre algún error al leer la entrada, debe escribir en la salida de error (System.err) el mensaje 'Error al leer los datos'.

**Pista:** Utiliza BufferedReader y InputStreamReader(System.in).

### Programa Padre.java

- Debe crear un proceso hijo ejecutando el programa Hijo.java usando la clase `ProcessBuilder`.
- Debe enviarle al hijo, a través del `OutputStream` del proceso, la cadena 'Hola desde el proceso padre'.
- Debe leer y mostrar en pantalla la respuesta que el proceso hijo devuelva por su `InputStream`.
- En caso de error, deberá capturar el flujo de error (`getErrorStream()`) y mostrar los mensajes de error que el hijo genere.

**Pista:** Usa `write()` y `flush()` para enviar datos al hijo, y `BufferedReader` para leer la respuesta.

### Ejemplo de ejecución esperada:

Salida del programa Padre.java:

```
> java Padre
El proceso hijo ha respondido:
Mensaje recibido: HOLA DESDE EL PROCESO PADRE
Código de salida del hijo: 0
```

### Requisitos técnicos:

- Utiliza las clases del paquete `java.io` para la lectura/escritura de flujos.
- El programa padre debe usar `waitFor()` para esperar la finalización del hijo.
- Añade control de excepciones (`try-catch`) para `IOException` e `InterruptedException`.
- Comenta el código explicando el uso de cada flujo (`stdin`, `stdout`, `stderr`).

### Opcional (nivel avanzado):

- Modifica el programa Padre.java para que:
  - Envíe varias líneas de texto al hijo.
  - El hijo responda con la longitud de cada línea recibida.
  - El padre guarde todas las respuestas en un fichero `salida.txt` usando `PrintWriter`.

### Entrega :

Código del programa Hijo.java:

Código del programa Padre.java:

Ejecución del programa en el equipo: