Recordatorio de las clases ProcessBuilder y Process

Clase java.lang.ProcessBuilder

https://docs.oracle.com/javase/7/docs/api/java/lang/ProcessBuilder.html

Método	Cometido
ProcessBuilder command (String args)	Define el programa que se quiere ejecutar indicando sus argumentos como una lista de cadenas separada por comas.
List <String> command()	Devuelve todos los argumentos del objeto ProcessBuilder.
Map <String, String> enviroment()	Devuelve en una estructura Map las variables de entorno del objeto ProcessBuilder.
ProcessBuilder redirectError (File file)	Redirige la salida de error estándar a un fichero.
ProcessBuilder redirectInput (File file)	Establece un fichero como fuente en entrada estándar.
ProcessBuilder redirectOutput (File file)	Redirige la salida estándar a un fichero.
File directory()	Devuelve el directorio de trabajo del objeto ProcessBuilder.
ProcessBuilder directory (File directorio)	Establece el directorio de trabajo del objeto ProcessBuilder.
Process start()	Inicia un nuevo proceso usando los atributos del objeto ProcessBuilder.

Clase java.lang.Process

https://docs.oracle.com/javase/7/docs/api/java/lang/Process.html

Método	Cometido
InputStream getInputStream ()	Devuelve el flujo de entrada conectado a la salida normal del subproceso. Nos permite leer el stream de salida del subproceso, es decir, podemos leer lo que el comando que ejecutamos escribió en la consola.
int waitFor()	Provoca que el proceso actual espere hasta que el subproceso representado por el objeto *Process* finalice. Devuelve 0 si ha finalizado correctamente.
InputStream getErrorStream ()	Devuelve el flujo de entrada conectado a la salida de error del subproceso. Nos permite leer los posibles errores que se produzcan al lanzar el subproceso.

OutputStream getOutputStream ()	Devuelve el flujo de salida conectado a la entrada normal del subproceso. Nos permite escribir en el stream de salida del subproceso, así podemos enviar datos al subproceso que se ejecute.
void destroy()	Elimina el subproceso.
int exitValue()	Devuelve el valor de salida del subproceso.
boolean isAlive()	Devuelve true si el subproceso representado por *Process* está vivo.

Poner en marcha procesos

```
Process pb = new ProcessBuilder ("CALC"); Process p= pb.start();

Process p = new ProcessBuilder("CMD", "/C", "DIR").start();
```

Comunicación con un proceso

```
pb.redirectOutput (new File("output.txt"));
pb.redirectError (new File("error.txt"));
pb.redirectInput ();
```

Directorio de trabajo

```
pb.directory ("/bin")  \\Directorio de trabajo
```

Variables de entorno y comandos

```
Map entorno = pb.environment ();

List li = pb.command ();
Iterator iter = li.iterator();
while (iter.hasNext())
     System.out.println(iter.next());
```

Capturar la salida de un proceso

```
InputStream is = process.getInputStream();    // Captura la SALIDA
del proceso
int c;
while ((c = is.read()) != -1)    //Lee caracter a caracter
    System.out.print((char) c);
is.close();
```

Enviar datos a un proceso

```
OutputStream os = p.getOutputStream();
os.write("01-12-22".getBytes());
os.flush(); // vacía el buffer de salida
```

Capturar el mensaje de error de un proceso

```
InputStream is = p.getErrorStream();  // Captura el ERROR de
ejecución del proceso
BufferedReader br = new BufferedReader(new
InputStreamReader(is));  //Lee línea a línea
String line;
while ((liner = br.readLine()) != null)
        System.out.println("ERROR >" + line);
```

Capturar el código de error de un proceso

```
try {
  int exitValue = process.waitFor();
  System.out.println("\nCódigo de salida: "+ exitValue);
} catch (InterruptedException e) {
  e.printStackTrace(System.err);
}
```

Redirige E/S/Err a fichero

```
pb.redirectInput(fBat);
pb.redirectOutput(fOut);
pb.redirectError(fErr);
pb.start();
```

Redirige la salida a consola

```
pb.redirectOutput(ProcessBuilder.Redirect.INHERIT);
Process p = pb.start();
```