**Programación de servicios y procesos**

**Tarea para PSP01.**

**Ejercicio 1**

* Primera parte: implementa una aplicación que ordena un conjunto indeterminado de números que recibe a través de su entrada estándar; y muestra el resultado de la ordenación en su salida estándar. La aplicación se llamará 'ordenarNumeros'.
* Segunda parte: implementa una aplicación, llamada 'aleatorios', que genere al menos 40 números aleatorios (entre 0 y 100), y que los escriba en su salida estándar.

Para la primera parte de este ejercicio, aprovechamos el consejo dado por el tutor reaprovechando código del ejemplo 5.1.

He intentado comentar el código lo mejor posible para no tener que añadir desde aquí muchas más anotaciones:

/\*

\* TAREA PARA PSP01 EJERCICIO 1.

\* PRIMERA PARTE:

\* Implementa una aplicación que ordena un conjunto indeterminado de números que recibe a través de su entrada estándar;

\* Muestra el resultado de la ordenación en su salida estándar.

\* La aplicación se llamará 'ordenarNumeros'.

\*/

package ordenarnumeros**;**

**import** java**.**io**.**BufferedReader**;**

**import** java**.**io**.**IOException**;**

**import** java**.**io**.**InputStreamReader**;**

**import** java**.**util**.**ArrayList**;**

**import** java**.**util**.**Collections**;**

**import** java**.**util**.**Random**;**

/\*\*

\*

\* **@author** juang <juangmuelas@gmail.com>

\* **@since** 22/10/2020

\*/

public class OrdenarNumeros **{**

/\*\*

\* **@param** args the command line arguments

\*/

public static void main**(**String**[]** args**)** **{**

// Parte del código lo aprovechamos desde el recurso facilitado de la plataforma

// Vamos a leer de la entrada estándar del proceso y escribir

// los datos que se reciben en la salida estándar del proceso.

InputStreamReader isr **=** **new** InputStreamReader**(**System**.**in**);**

BufferedReader bf **=** **new** BufferedReader **(**isr**);**

// Obtenemos el stream de lectura de la entrada estándar

// utilizamos un lector con Buffered, para no perder ningún dato

String lineaTeclado **=** **null;**

**try{**

System**.**out**.**println**(**"Proceso lector"**);**

System**.**out**.**println**(**"Pulsar intro para añadir números"**);**

ArrayList**<**Integer**>** listaNumeros **=** **new** ArrayList **<>();**

//Mostramos que el proceso que está escribiendo es el que está

//leyendo los datos.

**while** **((**lineaTeclado **=** bf**.**readLine**())!=** **null){**

//Para facilitar la recogida de datos, la automatizamos con un

//objeto Random sin restricciones de rango.

Random aleatorio **=new** Random**();**

int numeros **=** aleatorio**.**nextInt**();**

// System.out.println("Listando " + numeros + " como elemento aleatorio");

System**.**out**.**println**(**"Si lo desesa, vuelva a pulsar intro para añadir números"**);**

//Añadir, ordenar y mostrar.

listaNumeros**.**add**(**numeros**);**

Collections**.**sort**(**listaNumeros**);**

System**.**out**.**println**(**listaNumeros**);**

**}**

**}catch(**IOException ex**){**

System**.**err**.**println**(**"Se ha producido un error de E/S."**);**

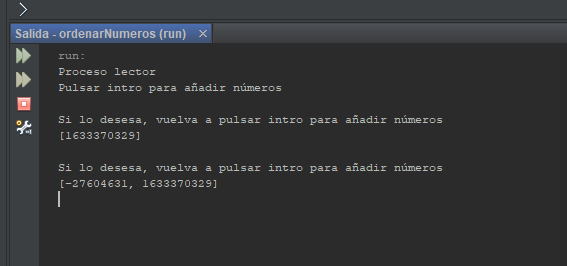
System**.**err**.**println**(**ex**.**toString**());**

**}**

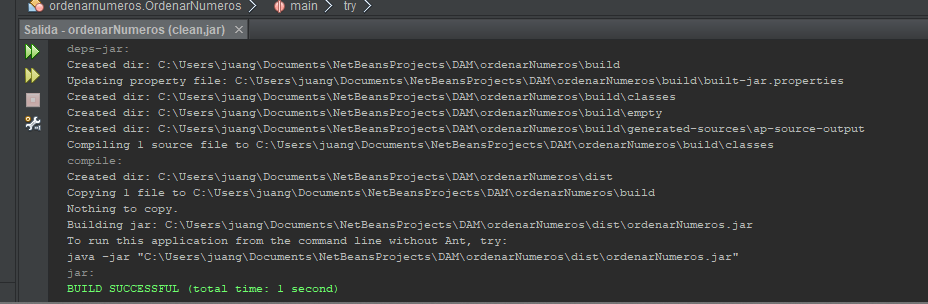
**}** //Fin main

**}** // Fin clase

Comprobar que funciona correctamente:



Generamos el archivo .jar con may+F11



En la segunda parte también añado comentarios para no extender mucho aquí y quede documentado en el archivo.

/\*

\* TAREA PARA PSP01 EJERCICIO 1.

\* SEGUNDA PARTE:

\* Implementa una aplicación llamada 'aleatorios', que genere al menos 40 números

\* aleatorios (entre 0 y 100), y que los escriba en su salida estándar.

\*/

package aleatorios**;**

**import** java**.**util**.**Random**;**

/\*\*

\*

\* **@author** juang <juangmuelas@gmail.com>

\* **@since** 28/10/2020

\*/

public class Aleatorios **{**

/\*\*

\* **@param** args the command line arguments

\*/

public static void main**(**String**[]** args**)** **{**

/\*\*

\* Necesitamos facilitarle limites.

\* **@param** limite cantidad de valores a retornar

\*/

int limite **=** 0**;**

//Usamos Random para reoger enteros al azar.

Random a **=** **new** Random**();**

//Aseguramos que no tenga menos de 40 valores

**while** **(**limite**<**40**){**

/\*\*

\* El ejercicio no da un límite superior, pero le ponemos uno para

\* no alargar mucho la ejecución

\* **@param** limite cantidad de valores a retornar

\*/

limite **=** a**.**nextInt**(**100**);**

**}**

// guardamos el dato

int**[]** aleatorios **=** **new** int**[**limite**];**

//Con el límite impuesto, procedemos a recoger los valores.

**for** **(**int i**=**0**;** i**<**aleatorios**.**length**;** i**++){**

//Volvemos a aprevechar Random para los valores.

Random numeros **=** **new** Random**();**

/\*\*

\* El ejercicio pide un valor máximo

\* **@param** valores valor limite de 100

\*/

int valores **=** numeros**.**nextInt**(**100**);**

aleatorios**[**i**]** **=** valores**;**

**}**

**for** **(**int v**=**0**;** v**<**aleatorios**.**length**;** v**++){**

//Recorremos y para mostrarlo mejor, añadimos un espacio entre los valores.

System**.**out**.**print**(**aleatorios**[**v**]** **+** " "**);**

**}**

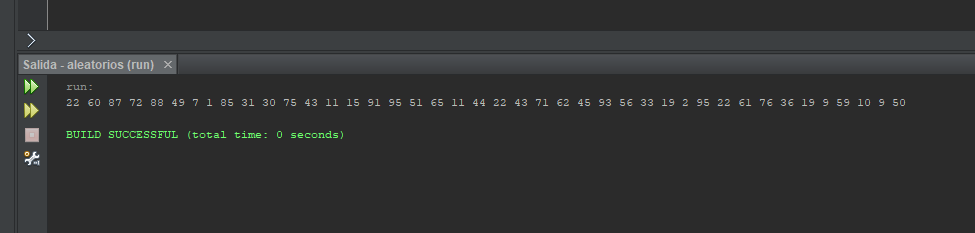
//Añado un salto de línea para mejorar un poco la visibilidad.

System**.**out**.**println**(**"\n"**);**

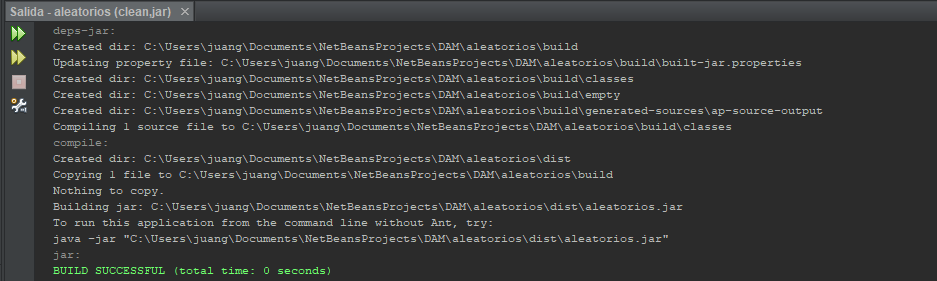
**}** // Fin main

**}** // Fin clase Aleatorios

Comprobar que funciona correctamente:

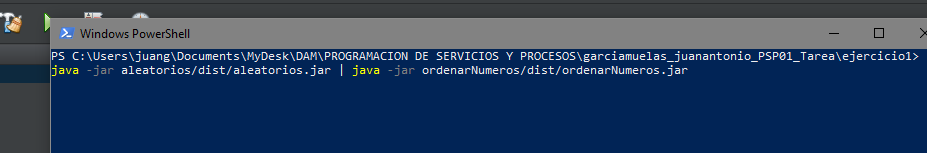


Generamos el archivo .jar con may+F11



Accedemos a la consola de windows para probar la comunicación entre ambos.

Para simplificar el ejemplo, accedemos a través del path del archivo .jar en vez de sacarlo de dist.



La ejecución no presenta errores.

