Qwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmrtyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasd

fghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmrtyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmrtyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmrtyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmrtyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmrtyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmrtyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnm

|  |
| --- |
| PRÁCTICA SGE02  Instalación y configuración de sistemas ERP-CRM.  20/10/2022  CÉSAR BOUZAS SOTO |

Contenido

[1 Descarga de Internet o del repositorio del sistema, el fichero de instalación de Odoo, en su versión más reciente. 2](#_Toc117202675)

[2 Realiza el proceso de instalación de la aplicación ERP. 3](#_Toc117202676)

[3 En los casos en que sea necesario, realiza la configuración del servidor, base de datos y servicios de acceso. 5](#_Toc117202677)

[4 Comprueba el correcto funcionamiento de las aplicaciones. 9](#_Toc117202678)

[5 Visualiza los módulos instalados en la aplicación. 10](#_Toc117202679)

# Ejercicio 1) Se compone de 3 partes:

## 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'.

Recibe uno por uno desde la entrada del sistema un serie de enteros , los ordena y los imprime entre corchetes y separados por comas.

[Código Ordenar números.](https://github.com/cesarbouzas/DAM/blob/main/MP0490_Programacion_servicios_y_procesos/001_unidad_1/001_practica/001_tuberia/OrdenarNumero/src/ordenarnumero/OrdenarNumero.java)

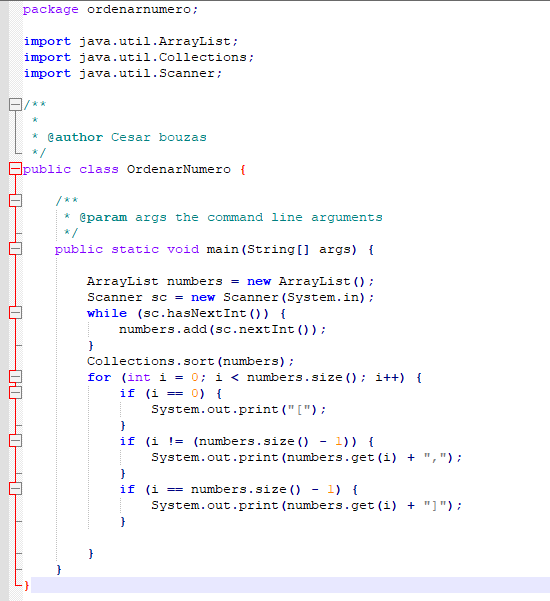


Ilustración Ordenar números

## Implementa una aplicación, llamada 'aleatorios', que genere al menos 40 números aleatorios (entre 0 y 100), y que los escriba en su salida estándar.

Genera40 números aleatorios entre 0 y 100.

[Código Aleatorio.](https://github.com/cesarbouzas/DAM/blob/main/MP0490_Programacion_servicios_y_procesos/001_unidad_1/001_practica/001_tuberia/Aleatorios/src/aleatorios/Aleatorios.java)

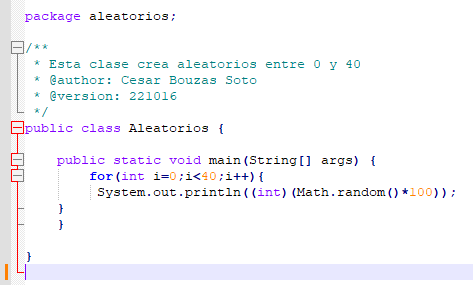


Ilustración Código Aleatorio

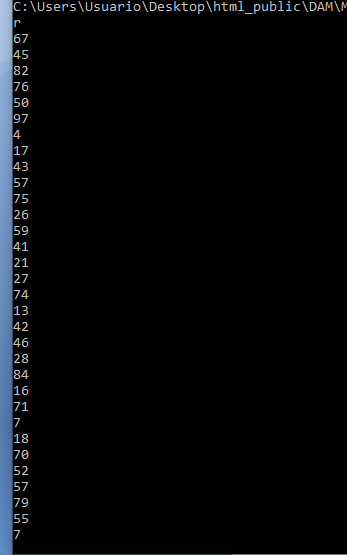


Ilustración salida de 40 números aleatorios.

## Realiza un pequeño manual (tipo "¿Cómo se hace?" o "HowTo"), utilizando un editor de textos (tipo word o writer) en el que indiques, con pequeñas explicaciones y capturas, cómo has probado la ejecución de las aplicaciones que has implementado en este ejercicio. Entre las pruebas que hayas realizado, debes incluir una prueba en la que utilizando el operador "|" (tubería) redirijas la salida de la aplicación 'aleatorios' a la entrada de la aplicación 'ordenarNumeros'.

### Copiamos los jar en una carpeta.

Mediante Clean and build del IDE Netbeans generamos los Jar de cada clase , los colocamos en la misma carpeta.

[Repositorio de la carpeta con los \*.jar.](https://github.com/cesarbouzas/DAM/tree/main/MP0490_Programacion_servicios_y_procesos/001_unidad_1/001_practica/001_tuberia)

### Entramos en la consola de sistema en la carpeta donde se encuentran los \*.jar.

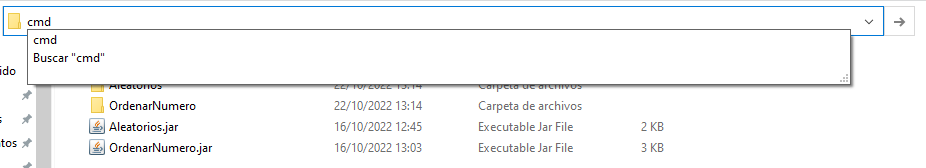


Ilustración cambiamos la dirección por cmd.

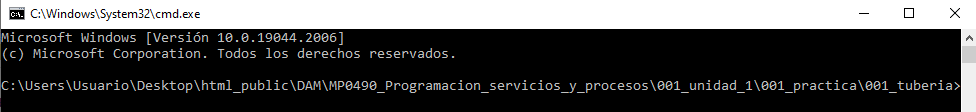


Ilustración Consola del sistema.

### Ejecución de java –jar | java -jar

Mediante la opción llamada tubería , los datos de salida de un procedimiento se introducen como entrada en el otro.

Si escribimos en línea de comandos java -jar Aleatorios.jar | java -jar OrdenarNumero.jar obtenemos 40 numeros aleatorios ordenados de menor a mayor.

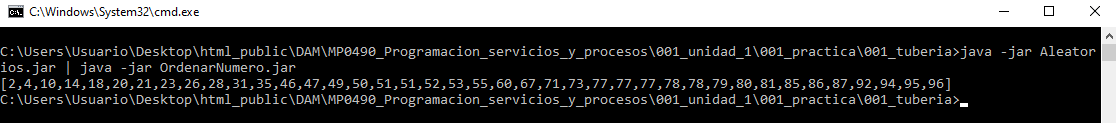


Ilustración Salida de la tubería Aleatroios|Ordenar.

# Ejercicio 2) Se compone de 3 partes:

## Implementa una aplicación que escriba en un fichero conjuntos de letras generadas de forma aleatoria. Escribiendo cada conjunto de letras en una línea distinta. El número de conjuntos de letras a generar por el proceso, también será dado por el usuario en el momento de su ejecución. Se Llamará Lenguaje.

[Repositorio de Lenguaje.class](https://github.com/cesarbouzas/DAM/blob/main/MP0490_Programacion_servicios_y_procesos/001_unidad_1/001_practica/002_colaborar/Lenguaje/src/vista/Lenguaje.java)

Genera y guarda en archivo ,un numero n de palabras de máximo 8 letras (alfabeticas) indicado por consola.

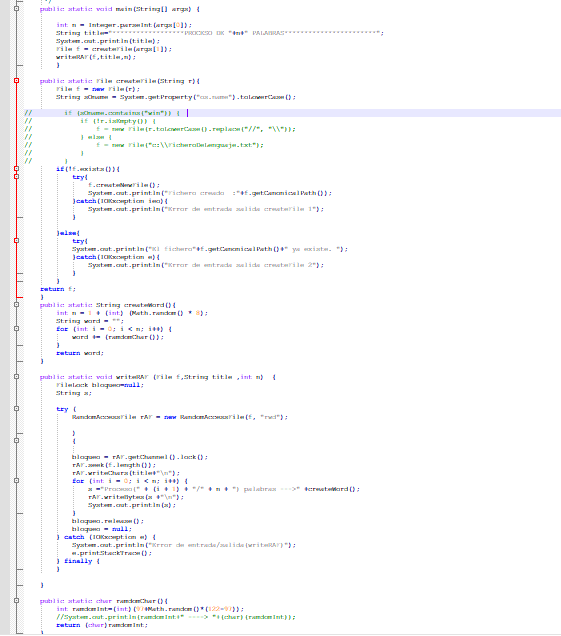


Ilustración Código lenguaje.class

[Ilustración 1 Ordenar números 3](#_Toc117340198)

[Ilustración 2 Código Aleatorio 4](#_Toc117340199)

[Ilustración 3 salida de 40 números aleatorios. 4](#_Toc117340200)

[Ilustración 4 cambiamos la dirección por cmd. 5](#_Toc117340201)

[Ilustración 5 Consola del sistema. 5](#_Toc117340202)

[Ilustración 6 Salida de la tubería Aleatroios|Ordenar. 5](#_Toc117340203)