Qwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmrtyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasd

fghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmrtyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmrtyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmrtyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmrtyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmrtyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmrtyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnm

|  |
| --- |
| PRÁCTICA PSP01  Programación multiproceso  22/10/2022  CÉSAR BOUZAS SOTO |

Contenido

[1 Ejercicio 1) Se compone de 3 partes: 2](#_Toc117351984)

[1.1 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'. 2](#_Toc117351985)

[1.2 Implementa una aplicación, llamada 'aleatorios', que genere al menos 40 números aleatorios (entre 0 y 100), y que los escriba en su salida estándar. 3](#_Toc117351986)

[1.3 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'. 4](#_Toc117351987)

[1.3.1 Copiamos los jar en una carpeta. 4](#_Toc117351988)

[1.3.2 Entramos en la consola de sistema en la carpeta donde se encuentran los \*.jar. 4](#_Toc117351989)

[1.3.3 Ejecución de java –jar | java -jar 4](#_Toc117351990)

[2 Ejercicio 2) Se compone de 3 partes: 5](#_Toc117351991)

[2.1 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. 5](#_Toc117351992)

[2.2 Segunda parte: implementa una aplicación, llamada 'colaborar', que lance al menos 10 instancias de la aplicación "lenguaje". Haciendo que todas ellas, colaboren en generar un gran fichero de palabras. Cada instancia generará un número creciente de palabras de 10, 20, 30, … Por supuesto, cada proceso seguirá escribiendo su palabra en una línea independiente de las otras. Es decir, si lanzamos 10 instancias de "lenguaje", al final, debemos tener en el fichero 10 + (Sumar.) 20 + 30 + … + 100 = (Igual.) 550 líneas. 6](#_Toc117351993)

[2.3 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. 7](#_Toc117351994)

[Reposito 1 Tubería](https://github.com/cesarbouzas/DAM/tree/main/MP0490_Programacion_servicios_y_procesos/001_unidad_1/001_practica/001_tuberia)

[Repositorio2 Lenguaje](https://github.com/cesarbouzas/DAM/tree/main/MP0490_Programacion_servicios_y_procesos/001_unidad_1/001_practica/002_colaborar)

# 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)

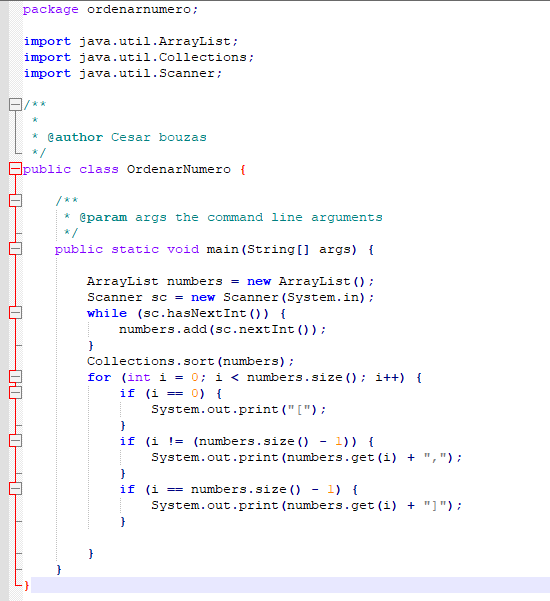


Ilustración Ordenar números

## 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)

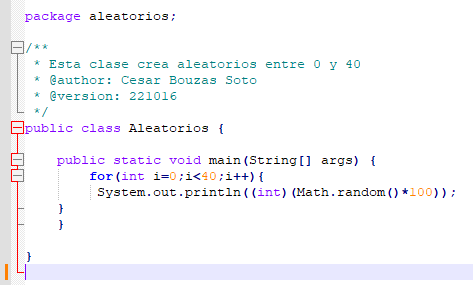


Ilustración Código Aleatorio

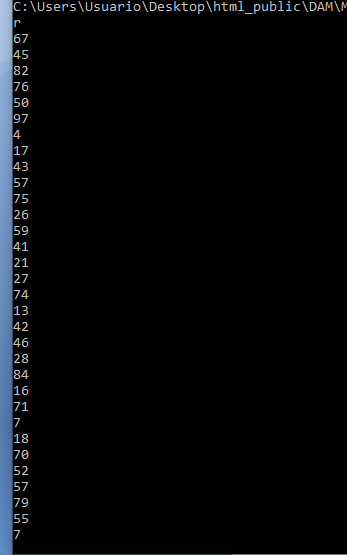


Ilustración salida de 40 números aleatorios.

## 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.

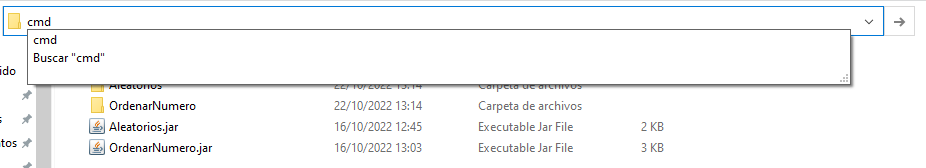


Ilustración cambiamos la dirección por cmd.

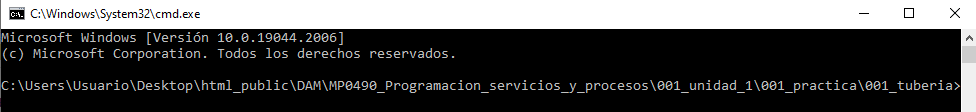


Ilustración Consola del sistema.

### 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 números aleatorios ordenados de menor a mayor.

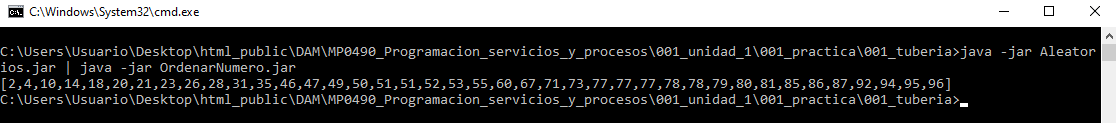


Ilustración Salida de la tubería Aleatorio |Ordenar.

# 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 (alfabéticas) indicado por consola.

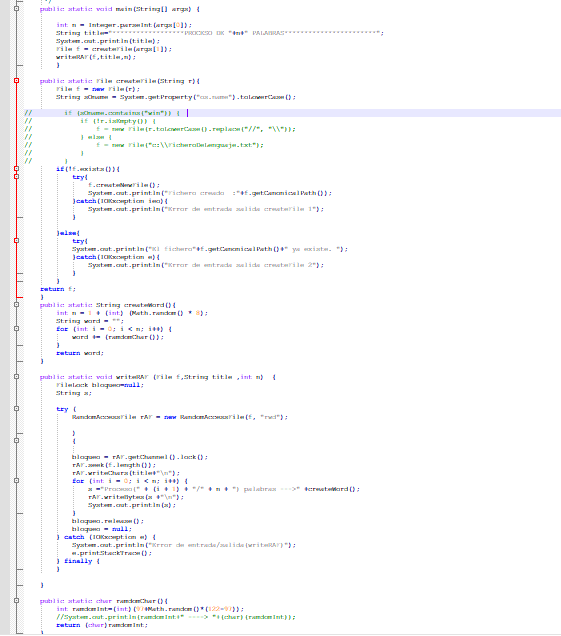


Ilustración Código lenguaje.class

## Segunda parte: implementa una aplicación, llamada 'colaborar', que lance al menos 10 instancias de la aplicación "lenguaje". Haciendo que todas ellas, colaboren en generar un gran fichero de palabras. Cada instancia generará un número creciente de palabras de 10, 20, 30, … Por supuesto, cada proceso seguirá escribiendo su palabra en una línea independiente de las otras. Es decir, si lanzamos 10 instancias de "lenguaje", al final, debemos tener en el fichero 10 + (Sumar.) 20 + 30 + … + 100 = (Igual.) 550 líneas.

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

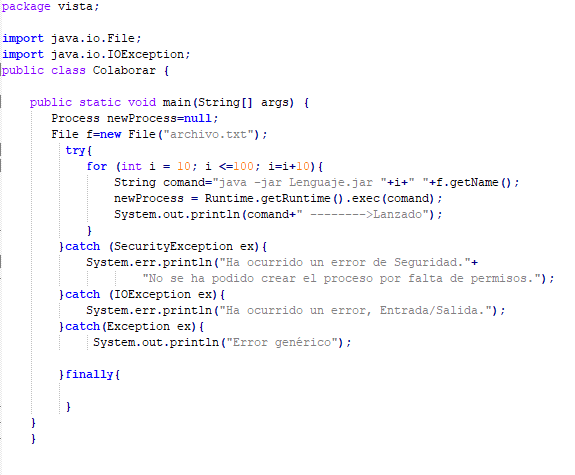


Ilustración Código de colaborar.class

## 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.

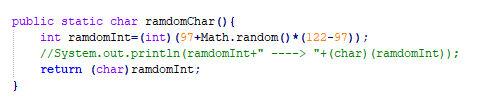
Por su lado lenguaje crea palabras compuestas por un numero aleatorio de caracteres de 1 a 8 , comprendidos dentro del código ascii comprendidos entre el 96->a y el 122->z. 

Ilustración generación de ascii aleatorio entre a-z

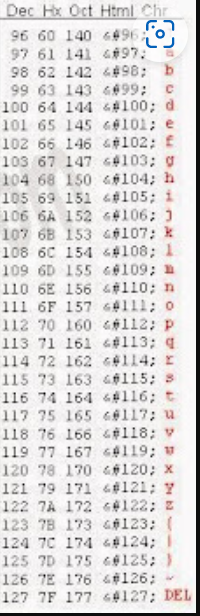


Ilustración tabla ascii

Cada uno de los procesos imprime un titulo donde n es el número de palabras a generar que recibe como primer argumento.



Ilustración Generación de titulo

Por si solo Lenguaje jar crea una cantidad de palabras que le pasamos como primer parámetro y como segundo el archivo donde las debe guardar: java -jar lenguaje.jar 10 archivo10.txt

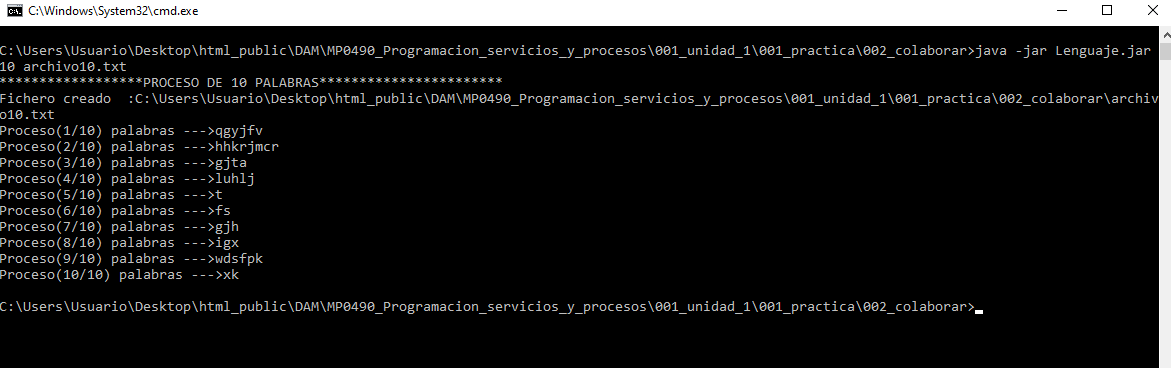


Ilustración java -jar lenguaje.jar 10 archivo10.txt

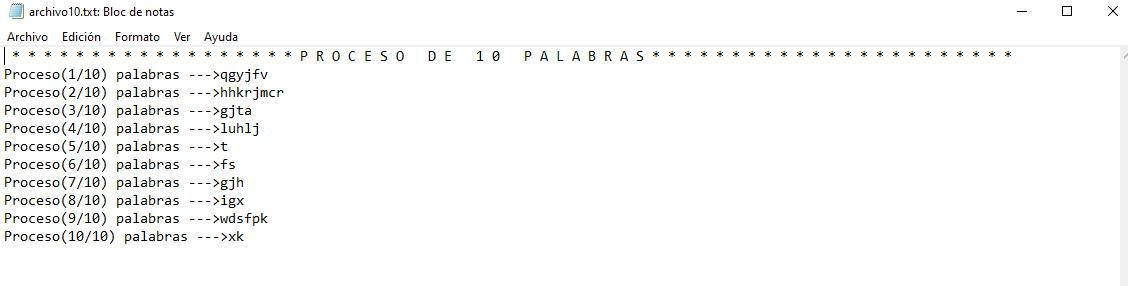


Ilustración contenido del archivo

Se indica con el titulo cual es el proceso que toma el control de la escritura , como proceso esta bloqueado debe empezar hasta terminar , no se garantiza el orden pues esto lo decide el gestor de procesos del SO , encargado de determinar el momento que cada proceso entra en el CPU.

La aplicación colaborar lo único que hace es llamar 10 veces a Lenguaje.jar introduciéndole como parámetro 10,20,,30…100 lo que hace que tengamos en el archivo.txt 550 líneas como además he decidido introducir un titulo debemos tener 550+10=560 líneas .

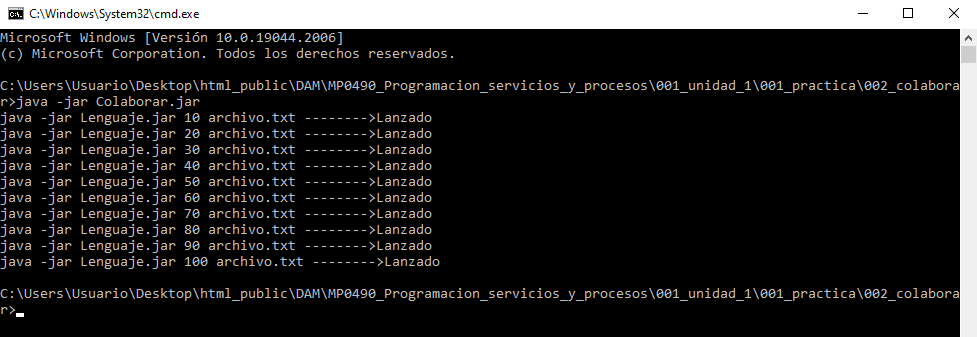


Ilustración Ejecución java -jar Colaborar.jar

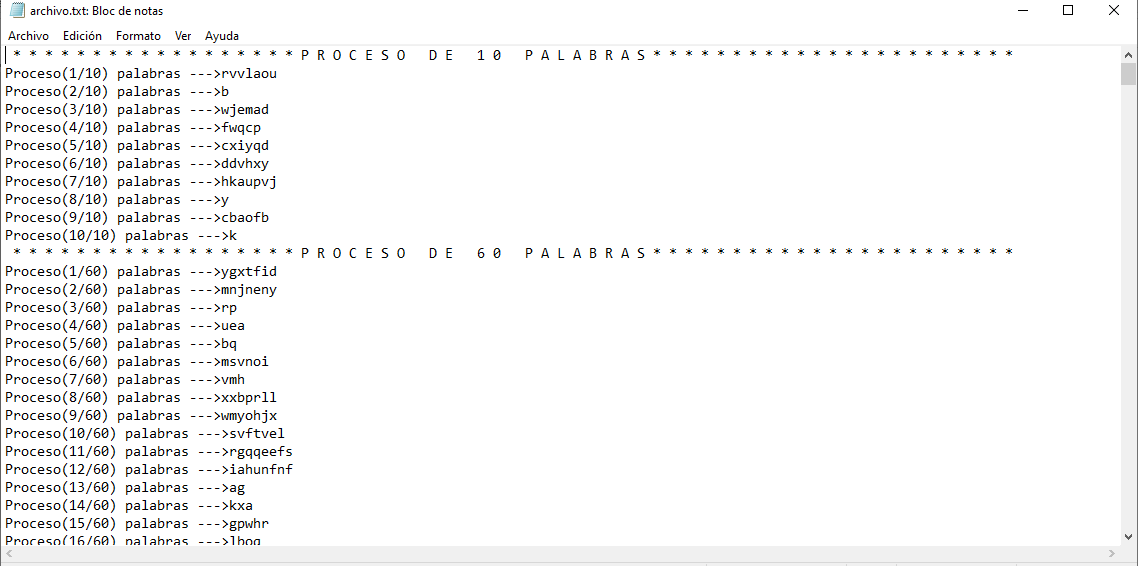


Ilustración contenido de archivo.txt

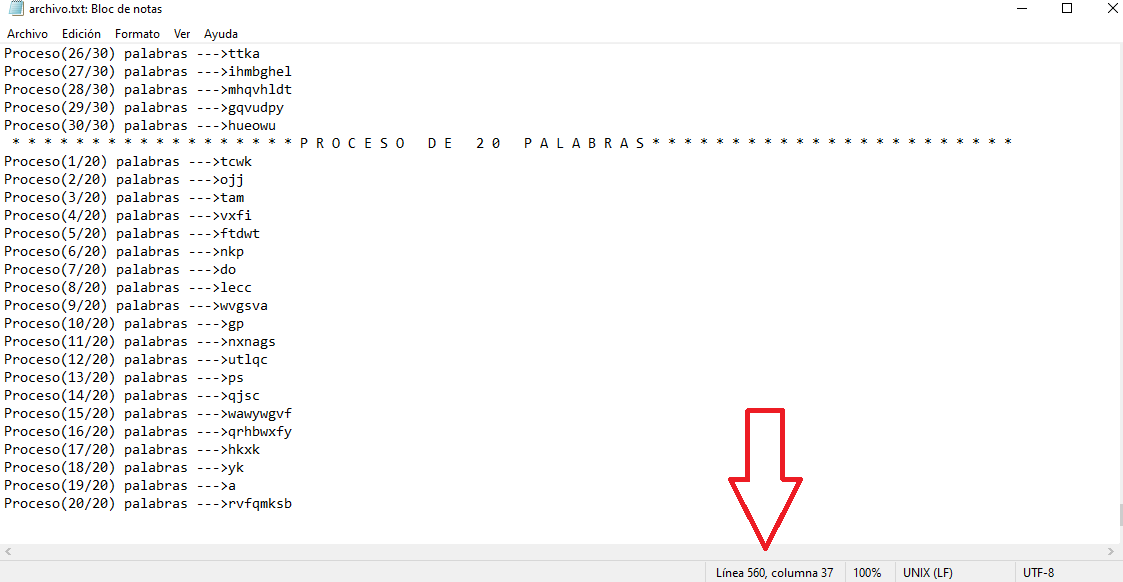


Ilustración Comprobación del total de líneas 550 palabras + 10 títulos

[Ilustración 1 Ordenar números 2](#_Toc117351661)

[Ilustración 2 Código Aleatorio 3](#_Toc117351662)

[Ilustración 3 salida de 40 números aleatorios. 3](#_Toc117351663)

[Ilustración 4 cambiamos la dirección por cmd. 4](#_Toc117351664)

[Ilustración 5 Consola del sistema. 4](#_Toc117351665)

[Ilustración 6 Salida de la tubería Aleatorio |Ordenar. 4](#_Toc117351666)

[Ilustración 7 Código lenguaje.class 5](#_Toc117351667)

[Ilustración 8 Código de colaborar.class 6](#_Toc117351668)

[Ilustración 9 generación de ascii aleatorio entre a-z 7](#_Toc117351669)

[Ilustración 10 tabla ascii 7](file:///C:\Users\Usuario\Desktop\html_public\DAM\MP0490_Programacion_servicios_y_procesos\001_unidad_1\001_practica\bouzas_soto_cesar_PSP01_Tarea.docx#_Toc117351670)

[Ilustración 11 Generación de titulo 8](#_Toc117351671)

[Ilustración 12 java -jar lenguaje.jar 10 archivo10.txt 8](#_Toc117351672)

[Ilustración 13 contenido del archivo 8](#_Toc117351673)

[Ilustración 14 Ejecución java -jar Colaborar.jar 8](#_Toc117351674)

[Ilustración 15 contenido de archivo.txt 9](#_Toc117351675)

[Ilustración 16 Comprobación del total de líneas 550 palabras + 10 títulos 9](#_Toc117351676)