Tarea PSP02

Yanet López Rodríguez
29/11/2023

Indice

Introducción

En esta actividad deberemos realizar las siguientes tres partes:

Primera parte: implementa una aplicación que escriba en un fichero indicado por el usuario conjuntos de letras generadas de forma aleatoria (sin sentido real). 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. Esta aplicación se llamará "lenguaje" y como ejemplo, podrá ser invocada así:
java -jar lenguaje 40 miFicheroDeLenguaje.txt

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 + 20 + 30 + … + 100 = 550 líneas.

Tercera parte: 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.

Primera parte: ***implementa una aplicación que escriba en un fichero indicado por el usuario conjuntos de letras generadas de forma aleatoria (sin sentido real). 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. Esta aplicación se llamará "lenguaje" y como ejemplo, podrá ser invocada así: java -jar lenguaje 40 miFicheroDeLenguaje.txt***

Para realizar esta actividad lo primero que haremos será crear un nuevo proyecto, en el tendremos nuestra clase Main y en ella tendremos main() y la función generarStringAleatorio().
Lo primero que haremos será recoger dos argumentos y guardarlos en variables, estos serán el número de conjuntos de letras que queremos y el nombre del archivo.
Ahora crearemos una nueva variable y la iniciaremos igualándola a null, esta será de tipo BufferedWrite y se llamará bufercito uwu.

Hacemos un try catch para controlar los errores que se puedan producir y dentro de este usaremos a bufercito para escribir en el documento, así que deberemos ponerlo guapo. Por último en nuestro try deberemos hacer un bucle for el cual se ejecutará la cantidad de veces que le indicamos al ejecutar el comando, este escribirá el String de letras aleatorias que le retornará la función generarStringAleatorio() y un salto de línea cada vez que se ejecute.
Ahora cerraremos el try con el catch, este mostrará por pantalla el mensaje de error si se produce alguna excepción.

Por último añadimos al try un finally donde comprobará con un if si se creó correctamente el archivo y si es así lo cerrara. El cierre lo meteremos en otro try catch para controlar los errores.

```
public class Main {
    public static void main(String[] args) {
        //RECOGEMOS EL NÚMERO DE CONJUNTOS DE LETRAS QUE DESEAMOS GENERAR
        int cantidad = Integer.parseInt(args[0]);

        //RECOGEMOS EL NOMBRE QUE LE DAREMOS AL ARCHIVO
        String archivo = args[1];

        //DECLARAMOS UNA NUEVA VARIABLE BufferedWriter Y LA INICIAMOS
        BufferedWriter bufercito = null;

        //USAMOS UN TRY CATCH PARA EL CONTROL DE ERRORES PARA QUE NO NOS EXPLOTE EL PC
        try {
            //PONEMOS WAPO A bufercito PARA ESCRIBIR EN EL ARCHIVO USANDO FileWriter
            //PONEMOS true PARA QUE AÑADA LAS LINEAS AL DOCUMENTO Y NO LO ELIMINE
            bufercito = new BufferedWriter(new FileWriter(archivo, true));

            //HACEMOS UN BUCLE FOR QUE SE EJECUTE LA CANTIDAD DE VECES QUE LE INDICAMOS
            for (int i = 0; i < cantidad; i++) {
                //ESCRIBIMOS EN EL DOCUMENTO EL CONJUNTO DE LETRAS PASANDOLE AL bufercito LA FUNCION generarStringAleatorio()
                //LE PASAMOS generarStringAleatorio() COMO PARAMETRO YA QUE ESTA RETORNARÁ EL STRING DE ALEATORIOS
                bufercito.write(generarStringAleatorio());
                //AGREGAMOS UNA NUEVA LINEA DE TEXTO
                bufercito.newLine();
            }

        //SI OCURRE UN ERROR DURANTE LA CREACIÓN DEL ONJETO BufferedWriter EL CATCH LO CAPTURA Y LO MUESTRA
        } catch (IOException e) {
            System.out.println("ERROR >:v " + e.getMessage());
        } finally {
            //VERIFICAMOS QUE EL bufercito NO SEA NULO, ESTO SIGNIFICA QUE SE CREO CORRECTAMENTE
            if (bufercito != null) {
                try {
                    //CERRAMOS EL BufferedWriter USANDO EL METODO close()
                    bufercito.close();
                    //SI OCURRE UN ERROR DURANTE EL CIERRE EL CATCH LO CAPTURA Y LO MUESTRA
                } catch (IOException e) {
                    System.out.println("ERROR AL CERRAR >:v " + e.getMessage());
                }
            }
        }
```

La función generarStringAleatorio() nos retornará un String de, en este caso 10 caracteres de letras aleatorias. Para esto, primero deberemos crear un String con todos los caracteres del abecedario, a este String le llamaremos caracteres, crearemos una nueva String llamada sandwich, la iniciaremos e instanciaremos la clase random.

Ahora haremos un bucle for que en este caso se ejecutará 10 veces haciendo que cada String tenga 10 caracteres. Dentro del bucle for, cada vez que se ejecute se añadirá una nueva letra a sandwich, esta letra sera la que este en la posición que indique el número aleatorio que se genere.

Por último retornamos el String Sandwich.

```
public static String generarStringAleatorio() {
    //HACEMOS UN STRING CON TODAS LAS LETRAS DEL ABECEDARIO EN MAYUSCULAS Y MINUSCULAS
    String caracteres = "ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz";

    //CREANMOS UNA VARIABLE STRING LLAMADA sandwich Y LA INSTANCIAMOS
    String sandwich = "";

    //INSTANCIAMOS LA CLASE RANDOM
    Random random = new Random();

    //HACEMOS UN BUCLE FOR QUE IRÁ INSERTANDOLE UNA NUEVA LETRA RANDOM AL STRING LAS
    //VECES QUE LE INDIQUEMOS ENH ESTE CASO 10
    for (int i = 0; i < 10; i++) {
        //COJEMOS LA LETRA QUE ESTA EN LA POSICION DEL NUMERO ALEATORIO Y SE LA AÑADIMOS A sandwich
        sandwich = sandwich + caracteres.charAt(random.nextInt(caracteres.length()));
    }

    //DEVOLOVEMOS EL CONJUNTO DE 10 LETRAS ALEATÓRIAS
    return sandwich;
}
```

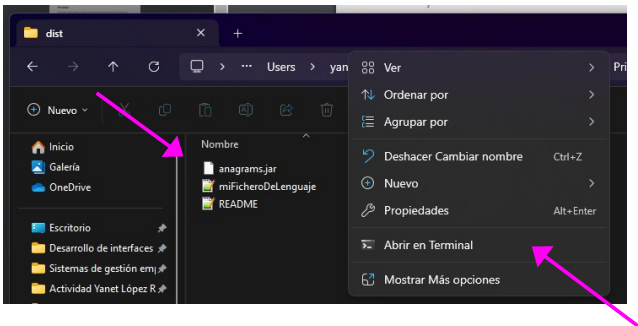
Por último haremos click derecho en nuestro proyecto y pulsaremos en “Build”, esto generará un archivo .jar en una carpeta llamada dir de nuestro proyecto, en mi caso la ruta de mi jar estará en: C:\Users\yanet\Documents\NetBeansProjects\PrimeraParteEjercicio2\dist

Pruebas

- Ejecución en modo aislado o secuencial.

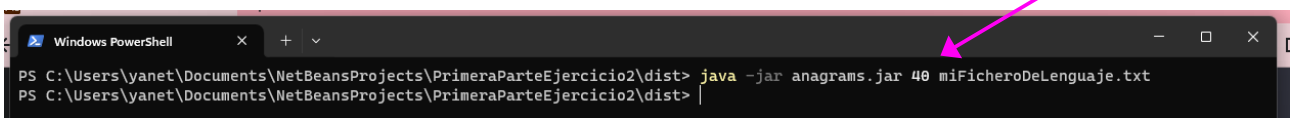
Para realizar la ejecución en modo aislado nos ubicaremos en la ruta de nuestro archivo .jar

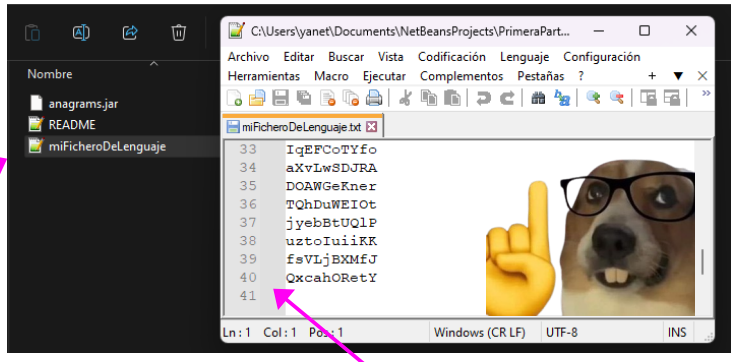
Aquí deberemos hacer click derecho y escoger la opción de “Abrir terminal”, esto nos abrirá el cmd ubicándonos en la ruta en la que estamos.



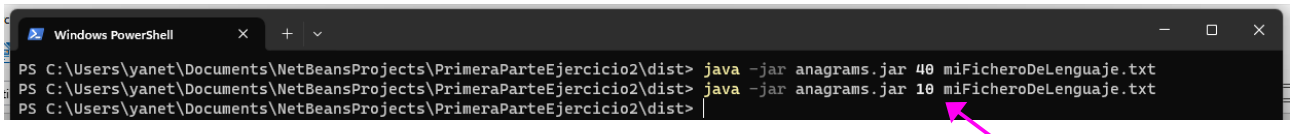
Ejecutamos en el terminal java java ***-jar lenguaje.jar 40 miFicheroDeLenguaje.txt*** y comprobamos que nuestro código funciona como queremos, haremos varias pruebas con distintos argumentos para probar que no hay fallos en nuestra aplicación.

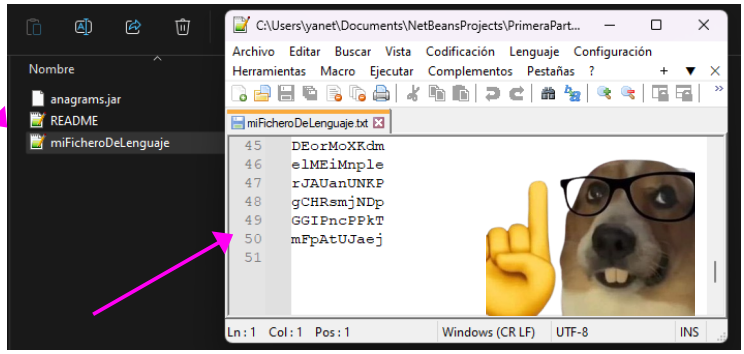
Aquí vemos como nuestro .jar crea correctamente un documento llamado miFicheroDeLenguaje.txt y le añade las 40 lineas que le indicamos.





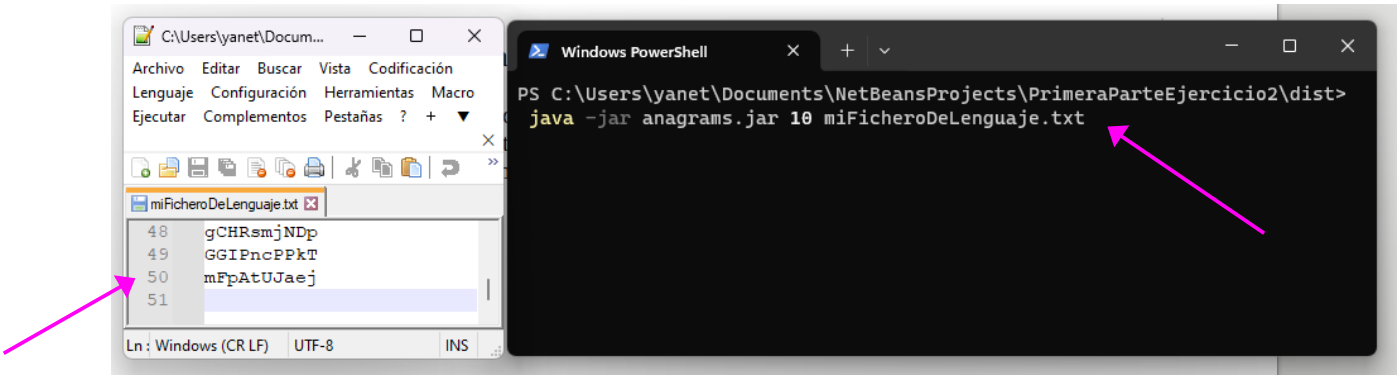
Aquí vemos como nuestro .jar añade correctamente las nuevas 10 lineas a las antiguas 40 lineas insertadas antes correctamente en el archivo miFicheroDeLenguaje.txt.



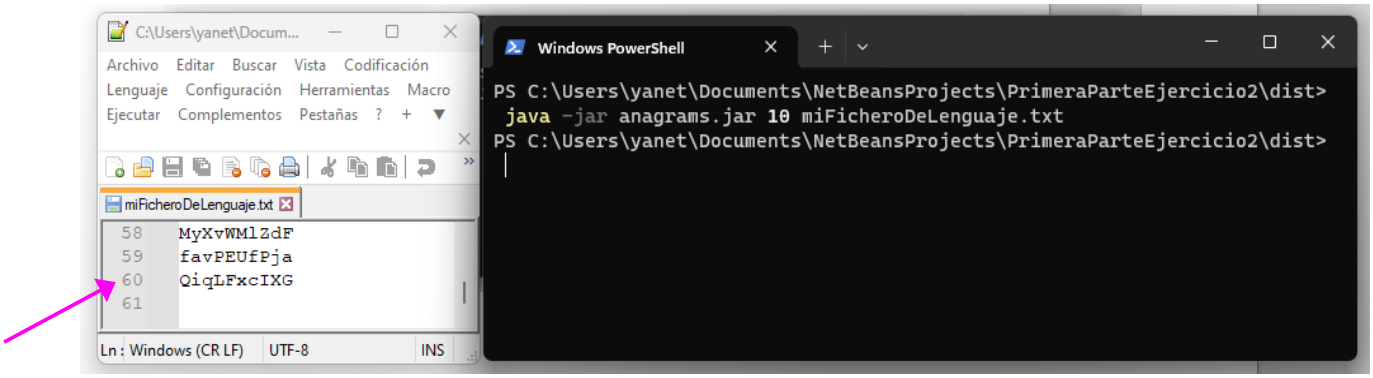


- Ejecución en un entorno concurrente

Para comprobar que se ejecuta correctamente en un entorno concurrente de ejecución compartiendo un recurso, lo que haremos será tener el archivo miFicheroDeLenguaje.txt abierto cuando ejecutemos el programa para comprobar que comparte el recurso correctamente sin generar ningún error.



Refrescamos el archivo y comprobamos que se ha actualizado correctamente.



Ahora ejecutaremos ***-jar lenguaje.jar 500 miFicheroDeLenguaje.txt*** y podremos comprobar que la lista es enorme.

Figura 1: Descripción gráfica de la lista (Es muy grande)

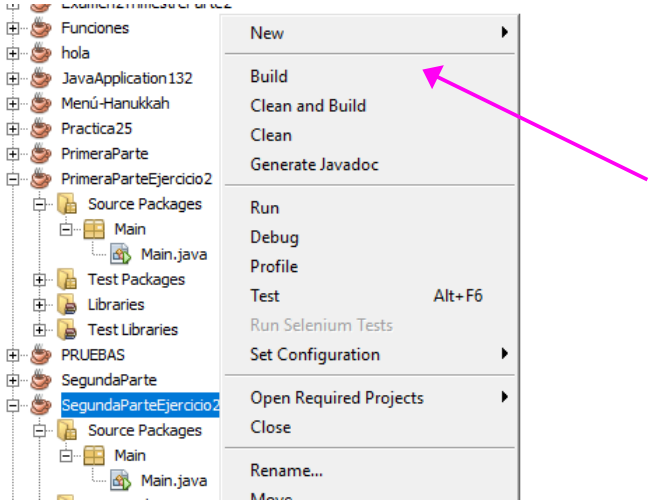
***Segunda parte**: implementa una aplicación, llamada 'colaborar', que lance al menos 10 instancias de la aplicación "lenguaje". Haciendo que todas ellas, colaboren en generar un gran fichero de palabras. Cada instancia generará un número creciente de palabras de 10, 20, 30, … Por supuesto, cada proceso 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 + 20 + 30 + … + 100 = 550 líneas*.

Para realizar esta actividad lo primero que haremos será crear un nuevo proyecto, en el tendremos nuestra clase Main y en ella tendremos main() y la función generarStringAleatorio().
Lo primero que haremos será crear un entero e inicializarlo, haremos un try cath para controlar los errores y dentro de este haremos un bucle for, el bucle se ejecutará 10 veces, y cada vez que se ejecute le sumará a cantidad 10, se creara un ProssesBuilder en el que se almacenará el comando indicándole la cantidad, redirijimos las entradas y salidas del proceso hijo al padre e iniciamos el proceso. Por último cerramos el try con el catch, este se ejecutará si hay errores dentro de nuestro catch mostrándonos el error por pantalla.

```
public class Main {
    public static void main(String[] args) {
        //CREAMOS UN ENTERO Y LO INICIALIZAMOS DANDOLE EL VALOR CERO
        int cantidad = 0;

        //USAMOS UN TRY CATCH PARA EL CONTROL DE ERRORES PARA QUE NO NOS EXPLOTE EL PC
        try {
            //HACEMOS UN BUCLE FOR EL CUAL SE EJECUTARÁ 10 VECES
            for (int i = 0; i < 10; i++) {
                //CADA VEZ QUE SE EJECUTE EL BUCLE FOR LA CAMTIDAD DE CONJUNTOS DE LETRAS INCREMENTARA EN 10 UNIDADES
                cantidad = cantidad + 10;
                //CREAMOS UN OBJETO ProcessBuilder PARA EJECUTAR EL COMANDO EN EL Q LLAMAMOS A lenguaje INDICANDOLE LA CANTIDAD
                ProcessBuilder processBuilder = new ProcessBuilder("java -jar lenguaje.jar " + cantidad + " miFicheroDeLenguaje.txt".split(" "));
                //REDIRIGIMOS LAS ENTRADAS Y SALIDAS AL PROCESO PADRE
                processBuilder.inheritIO();
                //INICIAMOS EL PROCESO
                Process proceso = processBuilder.start();
            }
        //CAPTURAMOS Y MANEJAMOS EL ERROR IOException SI LO HAY
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

Por último haremos click derecho en nuestro proyecto y pulsaremos en “Build”, esto generará un archivo .jar en una carpeta llamada dir de nuestro proyecto, en mi caso la ruta de mi jar estará en: C:\Users\yanet\Documents\NetBeansProjects\SegundaParteEjercicio2\dist

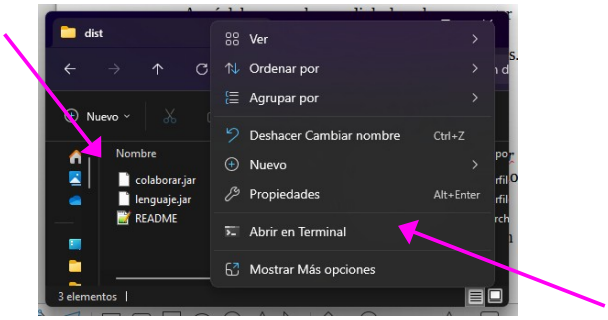


Pruebas

- Lanzando aplicaciones simultáneas.

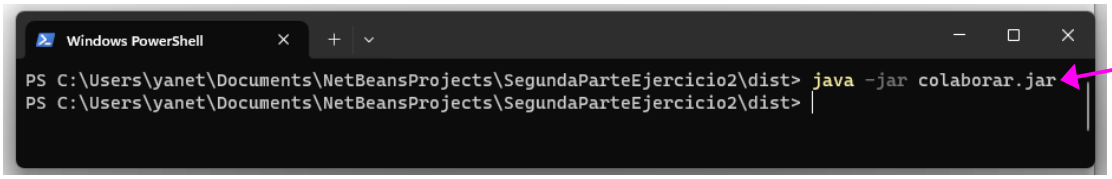
Para realizar la ejecución en modo aislado nos ubicaremos en la ruta de nuestro archivo .jar y pegamos el archivo lenguaje.jar

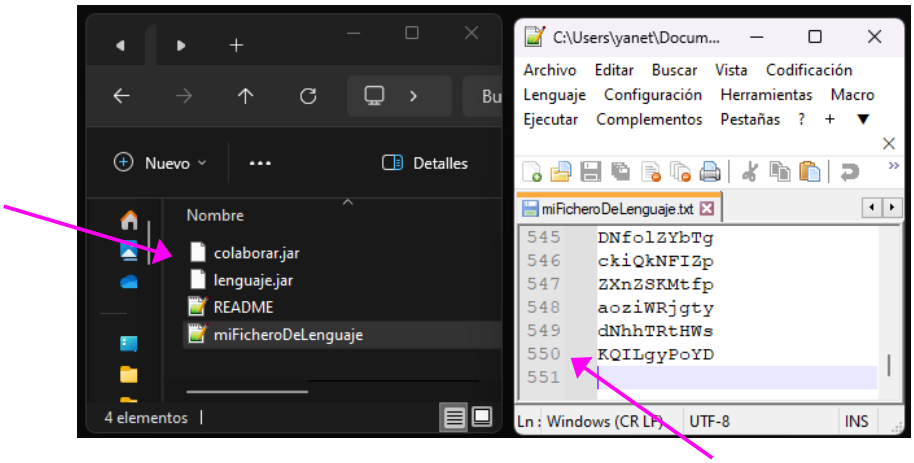
Aquí deberemos hacer click derecho y escoger la opción de “Abrir terminal”, esto nos abrirá el cmd ubicándonos en la ruta en la que estamos.



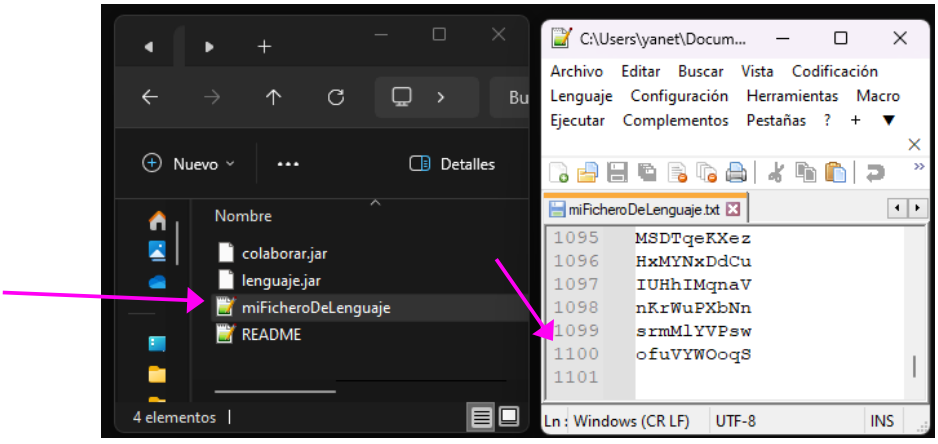
Ejecutamos en el terminal java java ***java -jar colaborar.jar*** y comprobamos que nuestro código funciona como queremos, haremos varias pruebas para probar que no hay fallos.

Aquí vemos como nuestro .jar crea correctamente un documento llamado miFicheroDeLenguaje.txt y le añade las 550 lineas esperadas.





Y aquí vemos como al ejecutarlo otra vez le añade 550 lineas mas al documento.



Conclusiones.

Después de realizar esta actividad, se pueden llegar a las siguientes conclusiones:

La implementación de la aplicación "lenguaje" permite generar conjuntos de letras de forma aleatoria y escribirlos en un fichero especificado por el usuario. Esto se logra utilizando argumentos en la línea de comandos para indicar el número de conjuntos de letras a generar y el nombre del fichero donde se guardará la información.

La implementación de la aplicación "colaborar" permite lanzar múltiples instancias de la aplicación "lenguaje" para generar un gran fichero de palabras colaborativamente. Cada instancia genera un número creciente de palabras de tamaño fijo y las escribe en el fichero correspondiente.

El resultado final es un fichero que contiene la suma de las palabras generadas por cada instancia de "lenguaje". El número total de líneas en el fichero se obtiene sumando los números de palabras generadas por cada instancia (10 + 20 + 30 + ... + 100).

Recursos.

Recursos necesarios para realizar la Tarea:

- IDE NetBeans.
- Contenidos de la unidad.
- Ejemplos expuestos en el contenido de la unidad.

Bibliografía.


https://www.biblia.es/biblia-online.php


https://www.churchofjesuschrist.org/study/scriptures?lang=spa


https://pastoralsj.org/biblia


https://www.biblija.net/biblija.cgi?l=es


https://www.bibliatodo.com/la-biblia