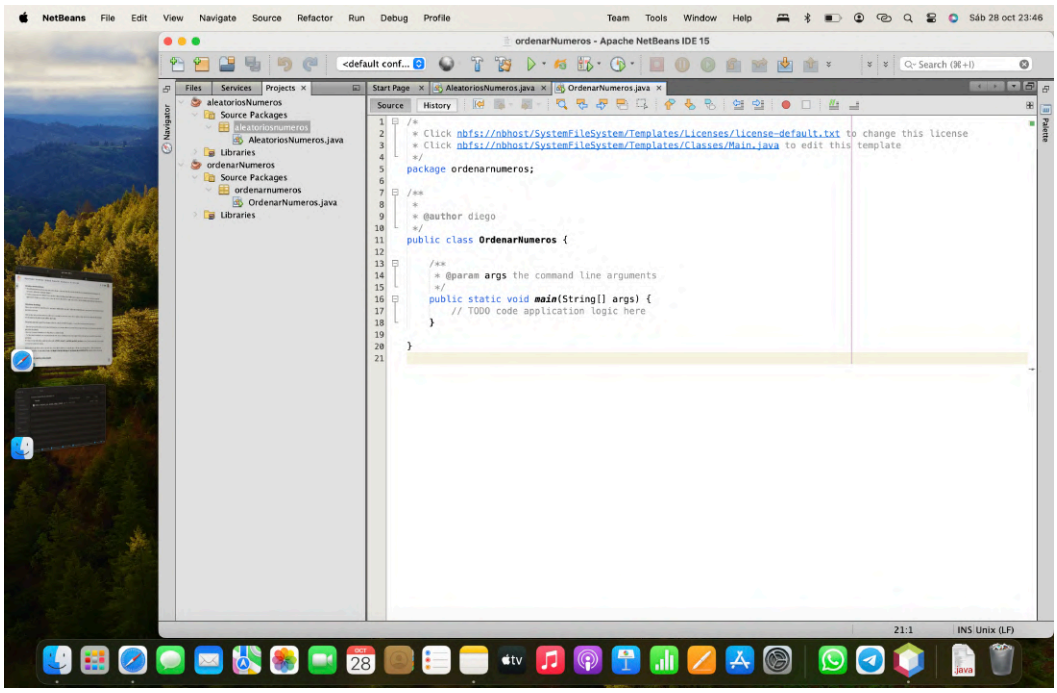
# Tarea 1 para Programación de Servicios y Procesos

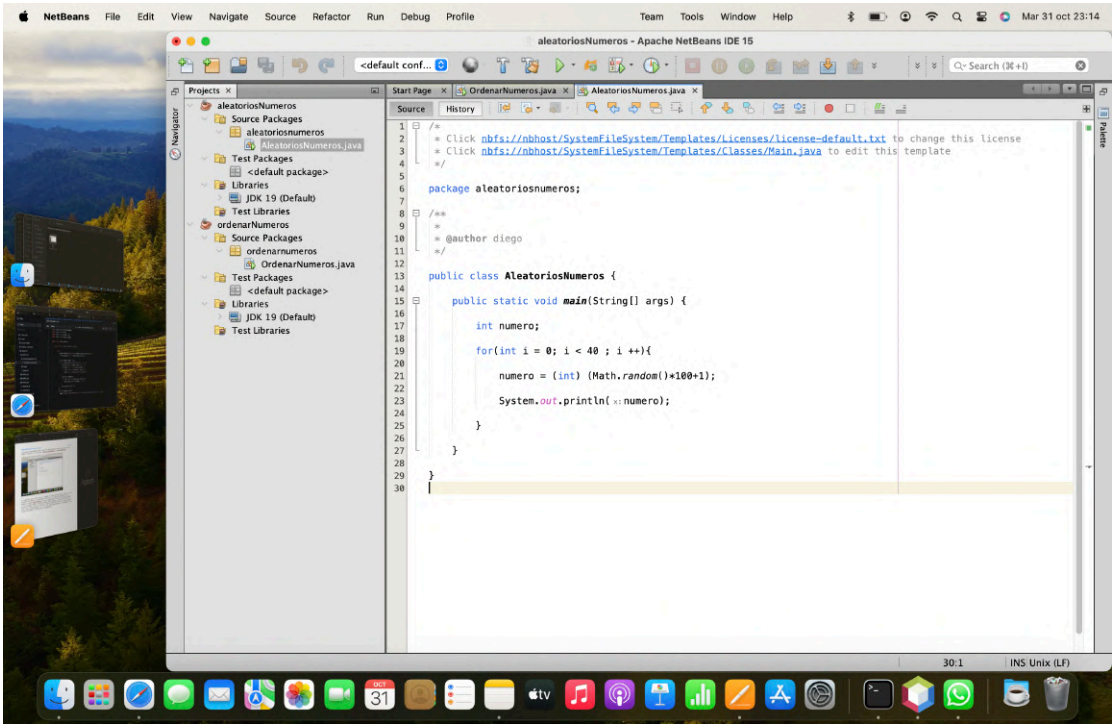
Diego Manuel Carrasco Castañares

Ejercicio 1:

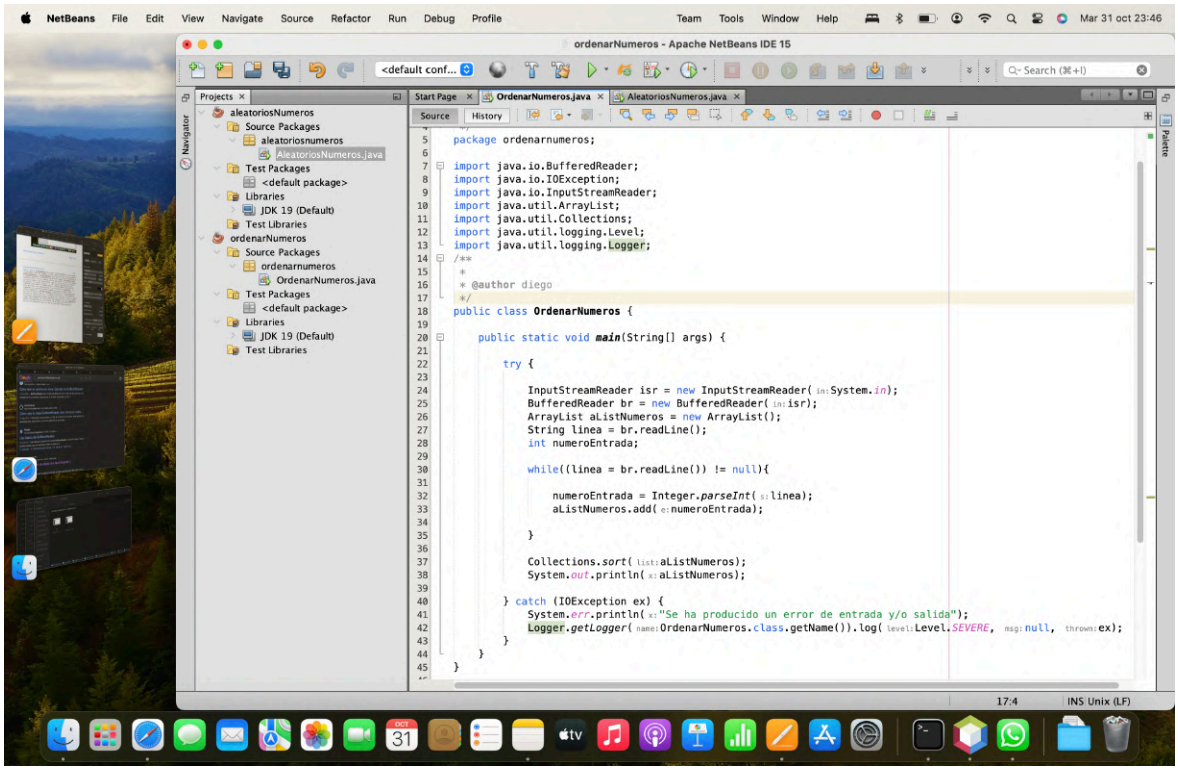
Lo primero que haremos será crear los dos proyectos en netbeans con los nombre indicados en la tarea, aleatoriosNumeros y ordenarNumeros.



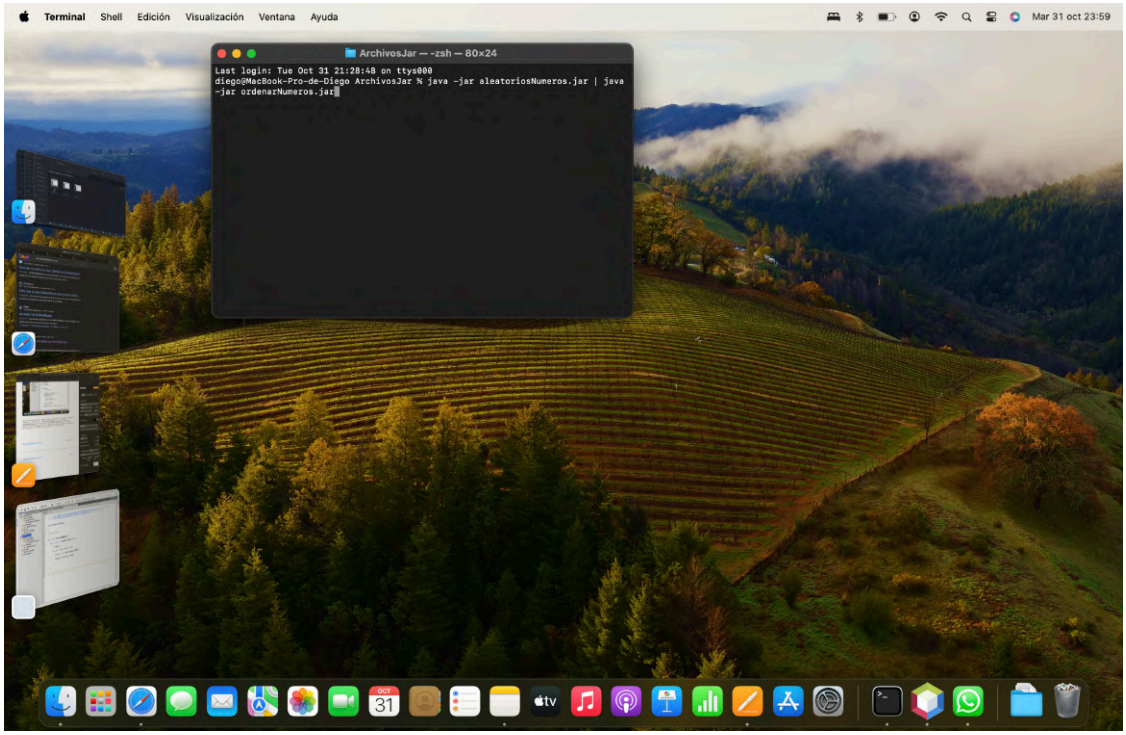

El siguiente paso será crear el código de cada una de ellas. Para el caso de aleatoriosNumeros lo que hemos realizado es crear un int llamado número, un bucle for que se repite 40 veces en cuyo interior tenemos la variable numero declarada anteriormente a la cual le asignamos un número random entre el 1 y el 100 para cada una de las veces que se repite el bucle for y lo imprimimos por pantalla.



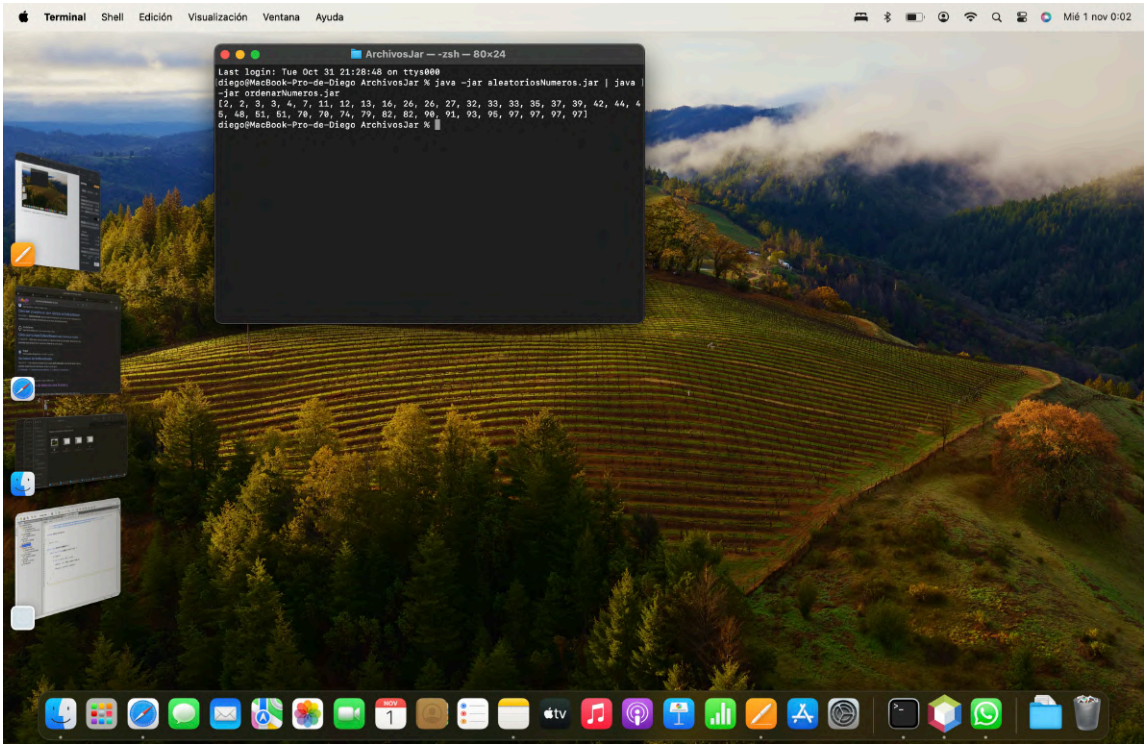
Para el caso de ordenarNumeros declaramos InputStreamReader para recoger lo que pueda entrar por sistema (en este caso la salida de la aplicación aleatoriosNumeros), un BufferedReader para leer las líneas almacenadas en el InputStreamReader, un ArrayList de enteros llamado aListNumeros para almacenar cada uno de los números enteros que recibe y tiene que ordenar, un string llamado línea para guardar cada uno de los número que recibe por pantalla en formato string, y un int llamado numeroEntrada para almacenar cada número en formato entero.
Posteriormente declaramos un while que tiene como condición que lo leído mediante el buffered y lo almacenado en linea sean iguales y ambos distintos de null. En su interior mediante un Integer.ParseInt, convertimos el string almacenado en linea en un int y ese int le añadimos al ArrayList.
Lo siguiente que haremos, mediante la clase Collections y su método sort, será ordenar de menor a mayor el ArrayList para posteriormente imprimirlo por pantalla.
Como el código creado anteriormente nos puede generar una excepción del tipo IOException, debemos envolver todo en un try-catch. En el catch imprimimos por pantalla un mensaje de error en caso de que salte dicha excepción.




Ahora, mediante terminal, vamos a crear la conexión de tubería. Para ello usaremos el comando java -jar aleatoriosNumeros.jar | java -jar ordenarNumeros.jar una vez que nos encontremos situados en la carpeta donde los tenemos almacenados. En nuestro caso hemos copiado los archivos jar generados por NetBeans en una carpeta en el escritorio.
En la siguiente imagen podemos ver el comando escrito en el terminal.




La siguiente imagen muestra el resultado una vez pulsado intro.




Como podemos apreciar nos muestra los 40 número generados por la aplicación aleatoriosNumeros ordenados de mayor a menor.

Ejercicio 2:

Para el ejercicio 2 crearemos dos proyectos.

El primero de ellos se llamará lenguaje, tal como indica el ejercicio. Como debemos controlar las excepciones vamos a hacerlo mediante try - catch, tres en nuestro caso. El primero controlara la excepción de pasarle dos parámetros o argumentos al programa. El segundo controlara que introduzcamos el número de letras y palabras que generara el programa y el tercero controlara que se introduzca el nombre con el que queramos nombrar al fichero donde guardaremos las palabras generadas por el programa.

En el primer try - catch crearemos un if y los dos tray-catch restantes.

En el segundo try - catch crearemos una variable que guardara el primer argumento que le pasemos (en nuestro caso el número de letras y palabras que generara el programa) y el tercer try - catch.

Dentro del último try - catch vamos a crear todo el código de la aplicación.
En nuestro caso constara de la creación de las variables necesaria tales como un string que contendrá cada palabra llamado cadena, un bloqueo llamado Lock, un string que contendrá el nombre del fichero donde guardaremos las palabras y un string donde almacenaremos el nombre el sistema operativo en uso.

Tras ello crearemos un if para saber si es windows u otro sistema operativo y lo guardamos en su variable correspondiente. Si es windows la path del archivo debe contener los caracteres \, en cambio para otro sistemas operativos dicha path contiene los caracteres / para llegar hasta el directorio.

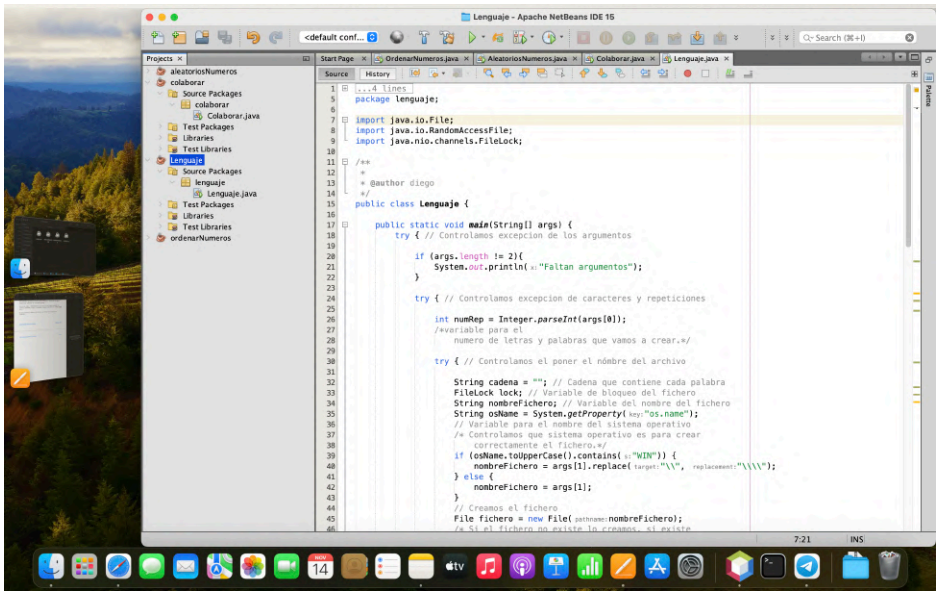
Posteriormente crearemos un variable fichero con el nombre del fichero. Tras ello, mediante un if, controlaremos que el fichero no exista, si es así lo creamos, en caso contrario no creamos un fichero al existir ya uno con el mismo nombre.

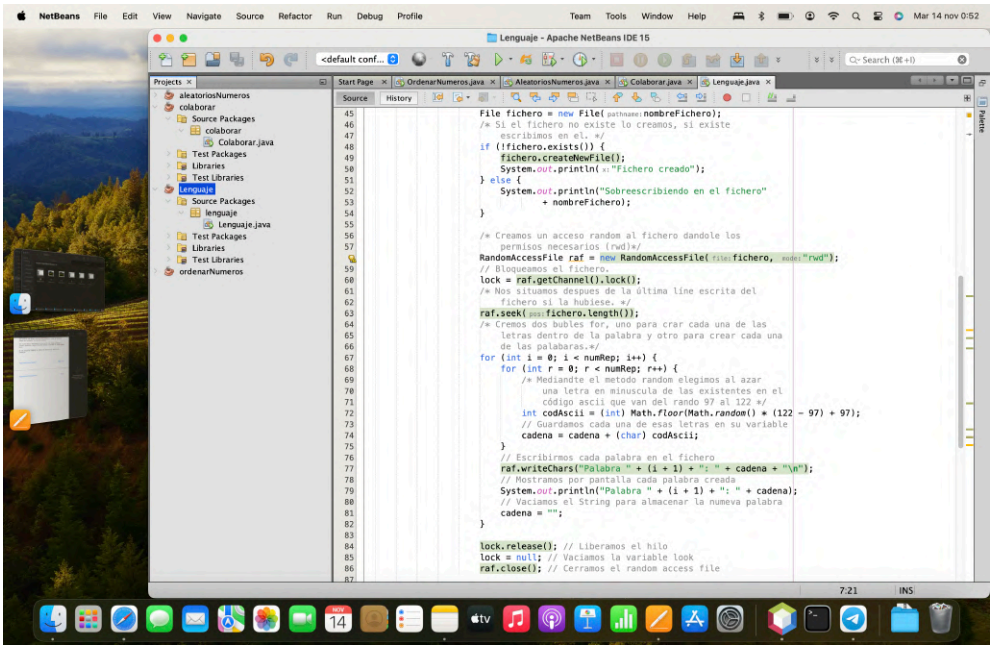
Tras ello crearemos un random access file llamado raf para crear un acceso por un hilo aleatorio al fichero, bloquearemos ese hilo para que ningún otro programa pueda acceder a dicho archivo mientras esta en uso y tras ello nos situaremos, mediante seek, al final de la última linea escrita en el fichero (si la hubiese).

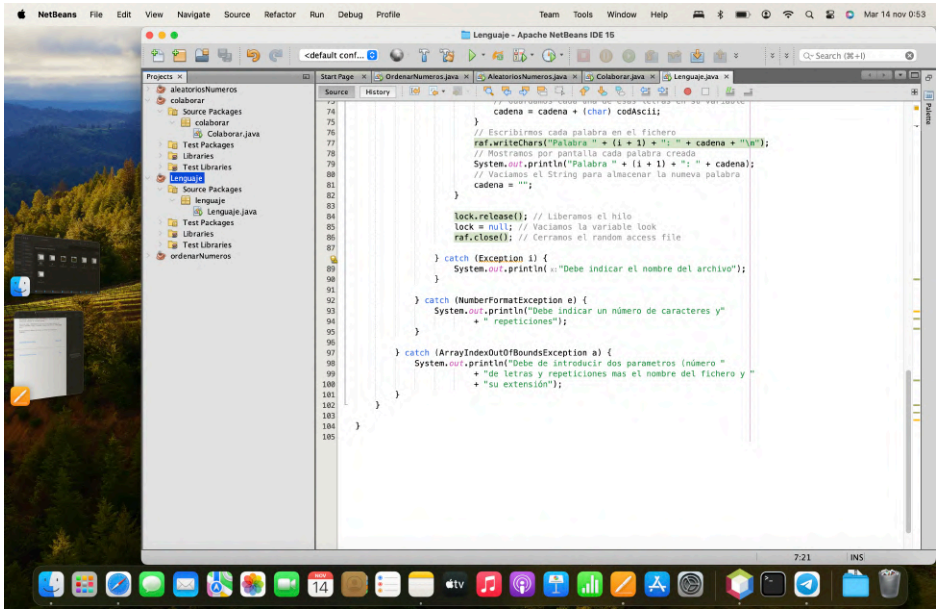
Mediante dos bucles for vamos a crear (mediante el bucle for interior) las letras que compondrán cada palabra a través de un random y los códigos ascii que lo componen (las letras del abecedario en minúscula sin contar la ñ, están comprendidas entre el 97 y el 122) para luego ir guardando en el string cadena cada una de esas letras. Una vez compuesta cada palabra por los caracteres necesarios vamos a escribir en el fichero dicha palabra mediante el comando writeChars para posteriormente mostrar por pantalla cada palabra escrita en el fichero y vaciar la variable cadena para almacenar la siguiente palabra.

Por último vamos a desbloquear el hilo por el cual hemos accedido al fichero, pasaremos a null la variable bloqueo y cerraremos el random access file.

En las siguientes imágenes se muestra el código de la aplicación lenguajes.


Lenguaje - Apache NetBeans IDE 15
Lenguaje.java
package lenguaje;
import java.io.File;
import java.io.RandomAccessFile;
import java.nio.channels.FileLock;
public class Lenguaje {
public static void main(String[] args) {
try { // Controlamos excepcion de los argumentos
if (args.length != 2){
System.out.println("Faltan argumentos");
}
try { // Controlamos excepcion de caracteres y repeticiones
int numRep = Integer.parseInt(args[0]);
try { // Controlamos el poner el nombre del archivo
String cadena = ""; // Cadena que contiene cada palabra
FileLock lock; // Variable de bloqueo del fichero
String nombreFichero; // Variable del nombre del fichero
String osName = System.getProperty("os.name");
if (osName.toUpperCase().contains("WIN")) {
nombreFichero = args[1].replace("\\", "\\\\");
} else {
nombreFichero = args[1];
}
// Creamos el fichero
File fichero = new File(nombreFichero);


Lenguaje - Apache NetBeans IDE 15
File fichero = new File(nombreFichero);
if (!fichero.exists()) {
fichero.createNewFile();
System.out.println("Fichero creado");
} else {
System.out.println("Sobreescribiendo en el fichero"
+ nombreFichero);
}
RandomAccessFile raf = new RandomAccessFile(fichero, "rwd");
lock = raf.getChannel().lock();
raf.seek(fichero.length());
for (int i = 0; i < numRep; i++) {
for (int r = 0; r < numRep; r++) {
int codAscii = (int) Math.floor(Math.random() * (122 - 97) + 97);
cadena = cadena + (char) codAscii;
}
raf.writeChars("Palabra " + (i + 1) + ": " + cadena + "\n");
System.out.println("Palabra " + (i + 1) + ": " + cadena);
cadena = "";
}
lock.release();
lock = null;
raf.close();


Lenguaje - Apache NetBeans IDE 15
} catch (Exception i) {
System.out.println("Debe indicar el nombre del archivo");
}
} catch (NumberFormatException e) {
System.out.println("Debe indicar un número de caracteres y"
+ " repeticiones");
}
} catch (ArrayIndexOutOfBoundsException a) {
System.out.println("Debe de introducir dos parametros (número "
+ "de letras y repeticiones mas el nombre del fichero y "
+ "su extensión");
}
}
}

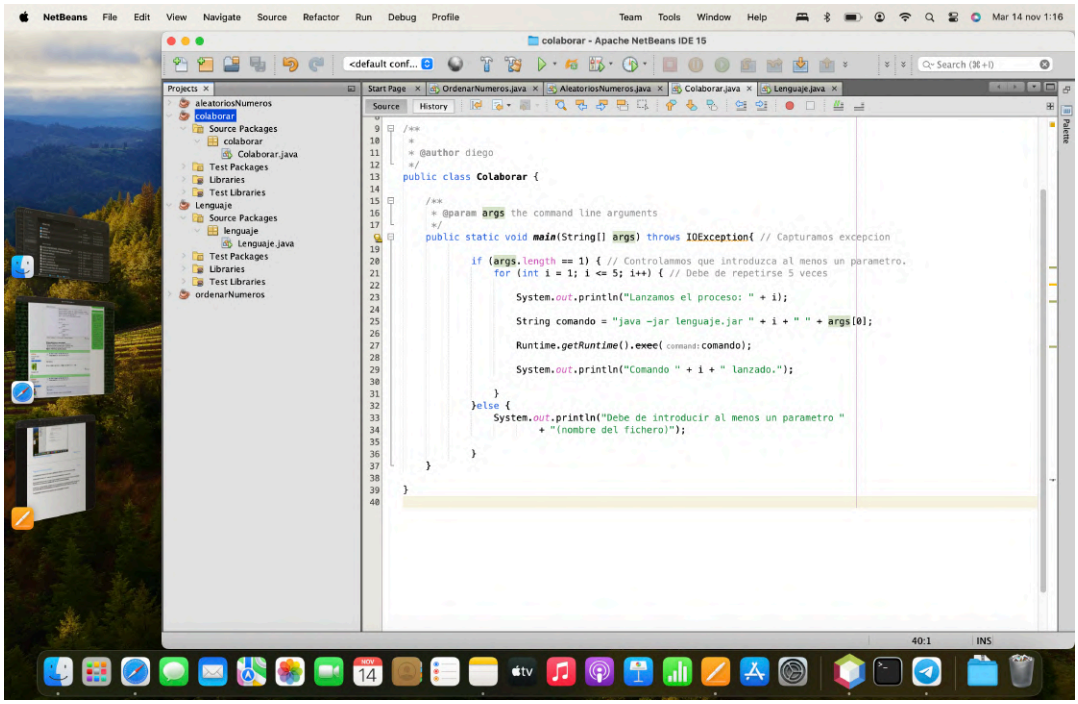
Lo siguiente que haremos será crear la aplicación colaborar, como indica el enunciado.

En ella vamos a crear un if - else par controlar que le pasamos un parámetro.

Dentro del if vamos a crear un bucle for que se repetirá cinco veces (como indica el enunciado) . Este bucle nos mostrara por pantalla cada uno de los procesos lanzados.

Crearemos un string llamado comando con de le pasaremos el comando que tiene que ejecutar (en nuestro caso java -jar lenguajes.jar y sus argumento (nombre del archivo en este caso)).

Posteriormente mostraremos por pantalla que se ha lanzado el comando.



**Enunciado.**

Ejercicio 1) Se compone de 3 partes:

- Primera parte : implementa una aplicación, llamada 'aleatoriosNumeros', que genere 40 números aleatorios (entre 1 y 100), y que los escriba en su salida estándar.
- Segunda 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'.
- Tercera parte: Usar el operador "|" (tubería) para que la salida de la aplicación 'aleatoriosNumeros' sea la entrada de la aplicación 'ordenarNumeros'.

Ejercicio 2) Se compone de 2 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.jar 5 texto.txt Indicando que se generarán 5 palabras de 5 letras y serán guardadas, en el recurso compartido, texto.txt

- Segunda parte: implementa una aplicación, llamada 'colaborar', que lance al menos 5 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. Por supuesto, cada proceso seguirá escribiendo su palabra en una línea independiente de las otras. Es decir, si lanzamos 5 instancias de "lenguaje", al final, debemos tener en el fichero 15 líneas (1+2+3+4+5) .

**Criterios de puntuación. Total 10 puntos.**

Actividad 1) 5 puntos, de los cuales:

- Aplicación "aleatorioNumeros"→ 2p.
- Aplicación "ordenarNumeros" → 2p.
- Combinar aleatorioNumeros con ordenarNumeros → 1p.
- Actividad 2) 4 puntos, de los cuales:
- Aplicación "lenguaje" → 2p.
- Aplicación "colaborar" → 2p.
- Documento de procesador de texto:
- Explicando la realización y ejecución de los ejercicios. → 1p.

**Recursos necesarios para realizar la Tarea.**

- NetBeans IDE 8.2
- Contenidos de la unidad.
- Ejemplos expuestos en el contenido de la unidad.

Específico para el Ejercicio2: Puedes tomar como base el ejemplo visto en el apartado 6.1 Regiones críticas de esta unidad. Si utilizas objetos de tipo RandomAccessFile para el acceso a ficheros, recuerda que puede ser necesario que te posiciones al final del fichero para ir añadiendo valores en el fichero uno detrás de otro.

**Consejos y recomendaciones.**

- Recuerda que para que un proceso tome como entrada los datos generados por otro, debes lanzar su ejecución desde el intérprete de comandos utilizando el operador tubería "|".
- Puedes programarte una aplicación que cuente el número de líneas de un fichero, para comprobar el correcto resultado del segundo ejercicio. En GNU/Linux, cuentas con el comando "wc -l nombreArchivo" que te devolverá el número de líneas que contiene nombreArchivo.

**Indicaciones de entrega.**

Debes crear una fichero comprimido en el que debes incluir el o los proyectos Netbeans y documentos que correspondan a la realización de los ejercicios de la tarea.

Además adjuntarás un documento realizado con un procesador de texto con unas breves explicaciones de cómo has realizado los ejercicios, con las capturas de pantallas que estimes oportunas.

*Por ejemplo para esta unidad 01, entregaras el fichero comprimido PSP01_Tarea01_... con los directorios, ejercicio1 y ejercicio2:*

*- Ejercicio1 (proyecto Netbeans) con aleatorioNumeros.java y ordenarNumeros.java además este ejercicio requiere una captura de pantalla de la ejecución con tubería;*

*-Ejercicio2, (proyecto Netbeans) con lenguaje.java y colaborar.java.*

*- Y el documento realizado con un procesador de texto con las explicaciones de ambos ejercicios con las capturas de pantallas que estimes oportunas.*

El fichero comprimido debe siguiente nomenclatura PSPXX_TareaZZ_ apellido1_apellido2_nombre, donde XX se corresponde con la unidad y ZZ con el numero de la tarea.

Realiza el envío de dicho archivo comprimido a través de la plataforma. Asegúrate que el nombre no contenga la letra ñ, tildes ni caracteres especiales extraños. Así por ejemplo la alumna Begoña Sánchez Mañas para la primera unidad del MP de PSP, debería nombrar esta tarea como...

**PSP01_Tarea01_sanchez_manas_begona**