

Práctica 2

Descripción del trabajo a realizar

Actividad 1.- Serie de Fibonacci

Diseñar, codificar y documentar adecuadamente (para la generación de **javadoc**) un programa Java (fichero `Fibonacci.java`) que calcule la serie de Fibonacci de un número concreto de términos y la guarde en un fichero de texto.

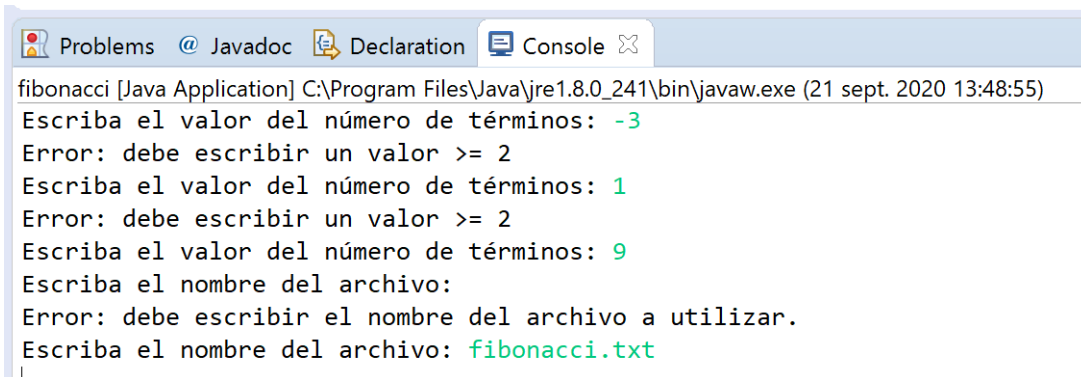
Información sobre la serie en https://es.wikipedia.org/wiki/Sucesi%C3%B3n_de_Fibonacci.

Para ello, empezará leyendo del terminal un número entero denominado **numTérminos**. Se pedirá este valor y se leerá hasta que el valor introducido sea **mayor o igual que 2**. A continuación, leerá del terminal el nombre de un fichero de texto en el que se quiere guardar la serie de Fibonacci (por ejemplo `fibonacci.txt`). Se pedirá el nombre de fichero y se leerá la cadena de caracteres introducida hasta que el valor introducido no sea una cadena vacía. Finalmente, se generará la serie con el número de términos escogido y se guardará en el fichero.

El alumno debe abrir el fichero y comprobar que la serie se ha generado y se ha guardado correctamente.

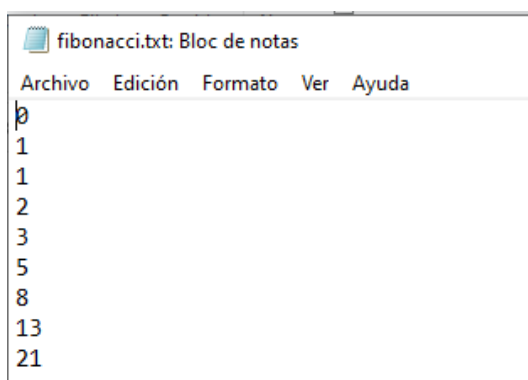
Ejemplo de ejecución:

Si se teclean los siguientes valores:



```
fibonacci [Java Application] C:\Program Files\Java\jre1.8.0_241\bin\javaw.exe (21 sept. 2020 13:48:55)
Escriba el valor del número de términos: -3
Error: debe escribir un valor >= 2
Escriba el valor del número de términos: 1
Error: debe escribir un valor >= 2
Escriba el valor del número de términos: 9
Escriba el nombre del archivo:
Error: debe escribir el nombre del archivo a utilizar.
Escriba el nombre del archivo: fibonacci.txt
```

El contenido del archivo `fibonacci.txt`, será:



```
fibonacci.txt: Bloc de notas
Archivo Edición Formato Ver Ayuda
0
1
1
2
3
5
8
13
21
```

Para familiarizarse con el uso de la clase `Scanner` se sugiere comenzar el trabajo mediante un programa de prueba que lea un único número entero por la entrada estándar (teclado) y lo muestre por la salida estándar (el monitor). Puede consultar la documentación de la clase `Scanner` en la API de Java (<https://docs.oracle.com/javase/8/docs/api/>). El código sería:

```
import java.util.Scanner;

public final class EjemploScanner
{
    public static void main ( String[] args )
    {
        final Scanner sc = new Scanner ( System.in );
        int numero;

        System.out.print ( "Escriba un número entero: " );
        numero = sc.nextInt();
        System.out.println ( "Se ha leído el número " + numero );
        sc.close();
    }
}
```

Actividad 2.- Sustitución de una cadena de caracteres por otra

Diseñar, codificar y documentar adecuadamente (para la generación de **javadoc**) un programa Java que sustituya las ocurrencias de una cadena de caracteres por otra en una serie de líneas de texto (que codificará en un archivo llamado `CambiaCadenas.java`). Para ello leerá de la entrada por defecto (la consola) los siguientes elementos:

- 1) El nombre del fichero de texto que contiene el texto original a procesar.
- 2) La secuencia de texto a buscar en cada línea del texto original.
- 3) La cadena de caracteres por la cual se debe sustituir cada secuencia de texto encontrada.

El programa leerá de manera secuencial cada una de las líneas del texto original y, para cada línea, realizará la sustitución de todas las secuencias encontradas por la cadena que la sustituye, obteniendo así una línea resultado. Estas líneas se guardarán en un fichero de texto de salida de nombre `"resultado.txt"`. Si el fichero existe antes de ejecutar la aplicación, su contenido se sustituye por el generado en esta ejecución.

Para que se distinga qué líneas se han modificado, se antepondrá la secuencia `"LÍNEA CAMBIADA:"` a cada línea modificada. Cuando la línea leída esté vacía (no contenga ningún carácter) se escribirá una línea con la secuencia `"LÍNEA VACÍA"` en el fichero de salida.

El programa terminará, escribiendo en la pantalla el número de líneas en las que ha realizado alguna sustitución y el número de líneas vacías en el fichero origen.

Por ejemplo, si el fichero de origen contiene:

```
El lento realiza las tareas de forma lenta.  
  
El constante siempre alcanza sus objetivos.  
  
Avanza lentamente por la senda.  
Este es el camino.  
  
Mejor llegar tranquilo que no llegar.
```

y la secuencia a buscar es `"lent"` y se cambiará por `"rapid"` debería crear un fichero de salida con:

```
LÍNEA CAMBIADA: El rapido realiza las tareas de forma rapida.  
LÍNEA VACÍA  
El constante siempre alcanza sus objetivos.  
LÍNEA VACÍA
```

LÍNEA CAMBIADA: Avanza rápidamente por la senda.
Este es el camino.
LÍNEA VACÍA
Mejor llegar tranquilo que no llegar.

Y por la salida debería aparecer un mensaje similar al siguiente:

Se han realizado sustituciones en 2 líneas.
Se han leído 3 líneas vacías.

El programa debe controlar los siguientes errores de introducción de datos:

- No introducir el nombre del archivo.
- No introducir la cadena a buscar.
- No introducir la cadena para la sustitución.
- Que la cadena a buscar y la cadena por la que se debe sustituir coincidan.

Se muestran a continuación unos ejemplos de ejecución cuando ocurre alguno de estos errores. El tratamiento de los errores se deja a su criterio, es decir, que no tiene por qué coincidir con los ejemplos. En estos ejemplos se ha decidido que cuando ocurre un error el programa termina, pero el error podría tratarse de otra forma, por ejemplo, obligando al usuario a introducir de nuevo los datos.

Ejemplo del error al no introducir el nombre del archivo a utilizar.

Escriba el nombre del archivo a utilizar:
Error: debe escribir el nombre del archivo a utilizar.

Ejemplo del error al no introducir la secuencia a buscar.

Escriba la secuencia a buscar:
Error: debe escribir la secuencia a buscar.

Ejemplo del error al no introducir la cadena de sustitución.

Escriba la secuencia a buscar: oro
Escriba la cadena para la sustitución:
Error: debe escribir la secuencia a buscar.

Ejemplo del error cuando la secuencia a buscar y la cadena para la sustitución coinciden.

Escriba la secuencia a buscar: oro
Escriba la cadena para la sustitución: oro
Error: la cadena a buscar y la cadena por la que se sustituye no pueden coincidir.

Para hacer esta actividad es imprescindible que consulte la API de Java sobre la clase `String` donde podrá encontrar los métodos que necesita; por ejemplo, para saber si un `String` está vacío, conocer su tamaño, compararlo con otro `String` para saber si son iguales o realizar la sustitución, dentro de un `String`, de todas las ocurrencias de un `String` por otro `String`. También es interesante consultar la documentación de la clase `Scanner`.

El API de Java para la versión 8 se puede consultar en:

<https://docs.oracle.com/javase/8/docs/api/>

El API de Java para la versión 15 se puede consultar en:

<https://docs.oracle.com/en/java/javase/15/docs/api/index.html>

<https://docs.oracle.com/en/java/javase/15/docs/api/java.base/java/lang/String.html>

<https://docs.oracle.com/en/java/javase/15/docs/api/java.base/java/util/Scanner.html>

Consideraciones sobre las implementaciones de las actividades a realizar

Para la realización de esta práctica se supondrá que los datos que se introducen son correctos, es decir, son numéricos cuando deben serlo, o de tipo `String` cuando sea el caso. Si no fuera así, el programa terminará al hacer las conversiones. En esta práctica no se considera esta situación como un error a tratar. Este tipo de situaciones se abordará en la asignatura más adelante.