PROG04 Tarea de evaluación 01. Realiza un programa en Java (100%)

Límite de entrega: viernes, 23 de diciembre de 2022, 23:59
Ficheros requeridos: simple.txt, tango.txt (Descargar)
Número máximo de ficheros: 3
Tipo de trabajo: Individual

PROG04 TE01 REALIZAR UN PROGRAMA EN JAVA (100%)

Enunciado

El objetivo de este programa es **trabajar la lectura y escritura de ficheros de texto y el manejo de cadenas de caracteres**.

Para ello, vamos a **gestionar una serie de "Mad Libs"**. Los "Mad Libs" son pequeñas historias a las que les falta una serie de palabras que hay que rellenar. Alguien que no conoce la historia rellenará esos huecos en base a las características que le pidan (sustantivo femenino, verbo infinitivo...). Luego, se leerá la historia resultante. Suelen utilizarse en el aprendizaje de idiomas.

En primer lugar, el programa mostrará la siguiente **introducción**:

```
Bienvenidos y bienvenidas al juego de los cuentos locos.
El programa te pedirá que introduzcas una serie de palabras
que se utilizarán para completar una historia.
El resultado se guardará en un fichero.
Puedes leer esas historias siempre que quieras.
```

En esta tarea, se presentará un **menú con 3 opciones**:

```
******* MENU *******
(C)rear un "Mad Lib"
(V)er un "Mad Lib"
(S)alir
Elija su opción: f

******* MENU *******
(C)rear un "Mad Lib"
(V)er un "Mad Lib"
(S)alir
Elija su opción: c
```

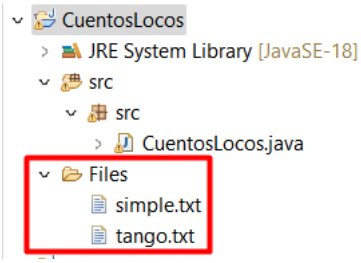
Se pulsarán C, V y S respectivamente para elegir cada una de ellas. Se podrán pulsar tanto en mayúsculas como en minúsculas y si se pulsa cualquier otra cosa, se pedirá una nueva opción. En caso de teclear una palabra en vez de una letra, se tratará la primera letra.
Ejemplo:
Ver, Viento ==> Se tratará la V que corresponde a la opción Ver
hola ==> al ser "h" el primer carácter volverá a aparecer el menú.

IMPORTANTE: se requiere el uso de **variables de tipo *char*** para tratar las diferentes opciones del menú.

El menú debe funcionar independientemente del orden en que se elijan las opciones y cada opción se podrá elegir todas las veces que se quiera.

PARA LEER FICHEROS DESDE ECLIPSE

Si tenéis problemas para que Eclipse os reconozca los TXT, os aconsejamos crear una carpeta Files, Ficheros,... etc. y añadir ahí los TXT.



Para leer simple.txt/tango.txt como cuando creamos salida.txt, tendremos que utilizar el nombre de la carpeta así, antes del nombre introducido.

```
File fichEntrada = new File("./Files/"+nombre);
```

Si se elige **crear un nuevo "Mad Lib"**, el programa pedirá un fichero de entrada y otro de salida.

```
Crear un cuento:
Nombre del fichero que quieres leer: simple
Fichero no encontrado. Inténtalo otra vez
Nombre del fichero: simple.txt
Nombre del fichero de salida: salida.txt
```

En el caso del fichero de entrada, tanto para crear como para leer un "Mad Lib", se pedirán nombres hasta que el fichero introducido sea válido y se pueda leer.

No ocurrirá lo mismo para el fichero de salida. Si éste no existe se creará y si ya existe se reescribirá (comportamiento por defecto). Se supone que el fichero de entrada y el de salida son distintos.

A continuación, el programa leerá el fichero, detectará los huecos y pedirá al usuario o usuaria que los rellene sin mostrarle el resto de la historia. Al mismo tiempo, guardará la historia resultante en el fichero de salida para que pueda ser leído cuando se elija la opción correspondiente.

```
infinitivo: cantar
adjetivo masculino: colorado
El cuento loco ha sido creado
```

Puedes **utilizar los ficheros de ejemplo que se adjuntan o crear tus propios ficheros**. Todos deberán guardarse en la misma carpeta que el programa.

Básicamente, son ficheros de texto en los que se indican los huecos mediante texto que empieza por el carácter '<' y acaba con '>'. Ese texto además, puede estar compuesto por varias palabras separadas por un guión.

Por ejemplo:

```
En vacaciones quiero <infinitivo>
Creo que es muy <adjetivo-masculino>
Deberías probarlo
```

El programa debe leer el fichero por líneas y luego elemento a elemento utilizando objetos de la clase Scanner. El texto normal se escribirá directamente en el fichero de salida mientras que cuando se encuentre uno de los elementos que identifican un hueco se pedirá al usuario o usuaria que lo rellene antes de escribirlo al fichero de salida.

Se debe aceptar cualquier respuesta que se dé, incluso **respuestas de varias palabras o respuestas vacías**.

Se supone que todos los elementos del fichero de entrada están separados por un espacio en blanco y que cuando se escriba en el fichero de salida también deberá haber un espacio detrás de cada elemento que se escriba. No hay que preocuparse por los espacios en blanco al final de la línea, detrás del último elemento.

El fichero de salida mantendrá los saltos de línea del fichero de entrada.

En algunos casos el texto que identifica el hueco está compuesto por varias palabras separadas por guiones, como <adjetivo-masculino>. El programa deberá convertir esos guiones en espacios en blanco. Por ejemplo:

Línea de fichero:	Imprimir por pantalla:
`Creo que es muy <adjetivo-masculino>`	`adjetivo masculino:`

Cuando se elige **ver un "Mad Lib"**, se pedirá el nombre del fichero y se mostrará por pantalla.

```
Ver un cuento:
Nombre del fichero que quieres leer: salida
Fichero no encontrado. Inténtalo otra vez
Nombre del fichero: salida.txt
En vacaciones quiero cantar
Creo que es muy colorado
Deberías probarlo

******* MENU *******
(C)rear un "Mad Lib"
(V)er un "Mad Lib"
(S)alir
Elija su opción: S

Agur
```

Cuando se pulse S, el programa acabará.

RECURSOS

Para realizar esta tarea puedes utilizar jGRASP o el editor VPL de Moodle que además te dirá si la salida del programa coincide con la propuesta.

INDICACIONES DE DESARROLLO

Realiza el programa por partes. Escribe el código poco a poco compilándolo y testeándolo frecuentemente.

> Lee datos desde un fichero y muéstralos. Esto te servirá para la opción crear un "Mad Lib" pero también para la opción ver un "Mad Lib" aunque todavía no los hayas creado.
> Cuando empieces con la opción crear un "Mad Lib", céntrate primero en leer el fichero, identificar los huecos y mostrarlos por consola. Luego añade la interactividad para que el usuario rellene esos huecos. Finalmente, escribe la historia en el fichero de salida.
> Al principio, puede venir bien utilizar nombres concretos para los ficheros y añadir println para poder ver lo que está pasando. Por ejemplo, puede estar bien mostrar por consola cada elemento que se lee del fichero para comprobar que lo estamos haciendo bien. Recuerda eliminar estos println de más antes de enviar la tarea.
> Es más fácil depurar el programa utilizando ficheros pequeños con pocos huecos. Para ello, dispones del fichero simple.txt.
> Por último, crea el menú para que se pueda elegir lo que se quiere hacer todas las veces que se quiera.

Utiliza métodos, parámetros y returns para estructurar el programa y evitar redundancias.

> Evita que un método se llama a sí mismo. Tendrás que usar un bucle para que se código se repita.
> Para conseguir la máxima nota, se utilizarán al menos 4 métodos con parámetros y al menos uno de ellos pasará un objeto como parámetro y otro devolverá un objeto como parámetro
> Esta vez se puede utilizar println en el main, siempre que el main esté bien estructurado y sea un resumen conciso del programa.
> También es correcto que el bucle del menú esté en el main, siempre que el código dentro de él sea corto y legible.
> Presta especial atención a las redundancias. Si tienes código que se repite, introdúcelo dentro de un método o un bucle para escribirlo una sola vez. Si tienes un método largo o incoherente, divídelo en partes más pequeñas.
> Debe existir un único objeto de tipo Scanner para la lectura de los datos que se requieran **por consola** y que para utilizarlo en cualquier otro punto del programa habrá que pasarlo como parámetro.

Lee todos los datos introducidos por consola utilizando el método **nextLine**.

Para resolver el programa necesitarás usar diferentes **métodos de la clase String**

Lee y escribe ficheros mediante objetos de la clase Scanner y PrintStream, pasándoles parámetros File.

> Para leer los ficheros, combina la lectura por líneas con la lectura elemento a elemento como hemos aprendido durante la UD.
> Para comprobar si los ficheros de entrada se pueden leer, se pueden usar métodos de la clase File.
> Utiliza la sentencia try-catch-finally para ver información de las excepciones que se puedan producir sin que el programa pare.

Ficheros de entrada incorrectos pueden generar **excepciones**. InputMismatchException indica que se está tratando de leer datos del tipo equivocado. NoSuchElementException suele ocurrir cuando se intenta leer un dato después haber llegado al final del fichero o de la línea.

> Si se produce una excepción, lee atentamente el mensaje visualizado para localizar la línea en la que se produce. Por ejemplo:
>
> Exception in thread "main" java.util.NoSuchElementException: No line found
>
> at java.util.Scanner.nextLine(Scanner.java:1516)
> at MadLibs.myMethodName(MadLibs.java:73)
> at MadLibs.main(MadLibs.java:20)

Una vez conocida la línea, puedes utilizar println o el debugger (la mariquita) para ver los valores de las diferentes variables y depurar el problema.

Guía de estilo: Da nombres significativos a métodos, variables y parámetros siguiendo además las reglas de Java. Utiliza correctamente la indentación y los espacios en blanco. Incluye cabeceras explicativas al principio del programa y al principio de cada uno de los métodos, así como junto a cualquier línea de código que lo pueda necesitar. En el caso de los métodos, como comentario, además de lo que hace indica los parámetros que utiliza y el valor que devuelve si es que existe. (Referencia)

Penaliza: Utilizar recursos que no se han trabajado en las UDs anteriores o actual. No cumplir con las correcciones e indicaciones descritas en las Tares evaluativas anteriores (feedback de la calificación).

INDICACIONES DE ENTREGA

El programa se escribirá directamente en el IDE proporcionado o se copiará en él una vez finalizado.

En la cabecera, además de la información que describe el programa, se añadirá el enlace a la evaluación realizada, un enlace a Google Drive o a Youtube según el formato elegido

CRITERIOS DE CALIFICACIÓN. TOTAL 10 PUNTOS

La tarea se evaluará siguiendo la siguiente rúbrica:

Funcionalidad (70%).

MUY BUENO (10PT)	Las opciones del menú funcionan y el usuario o usuaria puede elegir el nombre de los ficheros que quiere usar. En el caso de los ficheros de lectura, se pedirán nombres hasta que el nombre introducido sea válido. Se utilizará correctamente la sentencia try-catch-finally para evitar que el programa falle por una excepción producida cuando se intenta ver un "Mad Lib". El programa está bien estructurado y no hay redundancias. Se utilizan 4 métodos con paso de parámetros y al menos uno de ellos usa la sentencia return. Además a uno de ellos se le pasará un objeto como parámetro y otro devolverá un objeto como parámetro. Se utilizan los tipos de datos requeridos.
BUENO (7-9PT)	Las opciones del menú funcionan y el usuario o usuaria puede elegir el nombre de los ficheros que quiere usar. En el caso de los ficheros de lectura, se pedirán nombres hasta que el nombre introducido sea válido. El programa está bien estructurado y no hay redundancias. Se utilizan 3 métodos con paso de parámetros y al menos uno de ellos usa la sentencia return. Además a uno de ellos se le pasará un objeto como parámetro. Se utilizan los tipos de datos requeridos.
SATISFACTORIO (5-7PT)	Las opciones del menú funcionan y el programa crea correctamente un "Mad Lib" a partir del fichero "simple.txt" y lo guarda en "salida.txt". El programa podría estar mejor estructurado y tiene alguna redundancia.

Comentarios (15%).

Legibilidad (15%).

Con esta rúbrica se calculará la nota de vuestro programa y se comparará con la que habéis indicado en la autoevaluación.

Si la diferencia es menor de un punto, se calculará la nota media entre las dos.

Si la diferencia es mayor, se utilizará directamente la calificación de la profesora.

En los casos en los que falte la autoevaluación, se utilizará la calificación de la profesora con una penalización de un punto.

Ficheros requeridos

simple.txt

```
1  En vacaciones quiero <infinitivo>
2  Creo que es muy <adjetivo-masculino>
3  Deberias probarlo
```

tango.txt

```
1  El primer <sustantivo-masculino> para aprender a <infinitivo>
2  tango <adjetivo> es descubrir el mundo
3  de este fascinante <sustantivo-masculino>
4  Hoy en día es muy <adjetivo> lograr esto
5  viendo <sustantivo-masculino-plural> por Internet
6  de distintos bailarines de tango
```

VPL

Navega por la unidad

◄ PROG04 Tarea de aprendizaje 03. Escritura de ficheros

Ir a...

UD04 Encuesta valoración ►

Contacta con nosotros:

Dirección: Paseo de Ubarburu 39, Edificio EnerTIC of. 206 · Donostia San Sebastián
Telefono : 945 567 953
E-mail: info@birt.eus
Twitter: @Birt_LH