Técnico Superior en Desarrollo de Aplicaciones Web

Práctica: Excepciones, Ficheros y Colecciones MB

Lee bien todo lo que se indica en este enunciado y procura respetarlo.

Si tienes alguna duda o crees que hay alguna incoherencia, avisa al profesor para que te la solucione.

Normas de Realización

a) Los ejercicios deben realizarse en archivos .java separados . Cada archivo se nombrará con el nombre de la clase correspondiente. **Indica el modelo de practica que estás realizando como comentario en la primera línea de código de cada ejercicio.**

b) Puedes consultar tus apuntes y pero NO fuentes de internet para realizar el ejercicio.

c) No se pueden utilizar herramientas que generen código para completar la práctica. Si se detecta dicho uso, se considerará plagio.

d) Sólo se permite usar las herramientas del lenguaje que se han dado hasta este momento. (*Recuerda: yo evalúo competencias, no sólo conocimientos*).

e) **Se tendrá muy en cuenta la legibilidad del código**: sangrado de lineas correcto, uso de comentarios y nombre de variables adecuado.

Copia o plagio

Tal y como se indica en los estatutos del centro, está totalmente prohibido copiar código de otro/a compañero/a y/o de internet, incluidas herramientas que generan código. Si se detecta y se demuestra este tipo de comportamiento, la calificación de la actividad será de 0 puntos.

Entrega

El ejercicio puntuará para la evaluación práctica del 3er Trimestre de la asignatura.

- Al finalizar el ejercicio debes entregar en la plataforma un archivo comprimido en .zip .rar o.7z con todos los ficheros **.java** de los ejercicios que completes. Nombra ese archivo con tu primer apellido y con tu nombre: Ej: `Ramos_JoseFrancisco`.

- **Recuerda:** se puntúa lo que se entrega en la plataforma. Además, si un ejercicio no compila, ese ejercicio tendrá 0 puntos.

CAMPAÑA “No pongas triste a tu profesor”

Hay un ejemplo de ejecución al final de cada enunciado. **Respeta la salida de mensajes de cara al usuario como se indica.** Si no entiendes algo, pregunta. No te inventes nada.

Todos los archivos de texto que vayan a leerse o a crearse **deben estar en el mismo paquete que el archivo .java** correspondiente. Si se colocan en otro sitio, el ejercicio será incorrecto.

Si vas a utilizar algún método o herramienta que no se ha explicado en clase, **pregunta antes a tu profesor**. Si no lo haces, el ejercicio en el que lo uses quedará anulado (independientemente de si está correcto o no).

Ya que tu profesor se ha molestado en ponerte avisos como los anteriores por diversas partes de la práctica, no pongas triste a tu profe, LEE BIEN TODO y RESPETA exactamente lo que se te pide. **Si eso no ocurre y tu profe se pondrá triste y te penalizará con 0,5 puntos por cada elemento no respetado.**

Además, tu profe ha explicado en clase **cuándo y dónde se capturan y se propagan las excepciones**, verdad? Pues tu profe llorará muy fuerte si no respetas eso que se ha visto en clase mil veces y que se ha repetido en los videos. **Llorar fuerte implica perder de 1 a 2 puntos por fallo de este estilo**.

EJERCICIO 1 *(2 puntos)*

Nombra este archivo como T3P1E1.

Realiza un programa que pida al usuario dos números: L y N. Esos números deben ser enteros y mayores a 2. Si no se cumple alguna de esas condiciones, el número en cuestión debe volver a pedirse.

A continuación debe crear el archivo `numerosB.txt` y llenarlo de la siguiente forma:

- El archivo debe tener tantas líneas como indique el valor L.
- En cada linea deben aparecer separados por punto (.) tantos números aleatorios entre 30 y 200 como indique el valor N

Si el fichero se crea correctamente, debe indicarse por pantalla.

Ejemplo de ejecución:

```
Numero de lineas del fichero: -8
Valores mayores a 2!!
Numero de líneas del fichero: 3

Cantidad de números por linea: muchos
ERROR: debes introducir un número!!
Cantidad de números por linea: 5


El fichero numeros.txt se ha generado con éxito.
```

Un posible contenido del fichero `numerosB.txt` según el ejemplo anterior sería:

```
1   120.60.70.150.50
2   70.30.200.90.110
3   50.180.120.50.50
```

3 filas con 5 números por fila.

EJERCICIO 2 *(3 puntos)*

Nombra este archivo como: T3P1E2.

Una linea tiene los corchetes balanceados si en esa línea hay tantos corchetes de apertura '[' como de cierre ']'. *Nota: Aunque realmente esto no es así del todo, para nosotros es mas que suficiente.*

Haciendo uso del fichero `analisisB.txt` que se adjunta en el paquete de archivos necesarios, crea un programa que lea el fichero y muestre por pantalla si cada linea del fichero está balanceada o no.

Ejemplo de ejecución para el fichero que se muestra a continuación:

```
1   [([])]
2   Esto es [un ejemplo] correcto
3   Esto es [un ejemplo] incorrecto]
```

```
Linea balanceada.
Linea balanceada.
Linea NO balanceada.
```

Fijate que la primera y la segunda línea tienen el mismo numero de corchetes de apertura y de cierre, pero la última no los tiene.

Jaime, no sé hacer eso y me estoy agobiando, ¿puedo hacer algo aunque sea por la mitad de los puntos?

En caso de que no seas capaz de hacer lo que te pide el ejercicio, prueba lo siguiente: haz que el fichero `analisisB.txt` solo tenga una linea (borra el resto) y saca por pantalla si esa única línea está balanceada o no.

Jaime, voy de sobrad@ y quiero ganarme algún puntillo más, ¿puedo hacer alguna mejora al programa?

Esa es muy buena actitud!!! Lo único que debes hacer es cambiar la salida: ahora en vez de sacar la información por pantalla, escribela en un fichero llamado

`resultadosB.txt`

EJERCICIO 3 *(5 puntos)*

Hedack, el departamento de seguridad y hacking ético de Medac ha hablado con tu profesor de programación y este te ha recomendado como figura destacada de tu clase.

Así que Hedack confía en ti para que les programes una aplicación de encriptación de textos que puedan utilizar evitar que el resto de institutos de la provincia copien el buenísimo contenido de los apuntes oficiales.

Para ello se ha pensado en crear la clase **Codificador**.

Las propiedades de esa clase van a ser las siguientes (todas ellas privadas):

- `ruta`: cadena que guardará la ruta del texto a leer para codificarlo.
- `textoOriginal`: cadena que guardará el texto que se lea del fichero indicado en la ruta.
- `textoCodificado`: cadena que guardará el texto codificado una vez se le aplique el algoritmo que se explica más adelante.

No se pueden añadir más propiedades a la clase. En caso de estar en desacuerdo, pregunta a tu profesor y justificale claramente el añadido que quieres hacer.

Los métodos de esta clase son los siguientes:

- `constructor(String)`: recibe la ruta completa del archivo a codificar. El resto de propiedades se inicializarán como cadenas vacías.
- `public string leerTexto()`: este método lee el contenido del fichero indicado en la propiedad `ruta` y vuelca su contenido a la propiedad `textoOriginal`. Devuelve todo el texto leído.
- `public string codificar()`: este método toma el contenido la propiedad `textoOriginal`, mira si está vacío o no y, en caso de que no lo esté, lo codifica usando los pasos que se indican mas adelante. A continuación, almacena esa transformación en la propiedad `textoCodificado`. Devuelve todo el texto codificado.
- `public void volcarCodificacion()`: este método toma el contenido de la propiedad `textoCodificado` y, si no está vacía, lo vuelca en un fichero nuevo. El nombre del fichero nuevo será el mismo que el fichero abierto para leer pero cambiando la extensión a .cod

Ejemplo:

*Imagina que el fichero que contiene el texto a codificar se llama `poema.txt` Pues el nuevo fichero con el texto codificado se llamará **`poema.cod`** y se guardará en la misma ruta que `poema.txt`*

IMPORTANTE: respeta el nombre, los parámetros y la visibilidad de cada método aquí explicado. Si no haces eso en algún método, ese método será calificado con 0 puntos.

Puedes añadir métodos propios para apoyarte siempre y cuando sean de visibilidad privada.

Cómo se codifica un texto:

La tabla Unicode es un estándar internacional de codificación de caracteres que asigna un número único a cada carácter. Así, por ejemplo, para los siguientes caracteres tenemos sus números asociados:

- A: 65, B: 66, C: 67, … Z:90
- a: 97, b:98, c: 99, … z:122
- 0: 48, 1: 49, 2: 50, ...9: 57
- Salto de línea: 10

La tabla Unicode permite que los caracteres sean representados de manera consistente en diferentes sistemas informáticos y lenguajes de programación, incluido JAVA.

Para codificar el texto tan solo hay que seguir los siguientes pasos:

- Se codifica carácter a carácter.
- Los saltos de línea se mantienen y no reciben ningún cambio.
- Las letras mayúsculas se mueven tres valores hacia adelante en la tabla unicode: la 'A' pasa a ser la 'D', la 'B' pasa a ser la 'E', ...
- Las letras minúsculas se mueven dos valores hacia adelante en la tabla unicode: la 'a' pasa a ser la 'c', la 'b' pasa a ser la 'd', ...
- Todos los dígitos que aparezcan deben moverse once valores hacia atrás en la tabla unicode.
- Cualquier otro carácter que aparezca en el texto original será movido doce posiciones hacia adelante en la tabla unicode. Pej, si aparece el carácter 64 (@) en el texto original, será cambiado por el carácter 76 (la L)

Para no complicar el ejercicio, no pasa nada si nos salimos de la tabla de caracteres visibles (tan solo saldrá un cuadradito blanco al codificar ese carácter).

Antes de agobiarte, recuerda lo siguiente:

- Los caracteres son en realidad números enteros.

- Avanzar en la tabla es sumar.
- Retroceder en la tabla es restar.
- Como los caracteres son números, dentro de un if puedes hacer cosas como estas:

```
char letra = ...
if((letra>= 'A') && (letra<= 'Z')){
   //la letra va a estar entre la A y la Z mayúsculas
}
```

Ejemplo:

Archivo poema.txt	Archivo poema.cod
En el aula, con paso diligente, El Señor Hormiga, maestro valiente, Explora el vasto mundo de la programación, Con paciencia y dedicación. Sus palabras, como gotas de miel, Fluyen con suavidad, sin papel, Intentando llegar al corazón, De cada alumno, sin excepción. Con la mente clara, y el alma en calma, Explica cada línea, como una palma, Intentando simplificar el código, Para que cada estudiante se sienta a bordo.	Hp,gn,cwnc8,eqp,rcuq,fknkigpvg8 Hn,Vgýqt,Kqtokic8,ocguvtq,xcnkgpvg8 Hzrnqtc,gn,xcuvq,owpfq,fg,nc,rtqitcocekÿp8 Fqp,rcekgpekc,{,fgfkecekÿp: Vwu,rcncdtcu8,eqoq,iqvcu,fg,okgn8 Inw{gp,eqp,uwcxkfcf8,ukp,rcrgn8 Lpvgpvcpfq,nngict,cn,eqtc|ÿp8 Gg,ecfc,cnwopq8,ukp,gzegrekÿp: Fqp,nc,ogpvg,enctc8,{,gn,cnoc,gp,ecnoc8 Hzrnkec,ecfc,nùpgc8,eqoq,wpc,rcnoc8 Lpvgpvcpfq,ukornkhkect,gn,eÿfkiq8 Sctc,swg,ecfc,guvwfkcpvg,ug,ukgpvc,c,dqtfq: