FAMILIA PROFESIONAL:	**Informática y Comunicaciones**
CICLO FORMATIVO:	**Desarrollo de Aplicaciones Multiplataforma**
MÓDULO:	**Acceso a Datos**

UNIDAD 1:
FICHEROS

ACTIVIDADES 1

AUTORES: **Fernando Rodríguez Alonso**
Sonia Pasamar Franco

FICHEROS DE TEXTO CON ORGANIZACIÓN SECUENCIAL 1

ACTIVIDAD 1x01

Codifica un programa principal **`main`** que:

- Lea un fichero de texto denominado `entrada.txt`.
- Cree otro fichero de texto denominado `salida.txt` y escriba en éste las mismas líneas del fichero de entrada pero en orden inverso.

Un ejemplo de ejecución del programa podría ser:

```
Fichero entrada.txt:
Este es el principio
Hola
Este es un fichero de texto
Adiós
Este es el fin
```

```
Fichero salida.txt:
Este es el fin
Adiós
Este es un fichero de texto
Hola
Este es el principio
```

ACTIVIDAD 1x02

Codifica un programa principal **`main`** que:

- Lea por teclado un nombre de fichero de texto.
- Si el nombre no se corresponde con ningún archivo (fichero, directorio o enlace) del sistema de archivos, escribirá en consola el mensaje:

  ```
  El archivo no existe en el sistema de archivos.
  ```

- Si el nombre se corresponde con un archivo del sistema de archivos pero no es un fichero, escribirá en consola el mensaje:

  ```
  El archivo no es un fichero.
  ```

- Si el nombre se corresponde con un fichero del sistema de archivos, leerá todas las líneas del fichero de texto indicado y creará un fichero de texto denominado `estadisticas.txt` que contenga lo siguiente:
  - El número de caracteres y el número de palabras de cada línea del fichero de texto.
  - El número total de líneas, el número total de caracteres y el número total de palabras del fichero de texto.

Un ejemplo de ejecución del programa podría ser:

```
Fichero lineas.txt:
Dos más tres son cinco.
Once menos uno son diez.
Siete por dos son catorce.
Quince entre dos son siete y medio.
```

```
Fichero estadisticas.txt:
Línea 1: 23 caracteres y 5 palabras
Línea 2: 24 caracteres y 5 palabras
Línea 3: 26 caracteres y 5 palabras
Línea 4: 35 caracteres y 7 palabras
Total: 4 líneas, 108 caracteres y 22 palabras
```

ACTIVIDAD 1x03

Codifica un programa principal **`main`** que gestione un fichero de texto denominado `frases.txt` mediante el siguiente menú de opciones:

0) Salir del programa.

1) Escribir varias frases en el fichero de texto.

Leerá por teclado de forma repetitiva frases y las escribirá en el fichero de texto de forma que su contenido previo no se sobrescriba. Este proceso deberá repetirse hasta que se introduzca la cadena ***, la cual no deberá escribirse en el fichero de texto.

2) Leer todas las frases del fichero de texto.

Leerá todas las frases del fichero de texto y las escribirá en consola en el mismo orden.

3) Consultar el número de líneas y el número de palabras del fichero de texto.

Calculará y escribirá en consola el número de líneas del fichero de texto.
Calculará y escribirá en consola el número de palabras del fichero de texto.

4) Buscar una palabra en el fichero de texto.

Leerá por teclado una palabra (que no tenga ningún espacio en blanco). Si la palabra no cumple la condición requerida, escribirá en consola un mensaje de error y leerá por teclado otra palabra. Este proceso se repetirá hasta que se introduzca una palabra válida.
Leerá las frases del fichero de texto para buscar la palabra en ellas, con dos situaciones posibles:

- Si la palabra está contenida en alguna de las frases del fichero de texto, escribirá en consola un mensaje indicando la línea en la que aparece por primera vez en el fichero de texto:
  `La palabra está en la línea 5 del fichero de texto.`
- Si la palabra no está contenida en ninguna de las frases del fichero de texto, escribirá en consola el mensaje:
  `La palabra no está en el fichero de texto.`

5) Crear otro fichero de texto con las frases del fichero de texto convertidas a mayúsculas.

Creará un fichero de texto denominado `mayusculas.txt`.
Leerá cada frase del fichero de texto `frases.txt`, la convertirá a letras mayúsculas y escribirá esta frase convertida en el fichero `mayusculas.txt`. La escritura de estas frases convertidas en el fichero `mayusculas.txt` deberá empezar por el principio del fichero de texto, de forma que se sobrescriba su contenido previo.

6) **Crear otro fichero de texto con las frases del fichero de texto convertidas a minúsculas.**

   Creará un fichero de texto denominado `minusculas.txt`.
   Leerá cada frase del fichero de texto `frases.txt`, la convertirá a letras minúsculas y escribirá esta frase convertida en el fichero `minusculas.txt`. La escritura de estas frases convertidas en el fichero `minusculas.txt` deberá empezar por el principio del fichero de texto, de forma que se sobrescriba su contenido previo.

Este programa principal deberá validar que la opción de menú elegida sea válida (comprendida entre 0 y 6). Si no lo es, escribirá en consola el mensaje:

```
La opción de menú debe estar comprendida entre 0 y 6.
```

FICHEROS DE TEXTO CON ORGANIZACIÓN SECUENCIAL 2

ACTIVIDAD 2x01

Codifica una clase **`Departamento`** para tratar la información de los diferentes departamentos de una empresa. De cada uno de ellos se desea guardar: un **código**, el **nombre** y la **ubicación**.

El código de cada departamento será un número entero positivo identificativo. Cada departamento tendrá su propio código, es decir, no podrá haber dos o más departamentos con el mismo código en la empresa.

Para esta clase **`Departamento`**:

- Codifica un constructor que reciba como parámetros el código, el nombre y la ubicación.
- Codifica otro constructor que reciba como parámetro una línea de texto, que extraiga varias subcadenas de esta línea de texto utilizando el carácter `';'` como separador y que inicialice los atributos del objeto con estas subcadenas generadas. Según el atributo, además podría ser necesario convertir una subcadena a número entero.
- Codifica el método de objeto **`toString`** para que devuelva una cadena de texto con el nombre de la clase y un resumen con los nombres y los valores de todos los atributos del objeto. Este método se utilizará para escribir un departamento en consola.
- Codifica otro método de objeto **`toStringWithSeparators`** que devuelva una cadena de texto con los valores de todos los atributos del objeto, separados entre sí por el carácter `';'`. Este método se utilizará para escribir un departamento como una línea en un fichero de texto.

Codifica una clase **`AccesoDepartamento`** que incluya las operaciones de mantenimiento necesarias sobre un fichero de texto `departamentos.txt`:

- El acceso al fichero de texto se realizará de modo **secuencial** mediante **flujos de caracteres** que manejen **líneas de texto**.
- Las operaciones de acceso al fichero de texto deberán cerrar, en cualquier caso, los flujos de caracteres utilizados, tanto si una operación se ejecuta con normalidad como si se produce algún error de lectura o escritura.
- Las operaciones de acceso al fichero de texto deberán propagar aquellas excepciones que haya que tratar de forma obligatoria.
- Un ejemplo del contenido de este fichero de texto, con 5 departamentos, podría ser:

```
1;Dirección;Planta3
2;Contabilidad;Planta2
3;Recursos Humanos;Planta2
4;Compras;Planta1
5;Ventas;Planta1
```

Codifica una clase **`Actividad_2x01`** que incluya un programa principal **`main`**. Este programa gestionará el fichero de texto `departamentos.txt` mediante el siguiente menú de opciones:

0) **Salir del programa.**

1) **Insertar un departamento en el fichero de texto.**

   Leerá por teclado el código, el nombre y la ubicación del departamento a insertar.
   Si el fichero `departamentos.txt` contiene un departamento con el código dado, escribirá en consola el mensaje:

   ```
   Ya existe otro departamento con ese código en el fichero de texto.
   ```

   Si el fichero `departamentos.txt` no contiene ningún departamento con el código dado, insertará el departamento al final del fichero de texto y escribirá en consola el mensaje:

   ```
   Se ha insertado un departamento en el fichero de texto.
   ```

2) **Consultar todos los departamentos del fichero de texto.**

   Si el fichero `departamentos.txt` no contiene ningún departamento, escribirá en consola el mensaje:

   ```
   El fichero de texto no tiene ningún departamento.
   ```

   Si el fichero `departamentos.txt` contiene uno o más departamentos:
   - Leerá todos los departamentos del fichero de texto.
   - Escribirá en consola un listado de todos los departamentos almacenados en el fichero de texto, indicando para cada departamento, un resumen con los nombres y los valores de todos los atributos de dicho departamento.
   - Escribirá en consola el número de departamentos consultados del fichero de texto.

   Por ejemplo:

   ```
   Departamento [Código = 1, Nombre = Dirección, Ubicación = Planta3]
   Departamento [Código = 2, Nombre = Contabilidad, Ubicación = Planta2]
   Departamento [Código = 3,
                 Nombre = Recursos Humanos, Ubicación = Planta2]
   Departamento [Código = 4, Nombre = Compras, Ubicación = Planta1]
   Departamento [Código = 5, Nombre = Ventas, Ubicación = Planta1]
   Se han consultado 5 departamentos del fichero de texto.
   ```

3) **Consultar un departamento, por código, del fichero de texto.**

   Leerá por teclado el código del departamento a consultar.
   Si el fichero `departamentos.txt` no contiene ningún departamento con el código dado, escribirá en consola el mensaje:

   ```
   No existe ningún departamento con ese código en el fichero de texto.
   ```

   Si el fichero `departamentos.txt` contiene un departamento con el código dado, consultará el departamento del fichero de texto y escribirá en consola un resumen con los nombres y los valores de todos los atributos de dicho departamento.
   El siguiente ejemplo muestra el departamento cuyo código es 3:

   ```
   Departamento [Código = 3,
                 Nombre = Recursos Humanos, Ubicación = Planta2]
   ```

4) **Actualizar un departamento, por código, del fichero de texto.**

Leerá por teclado el código, el nuevo nombre y la nueva ubicación del departamento a actualizar.
Si el fichero `departamentos.txt` no contiene ningún departamento con el código dado, escribirá en consola el mensaje:

```
No existe ningún departamento con ese código en el fichero de texto.
```

Si el fichero `departamentos.txt` contiene un departamento con el código dado, actualizará el departamento del fichero de texto con el nuevo nombre y la nueva ubicación y escribirá en consola el mensaje:

```
Se ha actualizado un departamento del fichero de texto.
```

Para realizar esta operación de actualización, será necesario utilizar una colección de objetos auxiliar o un fichero de texto auxiliar.

5) **Eliminar un departamento, por código, del fichero de texto.**

Leerá por teclado el código del departamento a eliminar.
Si el fichero `departamentos.txt` no contiene ningún departamento con el código dado, escribirá en consola el mensaje:

```
No existe ningún departamento con ese código en el fichero de texto.
```

Si el fichero `departamentos.txt` contiene un departamento con el código dado y el fichero `empleados.txt` contiene al menos un empleado referenciado a este departamento, escribirá en consola el mensaje:

```
Existe al menos un empleado referenciado a ese departamento.
```

Si el fichero `departamentos.txt` contiene un departamento con el código dado y el fichero `empleados.txt` no contiene ningún empleado referenciado a este departamento, eliminará el departamento del fichero de texto y escribirá en consola el mensaje:

```
Se ha eliminado un departamento del fichero de texto.
```

Para realizar esta operación de eliminación, será necesario utilizar una colección de objetos auxiliar o un fichero de texto auxiliar.

Este programa principal deberá validar que la opción de menú elegida sea válida (comprendida entre 0 y 5). Si no lo es, escribirá en consola el mensaje:

```
La opción de menú debe estar comprendida entre 0 y 5.
```

Este programa principal deberá tratar cualquier excepción que pueda ocurrir en cada opción de menú, causada por un error de entrada/salida al abrir, escribir, leer o cerrar el fichero de texto. El tratamiento de cada excepción consistirá en capturarla, escribir en consola un mensaje con el tipo de la excepción y escribir en consola otro mensaje con la causa de la excepción.

ACTIVIDAD 2x02

Codifica una clase **`Empleado`** para tratar la información de los diferentes empleados de una empresa. De cada uno de ellos se desea guardar: un **código**, el **código de departamento**, el **nombre**, la **fecha de alta** y el **salario** en euros.

El código de cada empleado será un número entero positivo identificativo. Cada empleado tendrá su propio código, es decir, no podrá haber dos o más empleados con el mismo código en la empresa.

El código de departamento de cada empleado hace referencia al departamento en el que dicho empleado trabaja. Cada empleado deberá pertenecer a un departamento, es decir, no podrá haber un empleado con el código de departamento sin asignar o referenciando un departamento inexistente.

Para esta clase **`Empleado`**:

- Codifica un constructor que reciba como parámetros el código, el código de departamento, el nombre, la fecha de alta y el salario.
- Codifica otro constructor que reciba como parámetro una línea de texto, que extraiga varias subcadenas de esta línea de texto utilizando el carácter `';'` como separador y que inicialice los atributos del objeto con estas subcadenas generadas. Según el atributo, además podría ser necesario convertir una subcadena a número entero o real.
- Codifica el método de objeto **`toString`** para que devuelva una cadena de texto con el nombre de la clase y un resumen con los nombres y los valores de todos los atributos del objeto. El salario se deberá indicar con 2 dígitos decimales. Este método se utilizará para escribir un empleado en consola.
- Codifica otro método de objeto **`toStringWithSeparators`** que devuelva una cadena de texto con los valores de todos los atributos del objeto, separados entre sí por el carácter `';'`. El salario se deberá indicar con 2 dígitos decimales y utilizando el carácter `','` para separar la parte entera de la parte fraccionaria. Este método se utilizará para escribir un empleado como una línea en un fichero de texto.

Codifica una clase **`AccesoEmpleado`** que incluya las operaciones de mantenimiento necesarias sobre un fichero de texto `empleados.txt`:

- El acceso al fichero de texto se realizará de modo **secuencial** mediante **flujos de caracteres** que manejen **líneas de texto**.
- Las operaciones de acceso al fichero de texto deberán cerrar, en cualquier caso, los flujos de caracteres utilizados, tanto si una operación se ejecuta con normalidad como si se produce algún error de lectura o escritura.
- Las operaciones de acceso al fichero de texto deberán propagar aquellas excepciones que haya que tratar de forma obligatoria.
- Un ejemplo del contenido de este fichero de texto, con 5 empleados, podría ser:

```
1;1;Alejandro Ruiz;03/05/1985;4025,77
2;2;María Latorre;27/11/2004;2010,05
3;3;Diego García;14/07/2005;1995,90
4;4;Isabel Hidalgo;01/09/1992;3105,82
5;5;Pilar Martínez;30/03/1993;3010,37
```

Codifica una clase **`Actividad_2x02`** que incluya un programa principal **`main`**. Este programa gestionará el fichero de texto `empleados.txt` mediante el siguiente menú de opciones:

0) Salir del programa.

1) Insertar un empleado en el fichero de texto.

Escribirá en consola un listado de todos los departamentos almacenados en el fichero `departamentos.txt`, indicando para cada departamento, un resumen con los nombres y los valores de todos los atributos de dicho departamento.
Leerá por teclado el código, el código de departamento, el nombre, la fecha de alta y el salario del empleado a insertar.
Si el fichero `empleados.txt` contiene un empleado con el código dado, escribirá en consola el mensaje:

```
Ya existe otro empleado con ese código en el fichero de texto.
```

Si el fichero `empleados.txt` no contiene ningún empleado con el código dado y el fichero `departamentos.txt` no contiene ningún departamento con el código de departamento dado, escribirá en consola el mensaje:

```
No existe ningún departamento con ese código en el fichero de texto.
```

Si el fichero `empleados.txt` no contiene ningún empleado con el código dado y el fichero `departamentos.txt` contiene un departamento con el código de departamento dado, insertará el empleado al final del fichero de texto y escribirá en consola el mensaje:

```
Se ha insertado un empleado en el fichero de texto.
```

2) Consultar todos los empleados del fichero de texto.

Si el fichero `empleados.txt` no contiene ningún empleado, visualizará en consola el mensaje:

```
El fichero de texto no tiene ningún empleado.
```

Si el fichero `empleados.txt` contiene uno o más empleados:

- Leerá todos los empleados del fichero de texto.
- Escribirá en consola un listado de todos los empleados almacenados en el fichero de texto, indicando para cada empleado, un resumen con los nombres y los valores de todos los atributos de dicho empleado.
- Escribirá en consola el número de empleados consultados del fichero de texto.

Por ejemplo:

```
Empleado [Código = 1, CódigoDepartamento = 1, Nombre = Alejandro Ruiz,
          FechaAlta = 03/05/1985, Salario = 4025,77 €]
Empleado [Código = 2, CódigoDepartamento = 2, Nombre = María Latorre,
          FechaAlta = 27/11/2004, Salario = 2010,05 €]
Empleado [Código = 3, CódigoDepartamento = 3, Nombre = Diego García,
          FechaAlta = 14/07/2005, Salario = 1995,90 €]
Empleado [Código = 4, CódigoDepartamento = 4, Nombre = Isabel Hidalgo,
          FechaAlta = 01/09/1992, Salario = 3105,82 €]
Empleado [Código = 5, CódigoDepartamento = 5, Nombre = Pilar Martínez,
          FechaAlta = 30/03/1993, Salario = 3010,37 €]
Se han consultado 5 empleados del fichero de texto.
```

3) **Consultar un empleado, por código, del fichero de texto.**

Leerá por teclado el código del empleado a consultar.
Si el fichero `empleados.txt` no contiene ningún empleado con el código dado, escribirá en consola el mensaje:

```
No existe ningún empleado con ese código en el fichero de texto.
```

Si el fichero `empleados.txt` contiene un empleado con el código dado, consultará el empleado del fichero de texto y escribirá en consola un resumen con los nombres y los valores de todos los atributos de dicho empleado.
El siguiente ejemplo muestra el empleado cuyo código es 5:

```
Empleado [Código = 5, CódigoDepartamento = 5, Nombre = Pilar Martínez,
          FechaAlta = 30/03/1993, Salario = 3010,37 €]
```

4) **Actualizar un empleado, por código, del fichero de texto.**

Escribirá en consola un listado de todos los departamentos almacenados en el fichero `departamentos.txt`, indicando para cada departamento, un resumen con los nombres y los valores de todos los atributos de dicho departamento.
Leerá por teclado el código, el nuevo código de departamento, el nuevo nombre, la nueva fecha de alta y el nuevo salario del empleado a actualizar.
Si el fichero `empleados.txt` no contiene ningún empleado con el código dado, escribirá en consola el mensaje:

```
No existe ningún empleado con ese código en el fichero de texto.
```

Si el fichero `empleados.txt` contiene un empleado con el código dado, actualizará el empleado del fichero de texto con el nuevo código de departamento, el nuevo nombre, la nueva fecha de alta y el nuevo salario y escribirá en consola el mensaje:

```
Se ha actualizado un empleado del fichero de texto.
```

Para realizar esta operación de actualización, será necesario utilizar una colección de objetos auxiliar o un fichero de texto auxiliar.

5) **Eliminar un empleado, por código, del fichero de texto.**

Leerá por teclado el código del empleado a eliminar.
Si el fichero `empleados.txt` no contiene ningún empleado con el código dado, escribirá en consola el mensaje:

```
No existe ningún empleado con ese código en el fichero de texto.
```

Si el fichero `empleados.txt` contiene un empleado con el código dado, eliminará el empleado del fichero de texto y escribirá en consola el mensaje:

```
Se ha eliminado un empleado del fichero de texto.
```

Para realizar esta operación de eliminación, será necesario utilizar una colección de objetos auxiliar o un fichero de texto auxiliar.

Este programa principal deberá validar que la opción de menú elegida sea válida (comprendida entre 0 y 5). Si no lo es, escribirá en consola el mensaje:

```
La opción de menú debe estar comprendida entre 0 y 5.
```

Este programa principal deberá tratar cualquier excepción que pueda ocurrir en cada opción de menú, causada por un error de entrada/salida al abrir, escribir, leer o cerrar el fichero de texto. El tratamiento de cada excepción consistirá en capturarla, escribir en consola un mensaje con el tipo de la excepción y escribir en consola otro mensaje con la causa de la excepción.