Ejercicios UT9 – Registros, ficheros y operaciones.

Programas en los que aparecen directorios y referencias a ficheros.

1. Escriba un programa en Java que solicite al usuario la ruta de un directorio existente en el sistema de archivos. El programa debe verificar si la ruta proporcionada corresponde a un directorio válido. Si es un directorio, el programa debe mostrar una lista de todos los archivos y subdirectorios que contiene. El programa debe indicar cuántos de ellos son archivos y cuántos son subdirectorios. Utilice el método isDirectory() para comprobar si es un directorio y el método listFiles() para obtener el contenido del directorio.

2. Escriba un programa en Java que permita realizar las siguientes acciones sobre un archivo: Crear un archivo llamado datos.txt en el directorio actual si no existe. Renombre el archivo a informacion.txt si ya existe. Elimine el archivo informacion.txt si el usuario lo solicita. Use createNewFile() para crear el archivo, renameTo() para renombrar y delete() para eliminar el archivo.

3. Cree un programa en Java que solicite al usuario la ruta de un directorio. El programa debe calcular y mostrar el tamaño total en bytes de todos los archivos contenidos dentro del directorio y sus subdirectorios. El programa debe verificar si la ruta proporcionada es válida y debe recorrer los subdirectorios de forma recursiva para incluir todos los archivos. Use el método isFile() para verificar si un elemento es un archivo y length() para obtener su tamaño. Aplique un método recursivo para recorrer subdirectorios.
4. Incorpore en el programa anterior un método que calcule el número de niveles que hay entre el directorio señalado y el directorio más profundo que parte de él.

Programas en los que interviene ficheros de texto

5. Genere una secuencia de 10 números reales aleatorios entre 0 y 100, y guarde dicha secuencia en un fichero llamado numeros.txt utilizando la clase **PrintWriter**. A continuación lea la secuencia de números desde el fichero usando la clase **Scanner** y calcule y muestre por pantalla: El número mayor, el número menor, la media de los números y el número más cercano a la media en valor absoluto.

6. Utilizando las clases **FileWriter** y **FileReader**, cree un programa que permita gestionar una lista de contactos. Cada contacto tiene un nombre, un número de teléfono y una dirección de correo electrónico. El programa debe comenzar leyendo un fichero de texto con al menos dos contactos que habrá creado previamente con un editor de texto plano (bloc de notas o similar). Para facilitar la demarcación de cada atributo en el fichero, separe cada uno de los campos por un punto y coma (;) y procure que cada contacto ocupe una línea diferente. Almacene los contactos del fichero en una colección y permita que el usuario añada nuevos contactos o visualice los existentes. Antes de finalizar el programa almacene todos los contactos en el fichero.

7. Desarrolle un programa en Java con las clases **BufferedReader** y **BufferedWriter** que realice las siguientes acciones sobre un fichero de texto generado con IA. El fichero de texto que debe generar la IA tendrá varios párrafos y expresamente se le indicará que lo genere con errores de puntuación. Concretamente habrá palabras después de un punto que no comenzarán por mayúsculas y palabras que no estarán precedidas de un espacio en blanco tras la ocurrencia de una coma. Lea el contenido del fichero y convierta a **mayúsculas** la primera letra de cada palabra que aparezca inmediatamente después de un punto y añada un **espacio en blanco** después de cada coma (si no está presente). Escriba el texto corregido en un nuevo fichero llamado Corregido.txt.

8. Cree un programa que permita encriptar y desencriptar el contenido de un fichero de texto utilizando el cifrado César. Dicho cifrado consiste en sustituir cada letra del texto por aquella que se encuentre desplazando un cierto número de lugares en el alfabeto. Por ejemplo, con un desplazamiento de 2, 'A' se reemplazaría por 'C', 'B' se convertiría en 'D'. Con desplazamiento 5, 'C' se reemplazaría por 'H', 'E' se convertiría en 'J', etc.

9. Desarrolle un programa que reciba 2 archivos de texto y combine el contenido de los 2 archivos. Para ello, se creará un nuevo archivo donde se debe añadir una palabra de cada archivo de forma consecutiva mientras queden palabras en cada uno de los archivos. Si algún archivo se queda sin palabras se deben añadir todas las palabras del otro archivo.

10. Construya con el bloc de notas un fichero llamado alumnos_notas.txt que contenga una lista de 10 alumnos y las notas que han obtenido en cada asignatura. El número de asignaturas de cada alumno será variable. Desarrolle un programa que muestre por pantalla la nota media de cada alumno junto a su nombre y apellido, ordenado alfabéticamente por nombre y apellido.

11. Desarrolle un programa que lea un documento de texto y muestre por pantalla algunos datos estadísticos: nº de caracteres, nº de palabras, nº de líneas. Además, el programa debe determinar cuál es la palabra o palabras (si hay varias que se repitan el mismo número de veces) más frecuente, indicando el número de veces que aparece. Utilice para este propósito un HashMap para la cuenta de las palabras.

12. Escriba un programa capaz de quitar los comentarios presentes en un programa escrito en lenguaje Java.

Programas en los que intervienen archivos binarios

13. Elabore un programa capaz de gestionar la puntuación final de los jugadores en un torneo de videojuegos. Cada jugador tiene una puntuación que debe ser almacenada en un archivo binario. Elabore un programa en Java que permita realizar las siguientes acciones: **Registrar puntuaciones:** Solicita al usuario que introduzca los nombres de los jugadores y sus respectivas puntuaciones (un número entero) y guarda esta información en un archivo binario. **Mostrar puntuaciones:** Lee las puntuaciones almacenadas en el archivo binario y muestra una tabla con el nombre del jugador y su puntaje. **Actualizar la puntuación de un jugador:** Permite al usuario buscar a un

jugador por su nombre y actualizar su puntaje en el archivo binario. Utilice las clases **FileOutputStream** y **DataOutputStream** para escribir en el archivo binario. Utilice las clases **FileInputStream** y **DataInputStream** para leer del archivo binario. Contemple el manejo básico de excepciones como IOException. Asegúrese de crear el archivo automáticamente si no existe.

14. Cree un programa en Java que permita la gestión de un archivo binario para almacenar información sobre una lista de productos. Cada producto debe tener los siguientes atributos: Código del producto (entero). Nombre del producto (cadena de texto). Precio (decimal). Cantidad en stock (entero). El programa debe proporcionar las siguientes funcionalidades: **Agregar productos:** Permite al usuario ingresar los datos de un nuevo producto y añadirlo al archivo binario. **Listar productos:** Lee todos los productos almacenados en el archivo binario y los muestra en la consola. **Buscar un producto:** Permite al usuario buscar un producto por su código y mostrar sus datos si existe. **Actualizar stock:** Permite modificar la cantidad en stock de un producto específico, identificándolo por su código. **Eliminar un producto:** Elimina un producto del archivo binario identificándolo por su código. Utilice las clases **FileInputStream** y **FileOutputStream** junto con **ObjectInputStream** y **ObjectOutputStream** para manejar los ficheros binarios. Recuerde que la clase Producto debe ser Serializable.

Programas en los que intervienen archivos con formato: JSON, XML, XLS, PDF

15. En el aula virtual tiene un proyecto Java que permite mostrar en un mapa una ubicación geolocalizada. En el interior del proyecto hay un fichero con formato .csv que contiene los datos de los aparcamientos públicos de la ciudad de Madrid. Adapte el proyecto para que muestre en el mapa, todos (o al menos algunos de ellos) de los aparcamientos geolocalizados.

16. Genere datos con el apoyo de la IA y construya con ellos un archivo Excel (empleados.xls) que contenga en su primera hoja información sobre los empleados de una empresa de acuerdo con el modelo siguiente:

ID Empleado	Nombre	Departamento	Salario (€)
100	María López	Ventas	2500
101	Juan Pérez	Recursos Humanos	3000
102	Laura Gómez	IT	3200

El programa debe incluir las funcionalidades siguientes: **Cargar y procesar el archivo Excel** (Utilice la biblioteca **Apache POI** para leer el archivo empleados.xls). Extraer los datos de la primera hoja del archivo y con ellos imprimir en la consola la tabla completa con los datos de los empleados. Calcular el salario total de los empleados. Encontrar el empleado con el salario más alto y más bajo. Crear un archivo de texto (reporte_empleados.txt) que contenga un resumen con el número total de empleados, el salario promedio y el nombre del empleado mejor pagado (podrían ser varios).

17. Desarrolle un programa en Java que consuma datos JSON procedentes del portal de datos abiertos de Renfe de avisos sobre modificaciones planificadas del servicio que afectan a la circulación de trenes de viajeros. Para descargar los datos en formato JSON puede acceder a la URL:

    https://data.renfe.com/dataset?res_format=JSON&tags=avisos

    Tras localizar el recurso en formato JSON analice su contenido para conocer el significado de cada uno de los campos y guarde el contenido JSON descargado en un archivo local llamado incidencias_circulacion.json.

    Finalmente desarrolle un programa que permita mostrar las incidencias filtrando las mismas por palabras clave del campo "paragraph" y por CCAA del campo "tags". Utilice una biblioteca popular como Gson o Jackson para procesar ficheros JSON.

    Elabore un gráfico que refleje el número de incidencias por CCAA. Utilice la biblioteca JFreeChart disponible en este enlace: https://www.jfree.org/jfreechart/

18. Desarrolle un programa en Java para la gestión de una biblioteca virtual. El programa debe permitir el manejo de información sobre libros utilizando ficheros binarios y XML para diferentes propósitos. Cada libro debe tener los siguientes atributos: **ISBN** (cadena de texto – atributo clave principal). **Título** (cadena de texto). **Autor** (cadena de texto). **Año de publicación** (entero). **Número de páginas** (entero). El programa debe contemplar a través de un menú de opciones las operaciones básicas de un CRUD: Alta de nuevos libros, bajas de libros existentes, consulta de libros por al menos dos criterios: título y por autor, listado de libros y procesos de exportación e importación de datos. Además, debe tener en cuenta las especificaciones siguientes:

Almacenar libros en un archivo binario: Permite al usuario añadir libros a la biblioteca y guardar toda la colección en un archivo binario. Utilice **ObjectOutputStream** para escribir en el archivo y **ObjectInputStream** para leer.

Exportar libros a un archivo XML: Genera un archivo XML con la lista completa de libros. Utiliza una biblioteca de Java para trabajar con XML, como javax.xml.parsers y org.w3c.dom. Asegúrese de estructurar el XML con elementos como <biblioteca>, <libro>, <titulo>, <autor>, etc.

Generar los códigos de barras de los libros de la biblioteca. Permite generar los códigos de barras asociados a los ISBN de los libros de la biblioteca. Utilice la biblioteca Barcode4J. Tiene un ejemplo sencillo de uso en este enlace:

https://dar10comyr.blogspot.com/2017/06/metodos-generar-codigos-de-barras-y.html

Listar el catálogo de la biblioteca en un fichero pdf. Utilice la biblioteca iText-5.0.1. Dispone de un ejemplo en este enlace:

https://chuidiang.org/index.php?title=Ejemplo_sencillo_de_creaci%C3%B3n_de_un_pdf_con_iText

Incorpore en el programa la internacionalización de mensajes. Para ello cree un directorio específico para los archivos de propiedades que contengan los mensajes traducidos. Por ejemplo:

```
resources/
        |-- messages_en.properties
        |-- messages_es.properties
        |-- messages_fr.properties
```

Cada archivo `.properties` contendrá las cadenas en un idioma específico. Por ejemplo:

messages_en.properties	**messages_es.properties**	**messages_fs.properties**
greeting=Hello farewell=Goodbye	greeting=Hola farewell=Adiós	greeting=Bonjour farewell=Au revoir

Para cargar los recursos utilice la clase `ResourceBundle` de Java que le permite gestionar los mensajes basados en el idioma seleccionado. A continuación se incluye un ejemplo sencillo.

```java
import java.util.Locale;
import java.util.ResourceBundle;

public class App
{
 public static void main(String[] args)
 {
    Locale locale;
    ResourceBundle messages;
    try
    {
       // Configurar el idioma deseado (por ejemplo, español)
       locale = new Locale("es", "ES"); // Idioma y país
       messages= ResourceBundle.getBundle("messages", locale);
    }
    catch (Exception e)
    {
       System.out.println("Idioma no soportado. Inglés por defecto.");
       locale = new Locale("en", "US"); // Idioma y país
       messages = ResourceBundle.getBundle("messages", locale);
    }

    // Obtener mensajes traducidos
    System.out.println(messages.getString("greeting")); //Hola
    System.out.println(messages.getString("farewell")); //Adiós
 }

 public static ResourceBundle loadMessages(String languageCode)
 {
    Locale locale = new Locale(languageCode);
    return ResourceBundle.getBundle("messages", locale);
 }
}
```