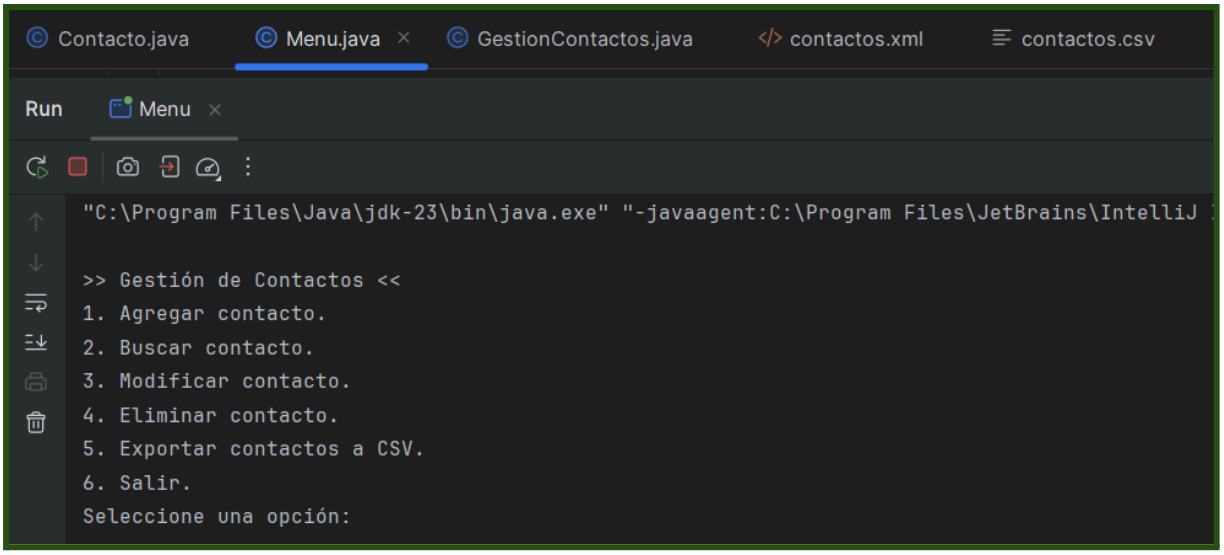
~ Gestión de Contactos en Ficheros XML ~

Ivan David Velazquez Aguilar

Desarrollo de Aplicaciones Multiplataforma

Curso 2024 / 2025

Acceso a Datos



ÍNDICE.

1. Descripción General.

La aplicación de gestión de contactos se encarga de almacenar, buscar, modificar, eliminar y exportar una lista de contactos. Los datos se almacenan en un archivo XML (contactos.xml) y, adicionalmente, se proporciona la opción de exportar la lista a un archivo CSV (contactos.csv). La aplicación se desarrolla en Java y utiliza las bibliotecas estándar para manejar archivos XML y CSV. La interfaz principal es una aplicación de consola, donde el usuario interactúa mediante un menú de opciones.

2. Estructura de la Aplicación.

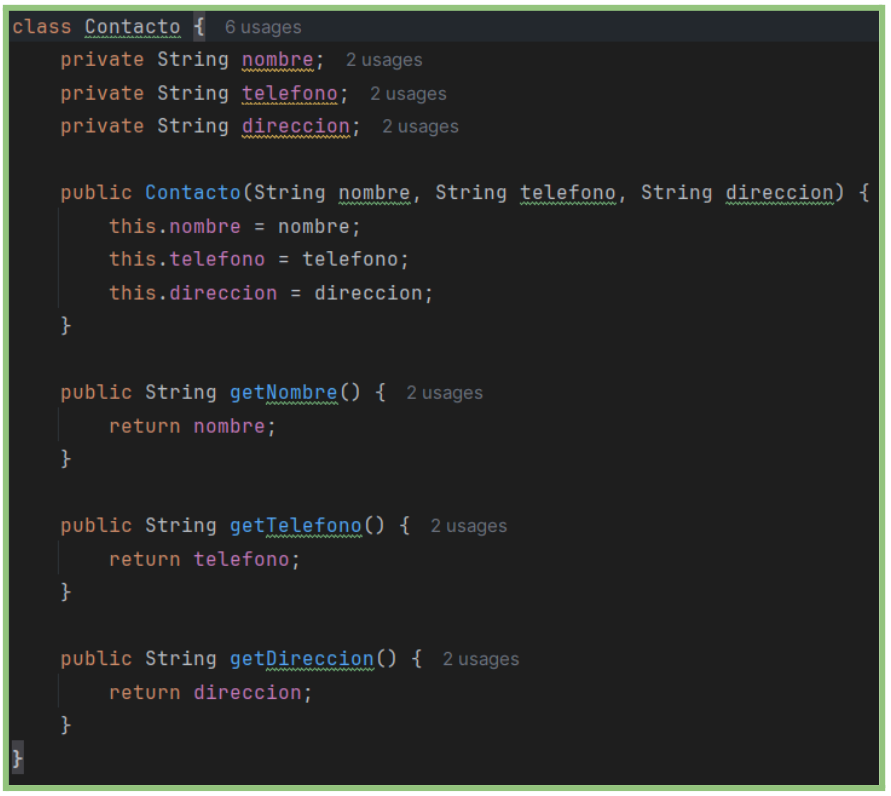
La aplicación consta de las siguientes partes principales.

2.1. Clases y Componentes.

- ***Clase GestionContactos:*** Esta clase contiene todos los métodos necesarios para realizar operaciones sobre los contactos, como ***agregar***, ***buscar***, ***modificar***, ***eliminar*** y ***exportar*** a CSV. Esta clase maneja la lógica de interacción con el archivo XML.

```
public class GestionContactos {  2 usages

    private static final String ARCHIVO_XML_PATH = "contactos.xml";  5 usages
    private static final String ARCHIVO_CSV_PATH = "contactos.csv";  2 usages

    // Método para agregar un nuevo contacto al fichero XML
    public void agregarContacto(Contacto contacto) {...}

    // Método para buscar un contacto por nombre
    public void buscarContacto(String nombreBuscado) {...}

    // Método para modificar un contacto por nombre
    public void modificarContacto(String nombreBuscado, Contacto nuevosDatos) {...}

    // Método para eliminar un contacto por nombre
    public void eliminarContacto(String nombreBuscado) {...}

    // Método para exportar contactos a CSV
    public void exportarContactosACSV() {...}

}
```

- ***Clase Contacto:*** Representa un contacto individual. Contiene atributos como nombre, teléfono y dirección, junto con sus métodos de acceso (getters).



```
class Contacto {  6 usages
    private String nombre;  2 usages
    private String telefono;  2 usages
    private String direccion;  2 usages

    public Contacto(String nombre, String telefono, String direccion) {
        this.nombre = nombre;
        this.telefono = telefono;
        this.direccion = direccion;
    }

    public String getNombre() {  2 usages
        return nombre;
    }

    public String getTelefono() {  2 usages
        return telefono;
    }

    public String getDireccion() {  2 usages
        return direccion;
    }
}
```

Y una clase Menu, que es donde se ejecuta (main) el programa y proporciona una interfaz de terminal para el usuario.

2.2. Métodos de GestionContactos.

1. ***agregarContacto****(Contacto contacto):* Agrega un nuevo contacto al archivo XML. Si el archivo no existe, se crea uno nuevo, con una raíz denominada `<contactos>`. Luego se agrega el nuevo contacto con sus elementos hijos (`<nombre>, <telefono>, <direccion>`).
2. ***buscarContacto****(String nombreBuscado):* Busca uno o varios contactos por su nombre en el archivo XML. En caso de que existan múltiples contactos con el mismo nombre, se muestran todos los resultados.
3. ***modificarContacto****(String nombreBuscado, Contacto nuevosDatos):* Permite modificar un contacto existente en el archivo XML. Si existen múltiples contactos con el mismo nombre, el usuario debe seleccionar cuál desea modificar.
4. ***eliminarContacto****(String nombreBuscado):* Elimina un contacto del archivo XML. Si existen varios contactos con el mismo nombre, el usuario selecciona cuál eliminar.

5. ***exportarContactosACSV()*:** Exporta todos los contactos almacenados en el archivo XML a un archivo CSV. Esto permite que los contactos se visualicen en otras aplicaciones, como hojas de cálculo.

2.3. Menú Principal.

El menú principal permite al usuario realizar las siguientes acciones:

- `Agregar contacto.`
- `Buscar contacto.`
- `Modificar contacto.`
- `Eliminar contacto.`
- `Exportar contactos a CSV.`
- `Salir.`

Cada opción está diseñada para ser intuitiva y permitir una interacción sencilla por parte del usuario mediante la consola.

```
System.out.println("\n>> Gestión de Contactos <<");
System.out.println("1. Agregar contacto.");
System.out.println("2. Buscar contacto.");
System.out.println("3. Modificar contacto.");
System.out.println("4. Eliminar contacto.");
System.out.println("5. Exportar contactos a CSV.");
System.out.println("6. Salir.");
System.out.print("Seleccione una opción: ");
opcion = sc.nextInt();
```

3. Interacción con el Archivo XML.

La aplicación utiliza las clases de Java ***`DocumentBuilder`***, ***`Document`***, y ***`Transformer`*** para gestionar la lectura, escritura, y modificación del archivo XML.

- **Creación y Escritura:** Cuando se agrega un nuevo contacto, se utiliza DocumentBuilder para crear el archivo XML si no existe o cargarlo si ya está presente. Los nuevos contactos se agregan al elemento raíz *<contactos>*, y luego se utiliza Transformer para guardar los cambios en el archivo.

```
File file = new File(ARCHIVO_XML_PATH);
DocumentBuilderFactory dbFactory = DocumentBuilderFactory.newInstance();
DocumentBuilder dBuilder = dbFactory.newDocumentBuilder();
Document doc;

// Si el archivo ya existe, cargar el documento existente
if (file.exists()) {
    doc = dBuilder.parse(file);
    doc.getDocumentElement().normalize();
} else {
    // Si no existe, crear un nuevo documento XML
    doc = dBuilder.newDocument();
    Element rootElement = doc.createElement( tagName: "contactos");
    doc.appendChild(rootElement);
}
```

```
// Guardar los cambios en el archivo XML
TransformerFactory transformerFactory = TransformerFactory.newInstance();
Transformer transformer = transformerFactory.newTransformer();
DOMSource source = new DOMSource(doc);
StreamResult result = new StreamResult(file);
transformer.transform(source, result);
```

- **Lectura:** Para buscar, modificar o eliminar contactos, la aplicación utiliza getElementsByTagName("contacto") para obtener todos los elementos *<contacto>*, del archivo y luego filtra según el nombre buscado. Si se encuentran múltiples coincidencias, se proporciona al usuario la opción de elegir cuál modificar o eliminar.

```
// Obtiene todos los elementos 'contacto' del documento XML y los almacena en un NodeList pa
NodeList contactos = doc.getElementsByTagName("contacto");
// Lista vacía para almacenar los elementos 'contacto' que coincidan con el criterio de búsq
List<Element> contactosEncontrados = new ArrayList<>();

for (int i = 0; i < contactos.getLength(); i++) {
    Element contacto = (Element) contactos.item(i);
    String nombre = contacto.getElementsByTagName("nombre").item( index: 0).getTextContent();

    if (nombre.equalsIgnoreCase(nombreBuscado)) {
        contactosEncontrados.add(contacto);
    }
}
```

- **Manejo de Errores:** Se han implementado manejadores de excepciones (try-catch) para manejar errores comunes, como problemas de configuración del parser XML, errores de entrada/salida, o problemas al transformar el archivo XML. Los mensajes de error se han personalizado para que sean más comprensibles para el usuario.

```
} catch (ParserConfigurationException | IOException | TransformerException | SAXException e) {
    System.out.println("ParserConfigurationException > Error en la configuración del parser XML.\n" +
            "IOException > Error de entrada/salida al manejar el archivo XML.\n" +
            "TransformerException > Error al transformar el documento XML.\n" +
            "SAXException > Error al analizar el archivo XML.\n" +
            e.getMessage());
}
```

4. Decisiones de Diseño.

4.1. Uso de XML para Almacenamiento.

Se decidió utilizar XML para almacenar los contactos debido a su capacidad para representar estructuras de datos jerárquicas de manera legible y su compatibilidad con diversas herramientas. XML permite una fácil expansión del formato si en el futuro se necesitan agregar más atributos a los contactos.

4.2. Manejo de Múltiples Contactos con el Mismo Nombre.

Cuando se encuentran varios contactos con el mismo nombre, se ofrece al usuario una lista para seleccionar el contacto específico que desea modificar o eliminar. Esto se hizo para evitar errores en los que el usuario elimine o modifique el contacto incorrecto.

4.3. Exportación a CSV.

Se agregó la opción de exportar a CSV para facilitar la interoperabilidad con otras herramientas, como hojas de cálculo, y para permitir a los usuarios visualizar sus contactos fuera de la aplicación.

5. Futuras Mejoras.

>> La validación de datos actualmente no se realiza una validación exhaustiva de los datos ingresados por el usuario (por ejemplo, formato del número de teléfono). Se podría agregar una validación adicional para mejorar la calidad de los datos.

>> Una interfaz gráfica para la aplicación, actualmente utiliza la consola para la interacción con el usuario. En el futuro, se podría implementar una interfaz gráfica para mejorar la experiencia del usuario.

>> Encriptación de datos para proteger la información de los contactos, se podría agregar encriptación al archivo XML.

6. Conclusión.

La aplicación de gestión de contactos proporciona una forma simple e intuitiva de almacenar y gestionar contactos utilizando archivos XML. Las decisiones tomadas se basan en la facilidad de uso, la capacidad de expansión y la claridad de los datos. La opción de exportar a CSV facilita la interoperabilidad y la manipulación de datos en otras plataformas, lo cual es un valor agregado importante.

~ FIN ~