



SISTEMA DE GESTIÓN DE PRODUCTOS

02/11/2024

Mario Prada Herruzo 2DAM

IES Antonio Gala

Palma Del Rio (Cordoba)

Descripción general

He realizado una aplicación que consiste en gestionar un almacén de productos en un archivo XML, mi aplicación te permite hacer operaciones como, Añadir, buscar, modificar, eliminar, exportar el archivo XML a otro formato... entre otras cosas. Esta herramienta puede servir como una primera aproximación para cualquier negocio pequeño que desee digitalizar y organizar su inventario sin necesidad de infraestructura complicada.

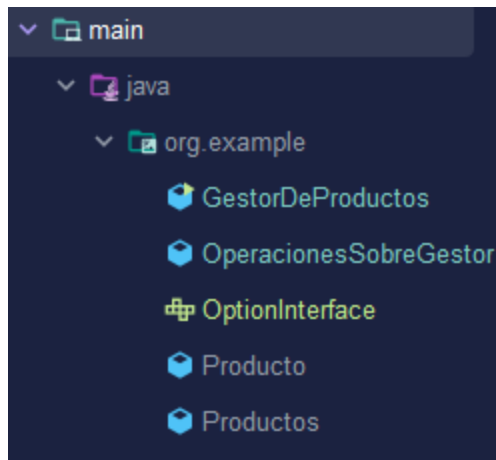
Objetivos

1. **Facilitar la gestión de productos mediante un archivo XML:** Crear una estructura XML donde se almacenen los productos de manera organizada, permitiendo su recuperación y actualización de manera sencilla.
2. **Automatizar tareas básicas de inventario:** Implementar funcionalidades que permitan agregar, modificar y eliminar productos del inventario sin necesidad de sistemas complejos o bases de datos externas. Asegurar que las operaciones sobre los productos se realicen de forma eficiente, minimizando errores de entrada de datos.
3. **Permitir la búsqueda rápida de productos específicos:** Implementar un sistema de búsqueda por nombre para facilitar la localización y verificación de productos. Proveer a los usuarios información detallada del producto buscado para una mejor toma de decisiones.
4. **Diseñar una aplicación de consola intuitiva y fácil de usar:** Desarrollar un menú de opciones claro y directo que permita a los usuarios realizar todas las operaciones sin dificultades, Minimizar la complejidad para que el sistema sea útil para usuarios con conocimientos básicos en informática.

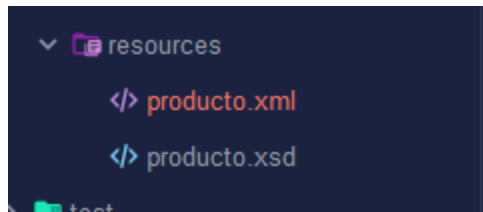
Estructura de Paquetes y Clases

Tengo el paquete Main, con las clases .java y los resources.

la parte de los .java es esta:



Y en cuanto a los resources tenemos esto:



A continuación haré una descripción de las clases e interfaces que componen mi proyecto:

Clase Gestor de Productos:

Esta clase se encarga de hacer las operaciones para gestionar los productos, contiene un menú de opciones el cual es muy intuitivo para el usuario y puede hacer operaciones como:

- Añadir Productos
- Buscar Productos
- Modificar Productos
- Eliminar Productos
- Exportar a CSV

```

System.out.println("=====");
System.out.println("1. Añadir Productos");
System.out.println("2. Buscar Producto");
System.out.println("3. Modificar Producto");
System.out.println("4. Eliminar Producto");
System.out.println("5. Exportar Producto a CSV");
System.out.println("6. Salir");
System.out.println("=====");
System.out.println("Elige una opcion: ");
opcion = sc.nextInt();
sc.nextLine();

```

Clase Operaciones Sobre Gestor

Esta clase contiene la lógica de los métodos los cuales va a utilizar la clase Gestor de Productos para hacer las operaciones que quiera el usuario.

Clase Producto

Esto es una clase que se conoce como clase POJO (Clase Clásica de Java que define un objeto) Que la uso en la Clase Operaciones Sobre Gestor de Productos para crear nuevos objetos Producto y añadirlos al XML.

Clase Productos

Esta clase almacena una Lista de Productos para guardar todos los productos del XML

Interfaz Option Interface

Esta interfaz se encarga de guardar con constantes las distintas Opciones del Menú.

Explicación Detallada Del Código

I. Método creaProducto()

```
/**
 * Crea un nuevo producto solicitando sus atributos al usuario.
 *
 * @return El producto creado.
 */
public Producto creaProducto() { 1 usage  Marioph03 *
    // Crear un nuevo objeto Producto
    Producto p = new Producto();

    // Solicitar el nombre del producto al usuario
    System.out.println("Ingrese un Nombre para el producto: ");
    p.setNombre(sc.nextLine());

    // Solicitar el precio del producto al usuario
    System.out.println("Ingrese un Precio para el producto: ");
    p.setPrecio(sc.nextDouble());

    // Solicitar la cantidad del producto al usuario
    System.out.println("Ingrese un Cantidad para el producto: ");
    p.setCantidad(sc.nextInt());

    // Limpiar el buffer del scanner
    sc.nextLine();

    return p;
}
```

Este Método se encarga de crear Producto introduciendo datos por consola

II. Método escribeProducto()

```

/**
 * Agrega un nuevo producto al archivo XML, serializándolo y almacenándolo.
 *
 * @param p El producto a escribir en el archivo XML.
 * @return true si la operación es exitosa, de lo contrario lanza una excepción.
 */
public boolean escribeProducto(Producto p) { 1 usage  🐼Manioph03 *
    // Crear un nuevo producto solicitando datos al usuario
    p = creaProducto();
    ArrayList<Producto> productos = new ArrayList<>();

    // Especificar la ubicación del archivo XML
    File f = new File( pathname: "C:\\Users\\Mario\\IdeaProjects\\SGProductos\\src\\main\\resources\\producto.xml");

    try {
        // Crear el contexto JAXB para la clase Productos
        JAXBContext jaxbContext = JAXBContext.newInstance(Productos.class);

        // Si el archivo XML ya existe, deserializar su contenido en una lista de productos
        if (f.exists()) {
            Unmarshaller um = jaxbContext.createUnmarshaller();
            Productos productosWrapper = (Productos) um.unmarshal(f);
            if (productosWrapper.getProductos() != null) {
                productos.addAll(productosWrapper.getProductos());
            }
        }
    }
}

```

```

    // Agregar el nuevo producto a la lista de productos
    productos.add(p);

    // Serializar la lista actualizada de productos y guardarla en el archivo XML
    Marshaller m = jaxbContext.createMarshaller();
    m.setProperty(Marshaller.JAXB_FORMATTED_OUTPUT, true);
    m.marshal(new Productos(productos), f);
    System.out.println("El producto se ha guardado correctamente");

} catch (JAXBException e) {
    throw new RuntimeException(e);
}

return true;
}

```

Este método se encarga de escribir los productos en el XML

Método buscarProducto()

```
/**
 * Busca un producto en el archivo XML según su nombre y lo muestra si es encontrado.
 *
 * @param nombre El nombre del producto a buscar.
 * @return El producto encontrado, o un objeto vacío si no existe.
 */
public Producto buscarProducto(String nombre) { 1 usage  🚩 MarioPh03
    Producto p = new Producto();
    try {
        // Ruta del archivo XML
        File archivo = new File( pathname: "C:\\Users\\Mario\\IdeaProjects\\SGProductos\\src\\main\\resources\\producto.xml");

        // Configurar el parser XML
        DocumentBuilderFactory docFactory = DocumentBuilderFactory.newInstance();
        DocumentBuilder docBuilder = docFactory.newDocumentBuilder();
        Document doc = docBuilder.parse(archivo);
        doc.getDocumentElement().normalize();

        // Obtener la lista de productos del archivo XML
        NodeList listaProductos = doc.getElementsByTagName("producto");

        boolean productoEncontrado = false;
        // Iterar por cada producto en la lista para buscar por nombre
        for (int i = 0; i < listaProductos.getLength(); i++) {
            Element producto = (Element) listaProductos.item(i);
            String nombreProducto = producto.getElementsByTagName("nombre").item( index: 0).getTextContent();

            if (nombreProducto.equalsIgnoreCase(nombre)) {
                // Si el producto es encontrado, extraer sus datos y asignarlos al objeto Producto
                productoEncontrado = true;
                p.setNombre(nombreProducto);
                p.setCantidad(Integer.parseInt(producto.getElementsByTagName("cantidad").item( index: 0).getTextContent()));
                p.setPrecio(Double.parseDouble(producto.getElementsByTagName("precio").item( index: 0).getTextContent()));
                System.out.println("Producto encontrado:\nNombre: " + p.getNombre() + "\nCantidad: " + p.getCantidad() + "\nPrecio: $" + p.getPrecio());
                break;
            }
        }
        if (!productoEncontrado) {
            System.out.println("El producto no existe.");
        }
    } catch (ParserConfigurationException | IOException | SAXException e) {
        e.printStackTrace();
    }
    return p;
}
```

Este método se encarga de buscar Productos en el archivo XML

Método modificarProductos()

```
/**
 * Modifica los atributos de un producto especificado por su nombre en el archivo XML.
 *
 * @param nombre El nombre del producto a modificar.
 * @return El producto modificado con los nuevos valores.
 */
public Producto modificarProducto(String nombre) { 1 usage  🐼 MarioPh03 *
    String nuevoNombre;
    int nuevaCantidad;
    double nuevoPrecio;
    Producto p = new Producto();
    try {
        // Ruta del archivo XML
        File archivo = new File( pathname: "C:\\Users\\Mario\\IdeaProjects\\SGProductos\\src\\main\\resources\\producto.xml");

        // Configurar el parser XML
        DocumentBuilderFactory docFactory = DocumentBuilderFactory.newInstance();
        DocumentBuilder docBuilder = docFactory.newDocumentBuilder();
        Document doc = docBuilder.parse(archivo);
        doc.getDocumentElement().normalize();

        // Obtener la lista de productos del archivo XML
        NodeList listaProductos = doc.getElementsByTagName("producto");
        boolean productoEncontrado = false;

        // Iterar por cada producto en la lista para buscar por nombre
        for (int i = 0; i < listaProductos.getLength(); i++) {
            Element producto = (Element) listaProductos.item(i);
            String nombreProducto = producto.getElementsByTagName("nombre").item( index: 0).getTextContent();
```



```

    if (nombreProducto.equalsIgnoreCase(nombre)) {
        // Solicitar nuevos valores y actualizarlos en el XML
        productoEncontrado = true;
        System.out.println("Introduce el nuevo nombre del producto: ");
        nuevoNombre = sc.nextLine();
        producto.getElementsByTagName("nombre").item( index: 0).setTextContent(nuevoNombre);
        System.out.println("Introduce el nuevo precio del producto: ");
        nuevoPrecio = sc.nextDouble();
        producto.getElementsByTagName("precio").item( index: 0).setTextContent(String.valueOf(nuevoPrecio));
        System.out.println("Introduce la nueva cantidad del producto: ");
        nuevaCantidad = sc.nextInt();
        producto.getElementsByTagName("cantidad").item( index: 0).setTextContent(String.valueOf(nuevaCantidad));

        // Actualizar los valores del objeto Producto
        p.setNombre(nuevoNombre);
        p.setPrecio(nuevoPrecio);
        p.setCantidad(nuevaCantidad);
        break;
    }
}

// Guardar los cambios en el archivo XML si se encontró el producto
if (productoEncontrado) {
    Transformer transformer = TransformerFactory.newInstance().newTransformer();
    transformer.transform(new DOMSource(doc), new StreamResult(archivo));
    System.out.println("Los cambios se han guardado en el archivo XML.");
} else {
    System.out.println("El producto no existe.");
}

```

```

    } catch (ParserConfigurationException | IOException | SAXException | TransformerException e) {
        e.printStackTrace();
    }
    return p;
}

```

Este método se encarga de modificar Productos del XML según el nombre que se le pase por parámetro.

Método eliminaProducto()

```

/**
 * Elimina un producto del archivo XML según su nombre.
 *
 * @param nombre El nombre del producto a eliminar.
 * @return true si el producto fue eliminado, false si no se encontró.
 */
public boolean eliminarProducto(String nombre) { 1 usage  🚩 MarioPh03
    boolean productoEliminado = false;

    try {
        // Ruta del archivo XML
        File archivo = new File( pathname: "C:\\Users\\Mario\\IdeaProjects\\SGProductos\\src\\main\\resources\\producto.xml");

        // Configurar el parser XML
        DocumentBuilderFactory docFactory = DocumentBuilderFactory.newInstance();
        DocumentBuilder docBuilder = docFactory.newDocumentBuilder();
        Document doc = docBuilder.parse(archivo);
        doc.getDocumentElement().normalize();

        // Obtener la lista de productos del archivo XML
        NodeList listaProductos = doc.getElementsByTagName("producto");

        // Iterar por cada producto en la lista para buscar por nombre
        for (int i = 0; i < listaProductos.getLength(); i++) {
            Element producto = (Element) listaProductos.item(i);
            String nombreProducto = producto.getElementsByTagName("nombre").item( index: 0).getTextContent();

```

```

            if (nombreProducto.equalsIgnoreCase(nombre)) {
                // Eliminar el producto encontrado
                producto.getParentNode().removeChild(producto);
                productoEliminado = true;
                System.out.println("Producto eliminado correctamente.");
                break;
            }
        }

        // Guardar los cambios en el archivo XML si se eliminó un producto
        if (productoEliminado) {
            Transformer transformer = TransformerFactory.newInstance().newTransformer();
            transformer.setOutputProperty(OutputKeys.INDENT, value: "yes");
            transformer.transform(new DOMSource(doc), new StreamResult(archivo));
        } else {
            System.out.println("El producto no existe.");
        }
    } catch (ParserConfigurationException | IOException | SAXException | TransformerException e) {
        e.printStackTrace();
    }

    return productoEliminado;
}

```

Este método se encarga de eliminar un producto del XML cuyo nombre se le especifique por parámetro.

Método leerProductosDesdeXML()

```
/**
 * Lee productos desde un archivo XML y los convierte en una lista de objetos Producto.
 *
 * @param archivoXML La ruta del archivo XML.
 * @return La lista de productos leídos del archivo.
 */
public ArrayList<Producto> leerProductosDesdeXML(String archivoXML) { 1 usage  Marioph03
    ArrayList<Producto> productos = new ArrayList<>();
    try {
        JAXBContext jaxbContext = JAXBContext.newInstance(Productos.class);
        Unmarshaller unmarshaller = jaxbContext.createUnmarshaller();
        Productos productosWrapper = (Productos) unmarshaller.unmarshal(new File(archivoXML));
        if (productosWrapper.getProductos() != null) {
            productos.addAll(productosWrapper.getProductos());
        }
    } catch (JAXBException e) {
        System.err.println("Error al leer productos desde XML: " + e.getMessage());
    }
    return productos;
}
```

Este método recorre el archivo XML entero y devuelve todos los productos que encuentre en una lista.

Método exportarProductosACSV()

```
/**
 * Exporta una lista de productos a un archivo CSV.
 *
 * @param productos La lista de productos a exportar.
 * @param archivoCSV La ruta del archivo CSV de salida.
 */
public void exportarProductosACSV(ArrayList<Producto> productos, String archivoCSV) { 1 usage  Marioph03 *
    try (FileWriter writer = new FileWriter(archivoCSV)) {
        writer.write( str. "Nombre,Precio,Cantidad\n");
        for (Producto producto : productos) {
            writer.write( str. producto.getNombre() + "," + producto.getPrecio() + "," + producto.getCantidad() + "\n");
        }
        System.out.println("Los productos se han exportado correctamente a " + archivoCSV);
    } catch (IOException e) {
        System.out.println("Error al exportar los productos a CSV: " + e.getMessage());
    }
}
```

Este método se encarga de escribir el contenido del archivo XML en un formato CSV.

Método exportarXMLaCSV()

```
/**
 * Lee productos de un archivo XML y los exporta a un archivo CSV.
 *
 * @param archivoXML La ruta al archivo XML de entrada.
 * @param archivoCSV La ruta del archivo CSV de salida.
 * @return true si la operación es exitosa.
 */
public boolean exportarXMLaCSV(String archivoXML, String archivoCSV) { 1 usage  Marioph03 *
    ArrayList<Producto> productos = leerProductosDesdeXML(archivoXML);
    exportarProductosACSV(productos, archivoCSV);
    return true;
}
```

Este método se encarga de convertir el archivo XML a un archivo CSV, si se ha hecho la operación con éxito devuelve un true.

Ejemplos de Uso

A continuación voy a mostrar el resultado de los métodos, al ejecutarlos por consola:

Este es el menú de la aplicación:

```
=====
1. Añadir Productos
2. Buscar Producto
3. Modificar Producto
4. Eliminar Producto
5. Exportar Producto a CSV
6. Salir
=====
Elige una opcion:
|
```

Ejemplo de uso agregarProductos()

```
-----
Elige una opcion:
1
Ingrese un Nombre para el producto:
Boligrafo
Ingrese un Precio para el producto:
2
Ingrese un Cantidad para el producto:
9
El producto se ha guardado correctamente
```

Ejemplo de uso buscarProducto()

```
Elige una opcion:  
2  
Introduce el nombre del producto que quieres buscar:  
Boligrafo  
Producto encontrado:  
Nombre: Boligrafo  
Cantidad: 9  
Precio: $2.0
```

Ejemplo de uso modificaProducto()


```
Elige una opcion:  
3  
Introduce el nombre del producto que quieres modificar:  
Boligrafo  
Introduce el nuevo nombre del producto:  
Lapiz  
Introduce el nuevo precio del producto:  
1,50  
Introduce la nueva cantidad del producto:  
10  
Los cambios se han guardado en el archivo XML.
```

Ejemplo de uso eliminaProducto()

```
Elige una opcion:  
4  
Introduce el nombre del producto que quieres eliminar:  
Lapiz  
Producto eliminado correctamente.
```

Ejemplo de uso exportarXMLaCSV()

```
Elige una opcion:
5
Introduce la ruta del archivo que quieres exportar:
C:\Users\Mario\IdeaProjects\SGProductos\src\main\resources\producto.xml
Introduce el nombre del archivo al que quieres exportar:
productosCSV
Los productos se han exportado correctamente a productosCSV
```

 productosCSV

Posibles Mejoras Futuras

Como futuras actualizaciones a mi proyecto:

- Se me ha ocurrido que se podría incluir una interfaz gráfica, para que sea mucho más llamativo y visual para el usuario.
- También se me ha ocurrido introducir nuevas opciones de exportación, por ejemplo, dar la opción a exportar a formato JSON u otros formatos existentes.

Conclusión

El desarrollo de este sistema de gestión de productos ha logrado construir una solución funcional y accesible para la administración de inventario básico mediante archivos XML. Este sistema proporciona una plataforma sencilla y confiable que permite a los usuarios realizar operaciones esenciales, como agregar, modificar, buscar y eliminar productos, asegurando la persistencia de los datos.

En conclusión, el sistema ha cumplido sus objetivos al ofrecer una solución de gestión de productos práctica, confiable y adaptable, ideal para pequeños negocios o individuos que requieren un control básico de inventario sin complicaciones.

