	Carátula para entrega de prácticas
Facultad de Ingeniería	Laboratorio de docencia

Laboratorios de computación salas A y B

Profesor: René Adrián Dávila Pérez

Asignatura: Programación Orientada a Objetos

Grupo: 01

No. de práctica(s): 05 y 06

Integrante(s):
322276824
322258516
425037384
320108116
322221415

No. de brigada: 01

Semestre: 2026-1

Fecha de entrega: 03 de octubre de 2025

Observaciones:

CALIFICACIÓN: _____

Índice

1. Introducción

- **Planteamiento del problema:**
  Desarrollar una aplicación en Java que simule el funcionamiento de un carrito de compras con interfaz gráfica. La aplicación debe permitir al usuario agregar, eliminar y visualizar artículos, implementando encapsulamiento, además de organizar el código dentro de un paquete que cumpla con la notación de *Full Qualified Name*.

- **Motivación:**
  El uso del encapsulamiento es un principio fundamental para proteger los atributos de una clase, al igual que el manejo de paquetes en Java para estructurar y organizar proyectos de forma profesional. Dominar estos aspectos resulta crucial para desarrollar un flujo de trabajo eficiente y escalable al ámbito profesional.

- **Objetivos:**
  Poner en práctica los conceptos de encapsulamiento y empaquetado en Java, integrando la manipulación de objetos a través de clases bien definidas y diseñar una interfaz gráfica que permita interactuar con los objetos de la aplicación ofreciendo una interacción sencilla para el usuario.

2. Marco Teórico

Encapsulamiento

El encapsulamiento es uno de los principios fundamentales de la programación orientada a objetos. Permite ocultar los detalles internos de una clase y exponer únicamente lo necesario mediante métodos públicos. Según Oracle: "Ocultar el estado interno y exigir que todas las interacciones se realicen a través de los métodos de un objeto se conoce como encapsulación de datos" [4]. De esta manera se logra un mayor control sobre los datos, mejorando la seguridad y reduciendo la posibilidad de errores en el manejo de objetos.

Paquetes en Java y Fully Qualified Name

Los paquetes en Java constituyen una herramienta para organizar y agrupar clases relacionadas dentro de un mismo espacio de nombres, estos ayudan a evitar conflictos de nombres y hacen que sea más fácil localizar y usar clases, interfaces, enumeraciones y anotaciones relacionadas. [1]
Fully Qualified Name, se refiere a la especificación de una clase incluyendo su paquete. Esto permite distinguir clases con el mismo nombre pero en paquetes diferentes. Por ejemplo: `mx.unam.fi.poo.p56.Articulo`.

Interfaces gráficas de usuario (GUI)

Una interfaz gráfica de usuario es un conjunto de elementos visuales que permite la interacción entre el usuario y el software. "Se refiere a las ventanas gráficas e interactivas con las que el usuario de una aplicación puede interactuar, en lugar de utilizar sólo la ventana de comandos del sistema." [3].
En Java, la biblioteca `Swing` ofrece un amplio conjunto de clases como `JFrame`, `JButton`, `JTextField`, `JList` y `JLabel`, que permiten la construcción de interfaces flexibles e independientes de la plataforma.

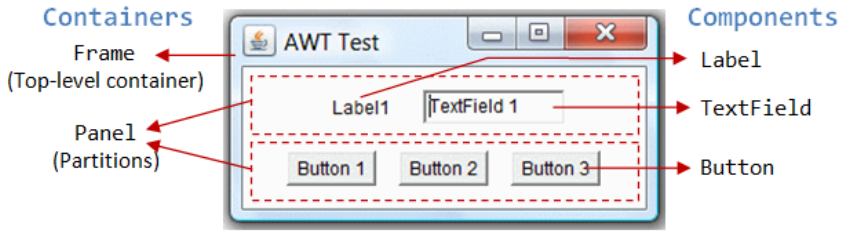

Figura 1: Elementos básicos de una GUI simple. [2]

3. Desarrollo

La aplicación se estructuró en cuatro clases principales, todas contenidas en el paquete `mx.unam.fi.poo.p56`: `Articulo`, `Carrito`, `Vista` y `MainApp`. Esta organización permitió separar responsabilidades y mantener un diseño modular, en el cual cada clase desempeña un rol específico.

Clase Articulo

La clase `Articulo` representa los objetos que se gestionan en el carrito. Sus atributos `nombre` y `precio` se declararon privados, aplicando encapsulamiento. Se implementaron métodos `get` y `set` para el acceso controlado de dichos atributos.
El método `toItemString()` le da formato al nombre y precio de un artículo en forma legible para su visualización en la interfaz gráfica, utilizando cálculos internos para asegurar siempre dos decimales en la representación monetaria.

Clase Carrito

La clase `Carrito` administra los objetos de tipo `Articulo` mediante un `ArrayList`. Se implementaron métodos para:

- `agregar()`: añade un artículo validando que no sea nulo ni tenga nombre vacío.
- `eliminarPorIndice()`: elimina un producto en función de su posición en la lista, validando primero que el índice sea válido.

- `eliminarPorNombre()`: busca y elimina un artículo cuyo nombre coincida con el texto proporcionado usando `equalsIgnoreCase` para tener un manejo más robusto de las cadenas.
- `limpiar()`: vacía completamente la lista.
- Métodos como `getArticulos()`, `getTotal()`, `getNumeroArticulos()` y `getVacio()` para obtener información del carrito.

Clase Vista

La clase `Vista` hereda de `JFrame` y concentra la definición de la interfaz gráfica. Implementa campos de texto para ingresar nombre y precio de artículos (`JTextField`), botones para las operaciones principales (`JButton`), un área de lista para mostrar el contenido del carrito (`JList`) y etiquetas que informan al usuario sobre el estado de la aplicación y el total acumulado (`JLabel`). Todos los atributos se declaran privados y se puede acceder a ellos mediante métodos `get`, respetando el principio de encapsulamiento.
En cuanto a la construcción de la interfaz, se definieron cuatro paneles para organizar los componentes, el primero de ellos para los datos del artículo que el usuario ingresará, el segundo para colocar los botones que le permiten agregar, eliminar o limpiar el carrito al usuario, el tercero para mostrar la lista de productos en el carrito y el último para incluir las etiquetas informativas. Con la instrucción `pack()` se ajustaron automáticamente las dimensiones de la ventana y `setLocationRelativeTo(null)` centró la interfaz en pantalla.

Clase MainApp

La clase `MainApp` contiene el método principal `main`, responsable de inicializar el programa y los eventos mediante `SwingUtilities.invokeLater()`. En este método se instancian los objetos de `Carrito` y `Vista`, además de asignar los `ActionListener` en cada botón de la interfaz:

- **Agregar:** valida entradas para evitar nombres nulos o precios negativos, crea un nuevo `Articulo` y lo añade al carrito, muestra un mensaje para informar al usuario si se logró o no añadir el artículo al carrito.
- **Eliminar seleccionado:** se obtiene el índice del artículo mediante el método `getSelectedIndex` y se remueve con el método `eliminarPorIndice` del objeto `carrito`.
- **Eliminar por nombre:** busca y elimina un producto a partir del nombre proporcionado invocando al método `eliminarPorNombre` del objeto `carrito`.
- **Limpiar:** vacía todo el carrito haciendo uso del método `limpiar`.

Asimismo, se implementaron métodos auxiliares como `isNombreValido` para validar que el nombre tenga al menos un caracter que no sea un espacio, `safeParsePrecio` que convierte de `String` a `double` de manera segura, `refrescarLista` para mantener sincronizada la lista con el estado real del carrito, `limpiarCampos` para vaciar la entrada y `formatMoney` para formatear a valores monetarios.

4. Resultados

Figura 2: Interfaz gráfica del programa.

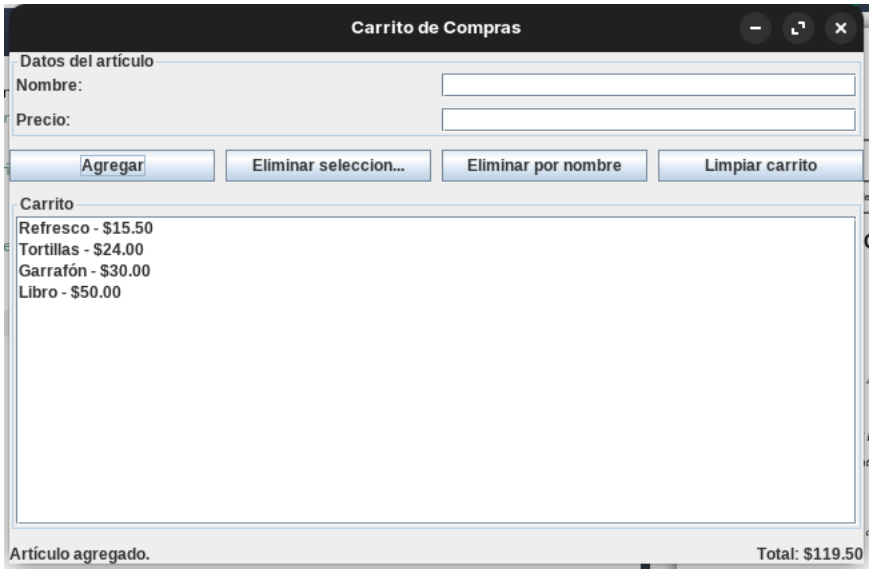

Figura 3: Agregando artículos al carrito.

Figura 4: Eliminación de un artículo por selección.

Figura 5: Eliminación de un artículo por nombre.

Figura 6: Carrito vacío tras ejecutar la opción de limpiar.

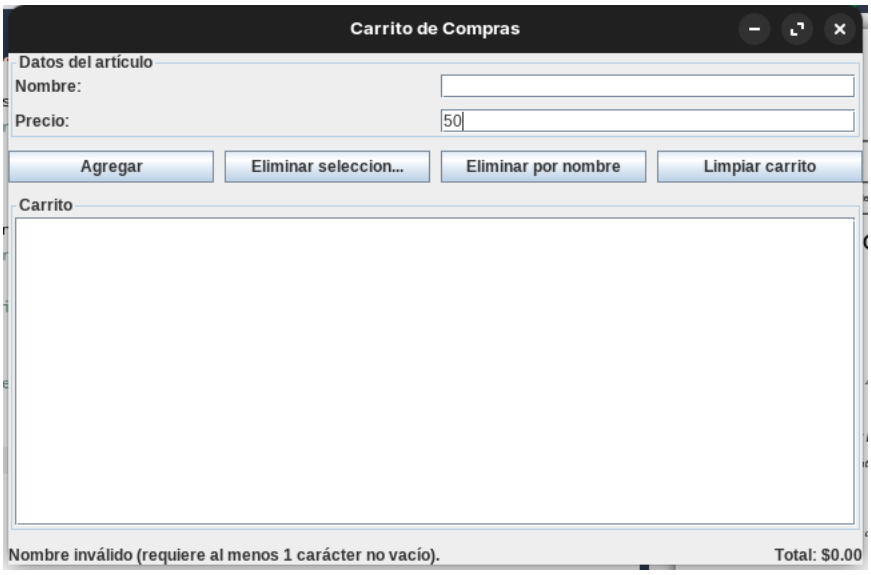

Figura 7: Probando agregar artículos con nombre inválido.

Figura 8: Probando agregar artículos con precio inválido.

5. Conclusiones

La práctica permitió integrar conceptos clave de la programación orientada a objetos, destacando el uso del encapsulamiento en las clases y la correcta organización del código mediante paquetes y nombres calificados completos.
El desarrollo de una interfaz gráfica con `Swing` reforzó la comprensión de los eventos, mostrando cómo capturar interacciones del usuario para manipular los objetos de un programa de manera dinámica.

Referencias

[1] *Creating and Using Packages*. Oct. de 2025. URL: https://docs.oracle.com/javase/tutorial/java/package/packages.html.

[2] *Programming Graphical User Interface (GUI)*. Abr. de 2021. URL: https://www3.ntu.edu.sg/home/ehchua/programming/java/J4a_GUI.html.

[3] J. A. Solano. *Interfaz gráfica de usuario*. Unidades de Apoyo para el Aprendizaje. CUAED/Facultad de Ingeniería - UNAM. Disponible en: https://repositorio-uapa.cuaed.unam.mx/repositorio/moodle/pluginfile.php/3058/mod_resource/content/1/UAPA-Interfaz-Grafica-Usuario/index.html. 2020.

[4] *What Is an Object?* Oct. de 2025. URL: https://docs.oracle.com/javase/tutorial/java/concepts/object.html.