	Carátula para entrega de prácticas
Facultad de Ingeniería	Laboratorios de docencia

Laboratorio de Computación Salas A y B

Profesor(a): M.C. RENE ADRIAN DAVILA PEREZ

Asignatura: PROGRAMACIÓN ORIENTADA A OBJETOS

Grupo: 01

No de Práctica(s): 5 y 6

Integrante(s): 322118311

322094028

322092842

322078673

322067738

No. de lista o brigada: 03

Semestre: 2026-1

Fecha de entrega: 03/10/2025

Observaciones:

CALIFICACIÓN: _____

Índice

1 Introducción

- **Planteamiento del problema.** Para esta práctica se realizara la modificación de una aplicación en Java que simula un carrito de compras, se aplicara correctamente el encapsulamiento en la clase Articulo. Los atributos como nombre y precio se deberan declarar privados y se accedera a ellos únicamente mediante métodos getters y setters, para que los datos sean manipulados de manera segura y controlada.
- **Motivación.** El encapsulamiento es fundamental en la programación orientada a objetos, ya que se encarga la integridad de datos y evita que sean modificados de forma de forma incorrecta. Implementarlo en esta práctica nos permitira reforzar la comprensión de el contrl del acceso a los atributos de los objetos y aplicar buenas prácticas de programación.
- **Objetivos.** El objetivo de la práctica sera asegurar que los atributos de la clase Articulo estén correctamente encapsulados, asegurando a su vez que la aplicación del carrito de compras y su interfaz gráfica funcione de forma correcta y asegurandonos que la manipulación de los datos se realice únicamente a través de métodos controlados.

2 Marco Teórico

2.1 Encapsulamiento

El encapsulamiento es una herramienta importante de la programación orientada a objetos que consiste en proteger los atributos de un objeto restringiendo su acceso directo y controlando su manipulación mediante métodos controlados(getters y setters).

2.1.1 Modificadores de acceso en Java

Controlan la visibilidad de los atributos y métodos:

- `private`: Solo son accesible dentro de la misma clase.
- `public`: Pueden ser accedido desde cualquier otra clase.
- `protected`: Accesibles dentro del mismo paquete o por clases derivadas.[1]

2.1.2 Getters y Setters

Los métodos **getters**permiten obtener el valor de un atributo, mientras que los **setters** permiten modicar los atributos de un objeto.

```
// Ejemplo de encapsulamiento
Persona p = new Persona();
p.setNombre("Juan");  //uso del setter
System.out.println(p.getNombre());  // Se obtiene el nombre
   mediante getter
```

2.2 Empaquetado (Packages)

El Packages consiste en organizar clases relacionadas dentro de paquetes, lo que facilita la organización del código.

2.2.1 Full Qualified Name (FQN)

El **Full Qualified Name** es el formato para el nombre completo de una clase, incluyendo su paquete. Por ejemplo:

```
// Ejemplo de uso de FQN
mx.unam.fi.poo.p56.Articulo a = new mx.unam.fi.poo.p56.Articulo("
    gansito", 15.0);
```

3 Desarrollo

En esta práctica se trabajo en la modificación de una aplicación que simula un carrito de compras en Java, enfocándose en la implementación de **encapsulamiento** y **empaquetado**. Para realizar correctamente el "Package"se ubicaron las clases dentro de un direcotorio **mx/unam/fi/poo/p56/** para posteriormente agregar al inicio de cada archivo package **mx.unam.fi.poo.p56;**. Para el encapsulamiento se modificaron las clases Articulo y Carrito, las cuales se describen a continuación.

3.1 Archivo: Articulo.java

3.1.1 Clase: Articulo

La clase Articulo define un objeto con los atributos nombre y un precio, los cuales originalmente accesibles directamente desde cualquier clase (publicos) dentro del mismo paquete, se modificaron aplicando el encapsulamiento, declaramos ambos atributos como private y agregando métodos getter y setter, getNombre, setNombre, getPrecio y setPrecio, para acceder y modificar los valores de manera controlada. Además se modifico el constructor de la clase ahora utilizara los setters para asignar los valores iniciales, en lugar de acceder directamente a los atributos, se incluyó también la declaración de paquete package mx.unam.fi.poo.p56; para ubicar la clase dentro de la estructura de empaquetamiento definida para la práctica. El método toItemString() mantiene su función (convertir los atributos en una cadena con formato, mostrando el precio con dos decimales).

3.2 Archivo: Carrito.java

3.2.1 Clase: Carrito

Esta clase la versión original, la lista articulos era accesible directamente y los atributos de los objetos Articulo también se manipulaban de manera directa, Para realizar corrrectamente el encapsulamiento y funcionara con la clase Articulo, la lista se declaró como private final, asimismo, todos los accesos a los atributos de los artículos se realizaron a través de los getters getNombre() y getPrecio(). El constructor y los métodos principales conservan su lógica, pero ahora interactúan con los artículos usando los métodos proporcionados por la clase Articulo. La clase `Carrito` contiene los siguientes metodos:

- **Agregar artículo:** Permite añadir un objeto `Articulo` al carrito, verificando que no sea nulo.
- **Eliminar por índice:** Elimina un artículo en la posición que indique el usuario, cuenta también con una validación de rango.
- **Eliminar por nombre:** Busca y elimina un artículo por nombre.
- **Limpiar:** Elimina todos los artículos en la lista.

- **Obtener total:** Calcula el precio total de los artículos agregados a la lista.

4 Resultados

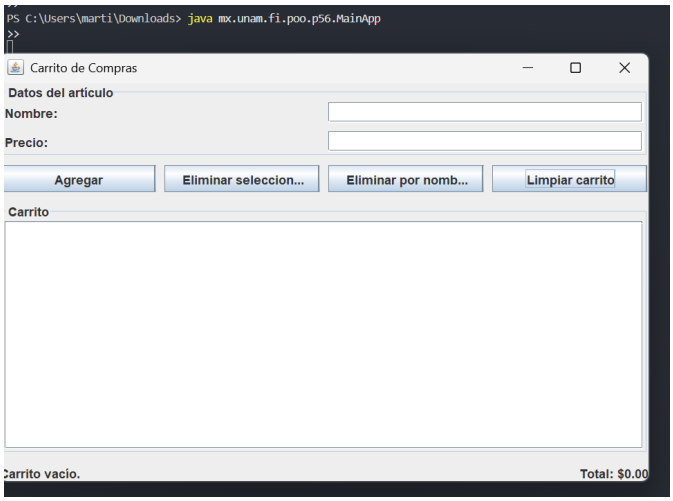

Figura 1: Se muestra la correcta ejecución del programa utilizando la clase principal MainApp a través de su Full Qualified Name, nos muestra la interfaz gráfica creada en la clase Vista con todos los componentes funcionales del carrito de compras, comprobando la correcta interacción entre los objetos Articulo y Carrito con encapsulamiento.

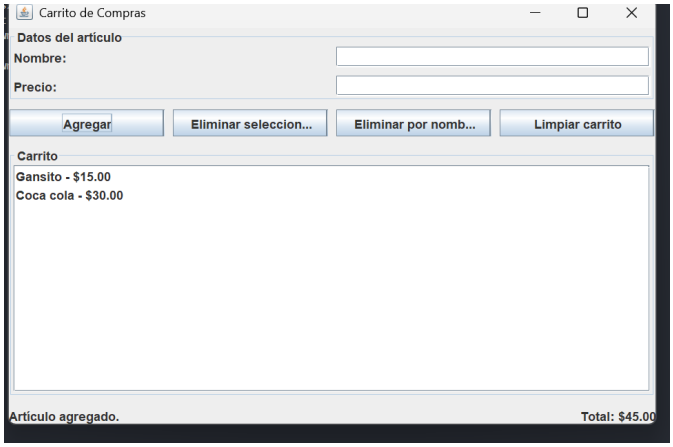

Figura 2: Uso del metodo agregar articulo

5 Conclusiones

La correcta implementación del encapsulamiento permitió controlar de manera segura el acceso a los atributos de los objetos, como nombre y precio. Esto garantizó que los datos fueran manipulados únicamente a través de métodos getters y setters, evitando modificaciones indebidas y manteniendo la seguridad de la información dentro de la aplicación. Ademas logramos organizar la aplicacion en paquetes siguiendo el "Full Qualified Name", lo cual nos facilita la organización del código en módulos claros y hace que el proyecto sea más legible.

6 Bibliografia

[**1**] J. López Blasco, Ïntroducción a POO en Java: Encapsulamiento,"*OpenWebinars*, 10 de noviembre de 2023. [Online]. disponible en: https://openwebinars.net/blog/introduccion-a-poo-en-java-encapsulamiento/

[**2**] 73.- Explicación de paquetes en Java- ¿Cómo organizar las clases en paquetes? [Video]. YouTube. https://youtu.be/iEi-xEW0MDU