	Carátula para entrega de prácticas
Facultad de Ingeniería	Laboratorios de docencia

Laboratorio de Computación Salas A y B

Profesor(a): Dávila Pérez René Adrián

Asignatura: Programación Orientada a Objetos

Grupo: 1

No de Práctica(s): Proyecto 1

Integrante(s): 322089020

322089817

322151194

425091586

322085390

No. de lista o brigada: Equipo 4

Semestre: 2026-1

Fecha de entrega: 26 de septiembre de 2025

Observaciones:

CALIFICACIÓN: _____

Índice

1. Introducción **2**

2. Marco Teórico **3**

2.1. Relaciones entre clases 3

2.2. Constructores 4

2.3. Estructuras de Control en Java 5

2.4. Tipos Fundamentales 5

2.5. Gestión Básica de Menús en Consola 6

3. Desarrollo **7**

4. Resultados **10**

5. Conclusiones **13**

1. Introducción

Planteamiento del Problema

El problema a resolver consiste en el desarrollo de una aplicación en Java que simule el comportamiento básico de un carrito de compras, la solución debe permitir la gestión de una colección de artículos, implementando funcionalidades esenciales como agregar, quitar y consultar los productos almacenados en dicho carrito, el reto principal es modelar este concepto del mundo real utilizando los principios de la Programación Orientada a Objetos.

Motivación

La simulación de un carrito de compras es un ejercicio fundamental para aplicar de manera práctica los conceptos teóricos de la POO. Este proyecto permite materializar ideas como la creación de clases (Carrito), la encapsulación de datos (atributos como artículo y precio) y la definición de comportamientos (métodos para agregar o quitar elementos). Dar solución a este problema es necesario para reforzar el entendimiento de cómo las clases y los objetos interactúan para construir una aplicación funcional y estructurada..

Objetivos

El objetivo general es implementar una aplicación funcional en Java que represente un carrito de compras. Los objetivos específicos podemos definirlos de la siguiente manera:

- Diseñar e implementar una clase Carrito que contenga los atributos y métodos necesarios para su operación.
- Utilizar una estructura de datos para almacenar de forma dinámica los artículos.
- Establecer la comunicación entre la clase principal y la clase Carrito a través de la instanciación de objetos.
- Demostrar que es posible manipular el contenido del carrito mediante las operaciones de agregar y quitar artículos.
- Verificar el estado final del carrito imprimiendo una lista con todo su contenido.

2. Marco Teórico

2.1. Relaciones entre clases

Cuando creamos nuestro código se realizan diversos componentes para poder identificar sus partes y funcionamiento a la hora de ejecutar el programa. Las clases que se relacionan son fundamentales en la Programación Orientada a Objetos. El lenguaje Unificado de Modelado define las siguientes relaciones fundamentales entre clases:

- **Asociación:** Representa la relación entre dos o más clases. Existe si un objeto de una clase requiere un objeto de otra clase para hacer su trabajo. [2]

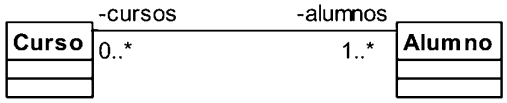

Figura 1: Diagrama de asociación de clases.

- **Agregación:** Se conoce como relación débil se reconoce como ***contiene*** o es ***contenido en***. Ambas clases son independientes, pero una trabaja en orden de otra. Se denota como una línea con rombo blanco, donde el rombo está en el huésped y el otro lado en el invitado. [7]

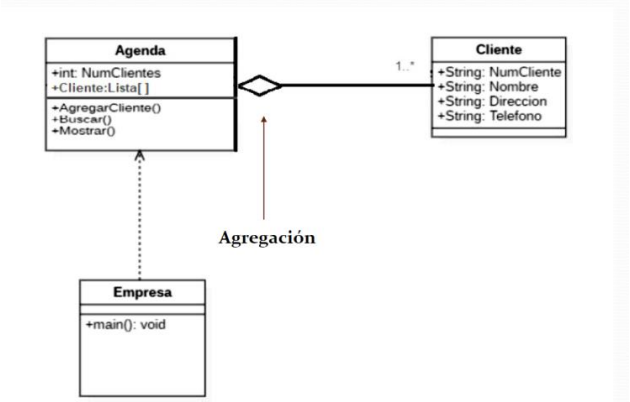

Figura 2: Diagrama agregación de clases.

- **Composición:** También conocida como relación fuerte, los objetos como atributos no tienen sentido fuera del objeto resultante. Los objetos deben dejar de existir cuando lo hace el objeto compuesto. Se ubica como ***es parte de*** o ***es un todo de***. Las dos partes necesitan de ellas para existir, por lo tanto existe una clase todo que utiliza las características de otra para la ejecución de alguna tarea. [2]

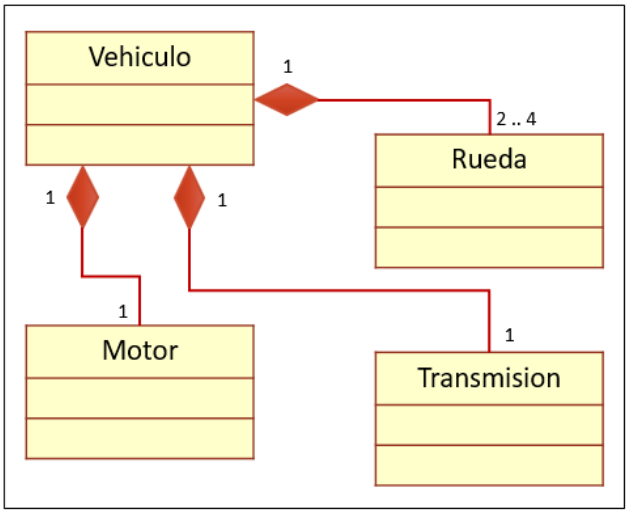

Figura 3: Diagrama composición de clases.

2.2. Constructores

En Java, los constructores son métodos que se utilizan cuando inicializamos objetos en el código. A diferencia de los métodos normales, estos se invocan cuando se crea una instancia de una clase. Es importante considerar que siempre tienen el mismo nombre que la clase y no tienen tipo de retorno (ni si quiera void). Son esenciales para poder establecer valores iniciales de los atributos del objeto y prepararlo.

Podemos mencionar 4 tipos de constructores: **por defecto**, el compilador proporciona automáticamente el constructor si no lo define el mismo programador, el constructor aparecerá en la clase. Inicializa igualmente el objeto con valores por defecto; **sin argumentos**, que es definido por el programador y no recibe ningún parámetro, parecido al anterior pero puede incluir código de inicialización personalizado; **el parametrizado**, acepta argumentos para inicializar un objeto con valores específicos; **copiar constructor**, que sería una copia de un constructor para crear un objeto nuevo como la copia de un objeto existente, se puede implementar manualmente. [1]

2.3. Estructuras de Control en Java

Las **estructuras de control** definen el **flujo de ejecución** de un programa, alterando la secuencia lineal de las instrucciones. Estas construcciones son esenciales para implementar la lógica de decisión y repetición. La documentación oficial de Oracle las categoriza como *Control Flow Statements*. [5]

2.4. Tipos Fundamentales

Las estructuras se dividen en tres clases principales:

1. **Condicionales (Decisión)**: Ejecutan bloques de código basados en la evaluación de una **condición booleana** (verdadera o falsa). Incluyen ‘if‘, ‘if-else‘, y ‘switch‘.

2. **Bucles (Iteración)**: Repiten un fragmento de código mientras se cumpla una condición. Incluyen ‘for‘, ‘while‘, y ‘do-while‘.

3. **Sentencias de Salto**: Modifican el flujo dentro de bucles o métodos, como ‘break‘, ‘continue‘, y ‘return‘.

En Java, a diferencia de otros lenguajes, las condiciones de decisión y bucle deben evaluarse estrictamente como valores de tipo **booleano**. [6]

—

2.5. Gestión Básica de Menús en Consola

La gestión de menús en consola se refiere a la creación de **interfaces de texto** que permiten al usuario elegir opciones para interactuar con el programa. Esta interacción se logra mediante la coordinación de la **entrada** y **salida** estándar junto con las **estructuras de control**. [3]

Componentes Clave

1. **Salida de Datos**: Se utiliza la clase `System.out` para **mostrar** las opciones disponibles.

2. **Entrada de Datos**: Se emplea la clase `java.util.Scanner` para **capturar** la selección del usuario desde el teclado.

3. **Lógica del Menú**: Se usan estructuras de control (principalmente `do-while` para el ciclo y `switch` para la decisión) para procesar la elección y repetir el menú. [4]

Buenas Prácticas de Entrada

Para asegurar la robustez del menú, es crucial realizar la **validación de entrada** antes de su procesamiento. La clase `Scanner` ofrece métodos como `hasNextInt()` que permiten verificar si el dato ingresado es del tipo esperado, previniendo errores de ejecución. [4]

3. Desarrollo

Implementación del Código

El programa actúa como un carrito de compras en el que el usuario puede agregar artículos con un nombre y un precio, ver los productos que ya ha ingresado y eliminar los de su elección, todo dentro de un menú interactivo. De esta forma, se maneja la lógica de almacenamiento y control de los artículos de manera sencilla pero organizada. Para lograr esto, se hace uso de la librería `java.util.Scanner`, que permite la interacción con el usuario a través de la lectura de datos ingresados. También se emplea la librería `java.lang`, incluida de forma implícita en cualquier programa en Java, que proporciona la base para la herencia de clases y métodos como `toString()`, el cual se sobrescribe en una de las clases del proyecto.

Archivos

Proyecto

La clase principal contiene el método `main` y es el método principal que se ejecuta. En ella se despliega un menú con cuatro opciones: agregar artículos, ver los artículos registrados, eliminar un artículo o salir de la aplicación. A través de un ciclo `while`, el menú se repite hasta que el usuario decida salir. Cada opción es gestionada mediante un `switch` que determina la acción a realizar. En este archivo se crea un objeto de la clase `Carrito` que funciona como el contenedor de los artículos que el usuario va registrando, y dependiendo de la opción elegida, se ejecutan los métodos correspondientes para modificar o consultar el contenido del carrito.

Articulo

Esta clase representa a los objetos que se almacenarán en el carrito. Cada artículo cuenta con dos atributos: `nombre` y `precio`, los cuales se asignan a través de su constructor al momento de instanciar un nuevo objeto. La clase también incorpora un método `verNombre()`, que devuelve únicamente el nombre del artículo, utilizado en particular al eliminar elementos del carrito, y un método sobrescrito `toString()`, que devuelve la información del artículo en un

formato legible para el usuario mostrando el nombre acompañado del precio, lo que facilita la presentación en consola.

Carrito

Esta clase administra la colección de artículos que el usuario va registrando. Internamente utiliza un arreglo de objetos de tipo `Articulo` con una capacidad máxima de 100 elementos y un atributo `cantidad` que controla el número actual de artículos guardados. Dispone de varios métodos que permiten gestionar el contenido: `agregar()` añade un nuevo artículo al arreglo siempre que no se haya alcanzado el límite máximo; `mostrar()` recorre el arreglo y despliega en pantalla los artículos registrados junto con su posición en el carrito, o bien informa si este se encuentra vacío; y `eliminar()` recibe la posición de un artículo y lo elimina recorriendo los elementos restantes para evitar huecos en el arreglo, además de notificar al usuario si el proceso fue exitoso o si se ingresó una posición no válida. Con ello se garantiza una manipulación sencilla pero efectiva de los objetos almacenados.

Pruebas

```
Input(Args): 1
Output(En ventana):
Introduce el nombre del articulo:
Input(Args): coca
Output(En ventana):
Introduce el precio del articulo:
Input(Args): 20
Output(En ventana):
--- Agregado

Input(Args): 2
Output(En ventana):
```

```
--- Articulos en el carrito
1. coca $20

Input(Args): 3
Output(En ventana):
--- Articulos en el carrito
1. coca $20
Que elemento deseas eliminar:
Input(Args): 1
Output(En ventana):
--- 'coca' eliminado

Input(Args): 4
Output(En ventana):
--- Saliendo
```

4. Resultados

El menú del carrito de compras permite al usuario agregar artículos, visualizarlos y eliminarlos mediante un menú interactivo en consola.

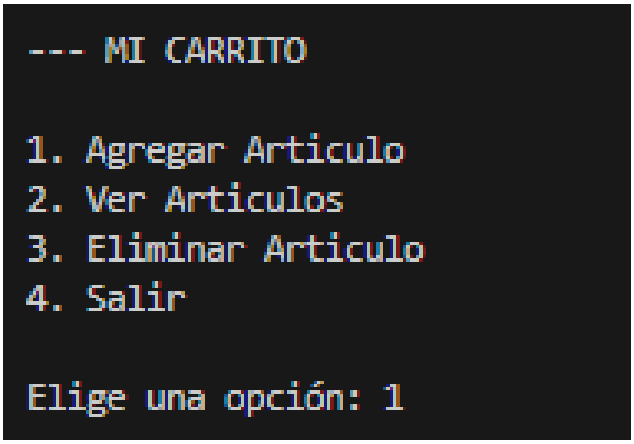

Figura 4: Ejecutando aparecerá el menú, elegimos la opción 1.

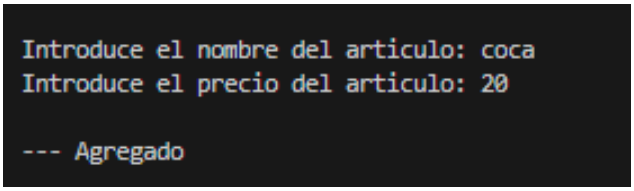

Figura 5: Introducir el nombre y el precio del artículo que se desea agregar al carrito.

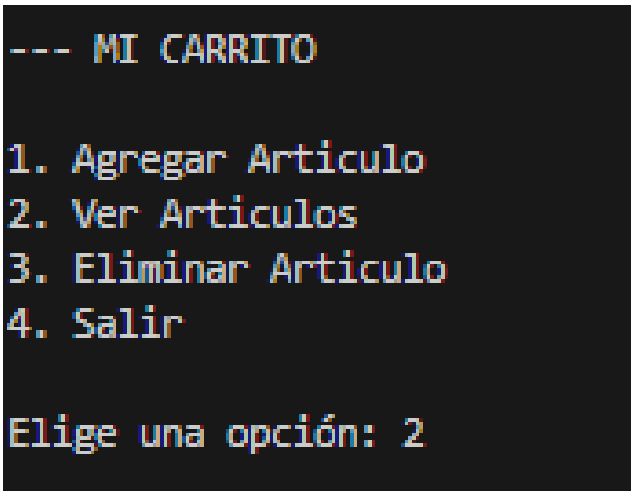

Figura 6: Aparecerá el menú de nuevo, elegimos la opción 2.

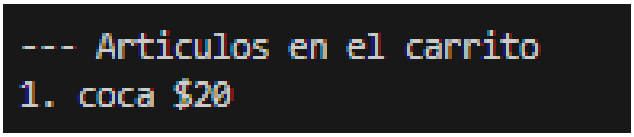

Figura 7: Se despliegan los artículos que fueron agregados previamente.

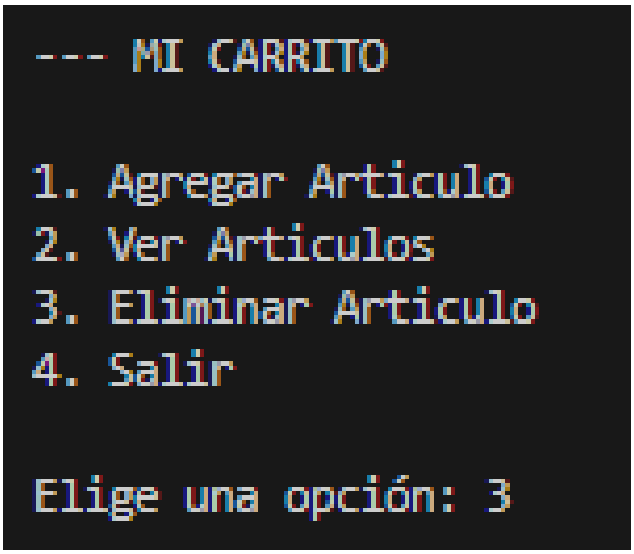

Figura 8: Nuevamente aparecerá el menú, elegimos la opción 3.

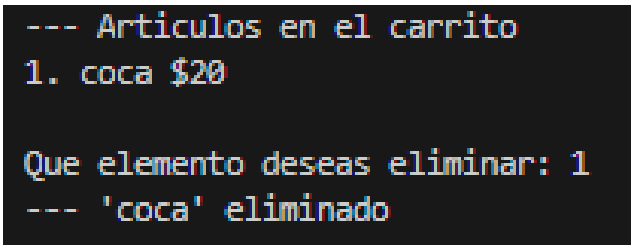

Figura 9: Se muestra la lista de los artículos de forma enumerada y se nos pregunta qué número de elemento se desea eliminar.

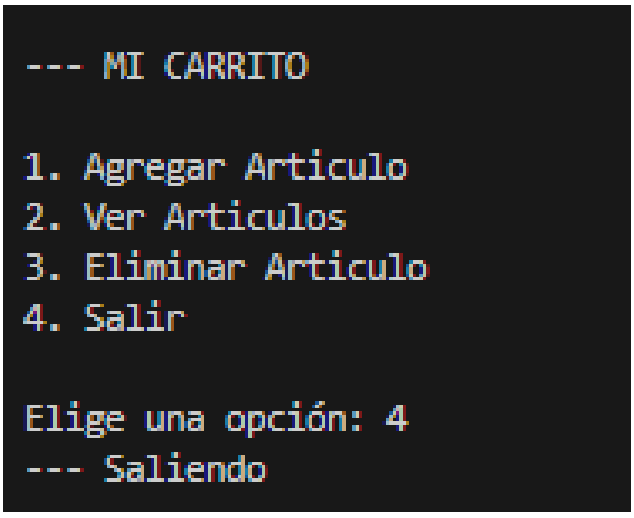

Figura 10: Para cerrar el programa, se elije la opción 4 del menú.

5. Conclusiones

El desarrollo de este proyecto permitió realizar un análisis sobre la aplicación de los conceptos teóricos de la Programación Orientada a Objetos para resolver un problema concreto, los resultados obtenidos, donde el programa fue capaz de gestionar correctamente los artículos del carrito, demuestran la importancia de una buena abstracción y modelado.

La creación de una clase Carrito fue muy importante para encapsular tanto los datos (la lista de artículos y sus precios) como las operaciones (agregar y quitar), lo que facilitó la organización del código y a su vez, el uso de una estructura de datos resultó ser un mecanismo indispensable para gestionar una colección dinámica de objetos, permitiendo que el contenido del carrito se modificara durante la ejecución del programa.

Finalmente, se comprobó que la interacción entre objetos es la base del paradigma orientado a objetos, ya que la comunicación entre el objeto Carrito y la clase principal fue lo que permitió articular la lógica completa de la aplicación. Por tanto, se concluye que los conceptos teóricos vistos en clase no son ideas abstractas, sino herramientas efectivas para la construcción de soluciones de software estructuradas y funcionales.

Referencias

[1] DataCamp. *Constructores en Java*. URL: https://www.datacamp.com/es/doc/java/constructors.

[2] Carmen Cerón (C. C.) Garnica. *1.6 Relaciones | POO en Java*. 2025. URL: https://ecosistema.buap.mx/forms/files/dspace-23/16_relaciones.html.

[3] Oracle. *Class PrintStream (System.out)*. 2024. URL: https://docs.oracle.com/javase/8/docs/api/java/io/PrintStream.html.

[4] Oracle. *Class Scanner*. 2024. URL: https://docs.oracle.com/javase/8/docs/api/java/util/Scanner.html.

[5] Oracle. *Control Flow Statements*. 2024. URL: https://docs.oracle.com/javase/tutorial/java/nutsandbolts/flow.html.

[6] Oracle. *Primitive Data Types - The boolean Type*. 2024. URL: https://docs.oracle.com/javase/tutorial/java/nutsandbolts/datatypes.html.

[7] Jhon (J.) Paul. *Relaciones entre clases*. 2020. URL: https://javajhon.blogspot.com/2020/06/relaciones.html.