



Carátula para entrega de prácticas

Facultad de Ingeniería

Laboratorio de docencia

Laboratorios de computación salas A y B

Profesor(a): René Adrián Dávila Pérez

Asignatura: Programación Orientada a Objetos

Grupo: 7

No. de práctica(s): 5,6

Integrante(s): 320065570
425133840
423020252
322229916
425032224

No. de lista o brigada: 3

Semestre: 2026-1

Fecha de entrega: 27/08/2025

Observaciones:

CALIFICACIÓN: _____

Índice

| | |
|------------------|---|
| 1. Introducción | 2 |
| 2. Marco Teórico | 3 |
| 3. Desarrollo | 4 |
| 4. Resultados | 5 |
| 5. Conclusiones | 7 |

1. Introducción

En esta práctica se implementará el encapsulamiento y empaquetamiento en un código previamente proporcionado. Haciendo las modificaciones necesarias para su óptimo funcionamiento.

El objetivo principal de esta práctica es comprender cómo funciona el encapsulamiento y empaquetamiento para ver la forma en la que podemos integrarlos en un proyecto real o a mayor escala. Por otro lado, se espera identificar las ventajas que nos dan en cuanto a la privacidad y seguridad en el código.

Es necesario e importante resolver este problema porque en el ámbito laboral es sencillo que sea frágil y vulnerable nuestro desarrollo sin aplicar ninguna de estas medidas, permitiendo que alguien con conocimiento e intenciones maliciosas sea capaz de penetrar y obtener información confidencial sobre lo desarrollado; por lo que es inevitable tener que implementar en proyectos importantes medidas como estas para proteger nuestro trabajo.

2. Marco Teórico

En esta práctica se utilizó la función encapsulamiento, que es un principio fundamental de la POO que consiste en agrupar los datos (atributos) y los métodos (comportamientos) relacionados con un objeto dentro de una sola unidad, llamada clase. Además, se controla el acceso a estos datos y métodos, permitiendo únicamente interacciones controladas desde el exterior de la clase [1]. Fue necesario usar métodos de acceso para definir la visibilidad de los atributos, al igual que métodos de acceso que nos permiten tener acceso a los atributos privados de un objeto y que son métodos públicos clasificados en consultores (get) y modificadores (set).

Por otro lado, se usó el empacamiento para agrupar clases relacionadas. Imagínalo como una carpeta en un directorio de archivos . Usamos paquetes para evitar conflictos de nombres y escribir código más fácil de mantener. Los paquetes se dividen en dos categorías:

- Paquetes integrados (paquetes de la API de Java)
- Paquetes definidos por el usuario (crea tus propios paquetes)

En nuestro caso usamos paquetes definidos por el usuario. Para crear nuestro propio paquete, debemos comprender que Java utiliza un directorio del sistema de archivos para almacenarlos. Al igual que las carpetas de nuestro ordenador. Ejemplo:

```
root
```

```
  mypack
```

```
    MyPackageClass.java [2]
```

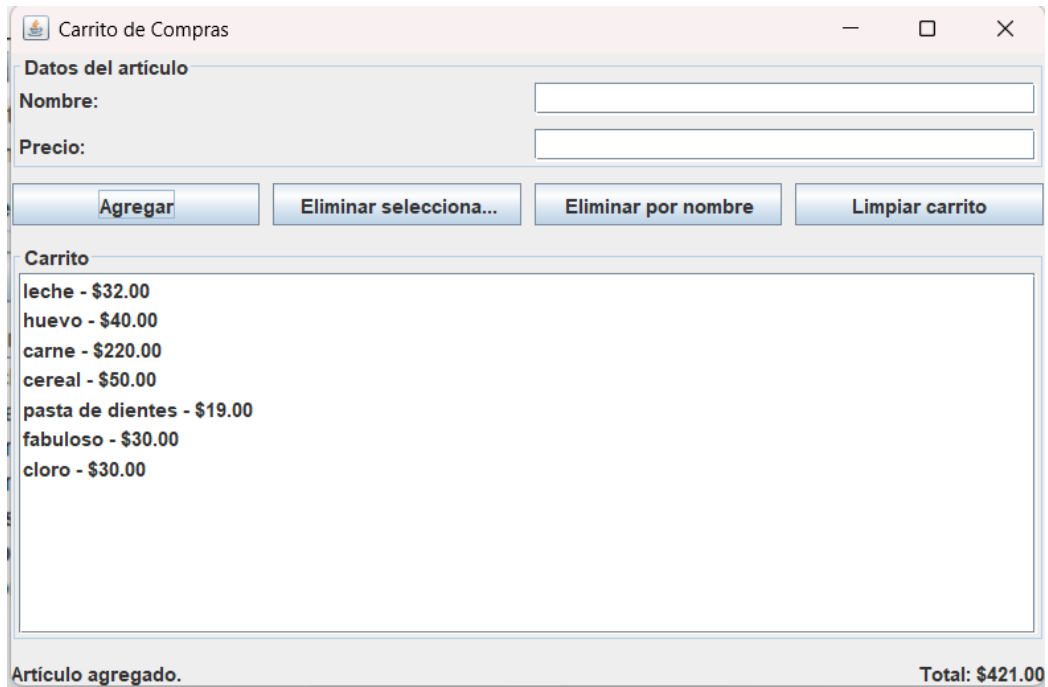
3. Desarrollo

Para hacer el empacamiento, se creó una estructura de carpetas anidadas siguiendo el orden `mx/unam/fi/poo/p56`, y dentro de esa ruta se colocaron los archivos `.java`. En cada clase se agregó al inicio la línea `package mx.unam.fi.poo.p56;` para que el compilador reconozca correctamente que pertenecen a ese paquete.

En la clase `Artículos` utilizamos el modificador de acceso `private` para los atributos `nombre` y `precio` y posteriormente inicializamos el objeto usando modificadores (`set`) para asignarles valores a las variables privadas. Posteriormente usamos métodos de acceso `setters` y `getters` para acceder y modificar de forma controlada el `nombre` y el `precio`.

En la clase `Carrito` de igual manera usamos `private` en el atributo `artículos` y en el método `agregar` usamos el método modificador para acceder al atributo privado y obtener su valor y utilizamos de igual manera `setter` para poder eliminar un elemento por `nombre` y así de igual manera en los métodos para calcular el total, saber si el carrito está vacío y poder ver el número de artículos totales.

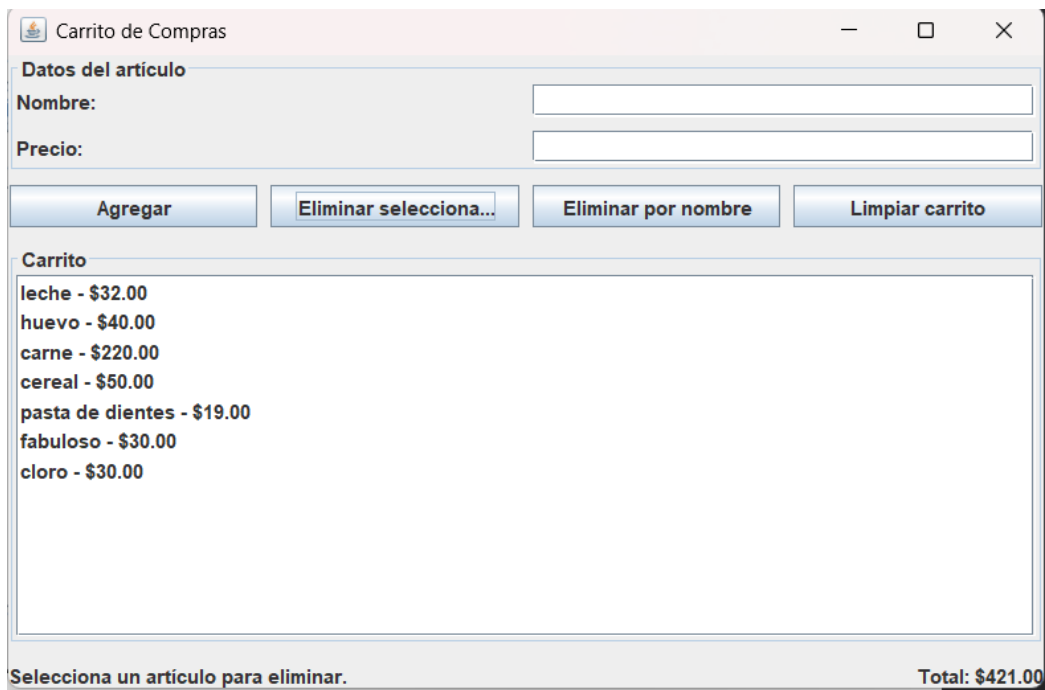
4. Resultados



The screenshot shows a window titled "Carrito de Compras" with a standard Windows interface (minimize, maximize, close buttons). The window is divided into several sections:

- Datos del artículo:** Contains two input fields labeled "Nombre:" and "Precio:".
- Buttons:** Below the input fields are four buttons: "Agregar", "Eliminar selecciona...", "Eliminar por nombre", and "Limpiar carrito".
- Carrito:** A list box containing the following items:
 - leche - \$32.00
 - huevo - \$40.00
 - carne - \$220.00
 - cereal - \$50.00
 - pasta de dientes - \$19.00
 - fabuloso - \$30.00
 - cloro - \$30.00
- Status Bar:** At the bottom, it displays "Artículo agregado." on the left and "Total: \$421.00" on the right.

Figura 1: Asignación de 7 artículos a la lista a partir de su nombre y precio.



This screenshot shows the same "Carrito de Compras" window as Figure 1, but with a different status bar message. The list of items in the "Carrito" section remains the same:

- leche - \$32.00
- huevo - \$40.00
- carne - \$220.00
- cereal - \$50.00
- pasta de dientes - \$19.00
- fabuloso - \$30.00
- cloro - \$30.00

The status bar at the bottom now displays "Selecciona un artículo para eliminar." on the left and "Total: \$421.00" on the right.

Figura 2: Intento eliminar un artículo por selección, al no haber nada seleccionado surge un aviso en la parte inferior de la ventana.

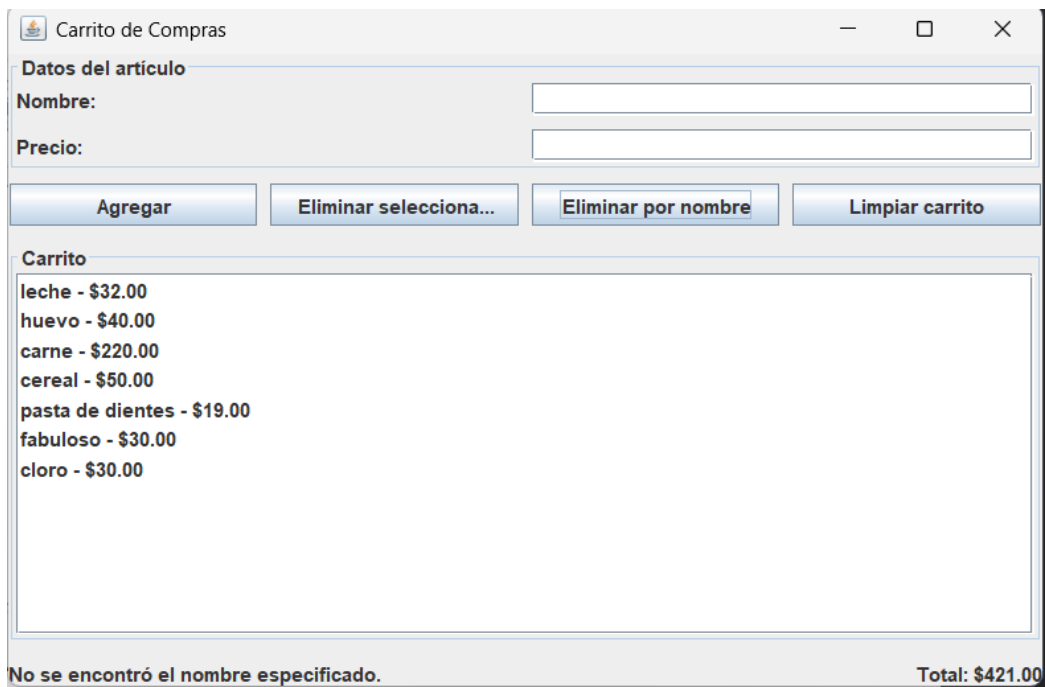


Figura 3: Intento eliminar un artículo por nombre, al no haber nada seleccionado surge un aviso en la parte inferior de la ventana.

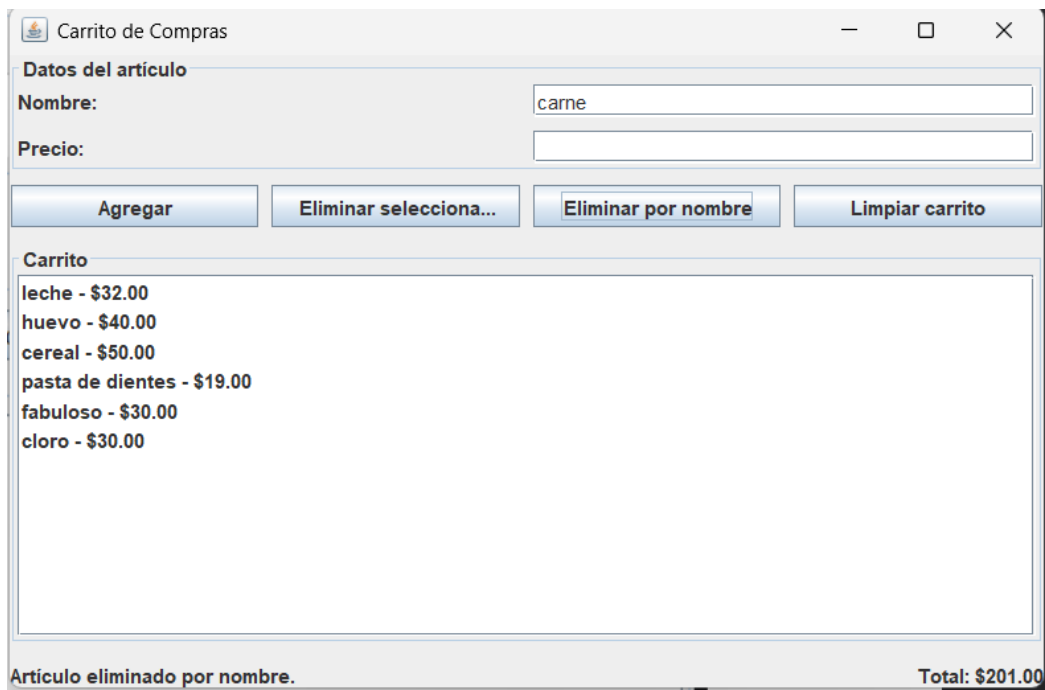


Figura 4: Eliminación de un artículo de la lista por nombre, se eliminó el artículo carne.

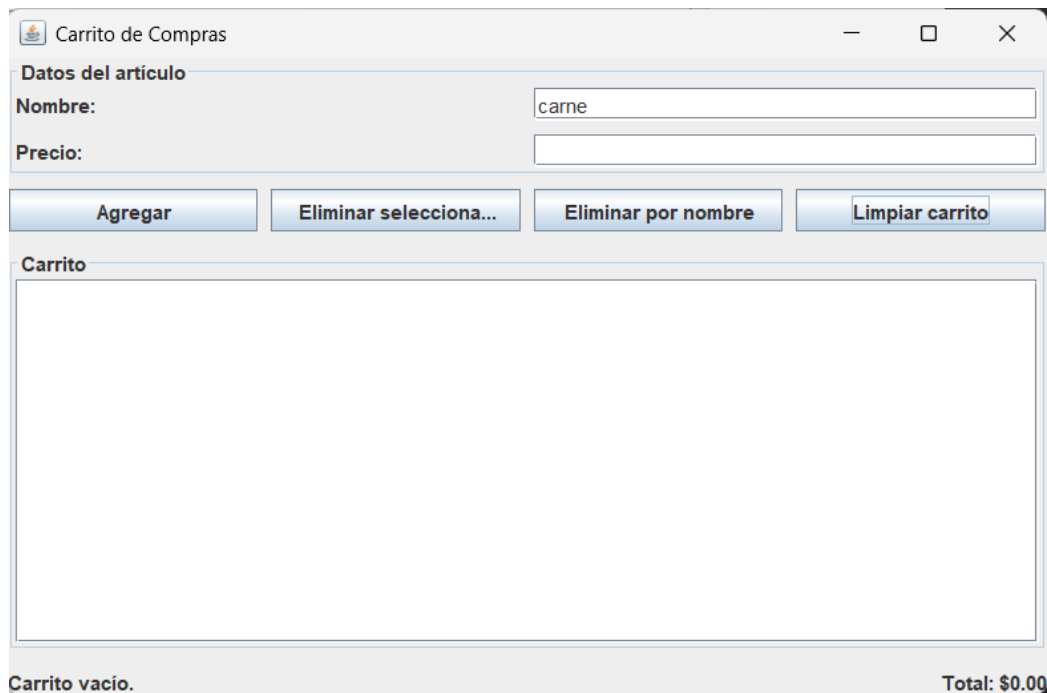


Figura 5: Eliminación de la lista.

5. Conclusiones

Una vez concluido el ajuste a la codificación que teníamos como base, con respecto a los requerimientos establecidos como lo fue el encapsulamiento y el empaquetado explicado en la sección desarrollo del presente escrito, se prosiguió a realizar el compilado y ejecución de los códigos para así comprobar que se tenga un buen funcionamiento que concuerde con lo esperado.

En la sección resultado se presentan las diferentes acciones que pueden ser realizadas con el código, como eliminar artículos como se puede ver en la figura 1, eliminar artículos como se puede observar en la figura 4 y borrar la lista completa como puede verse en la figura 5; todo esto dentro de una ventana emergente, permitiendonos ver de esta forma el correcto funcionamiento del código.

En conclusión, se asegura la correcta implementación del encapsulamiento y empaquetamiento en un código, asegurandonos del correcto funcionamiento de este, observando el comportamiento de estas dos funciones para poder hacer uso de ellas en trabajos futuros.

Referencias

- [1] J.L. Blasco. *Introducción a POO en Java: Encapsulamiento*. Nov. de 2023. URL: `OpenWebinars.net.%20https://openwebinars.net/blog/introduccion-a-poo-en-java-encapsulamiento/`.
- [2] *W3Schools.com*. URL: `https://www.w3schools.com/java/java_packages.asp`.