Icono

Descripción generada automáticamente

**Ejercicio 1**

**PC 3**

**Dilian Anabel HURTADO PONCE**

Semana: 15

**Integrantes:**

Roberto Agustín Mejía Collazos

21/11/24

**Docente:**

***Una Tienda de Libros***

***Command.java (Interface)***

***Interfaz de usuario gráfica, Texto, Aplicación, Chat o mensaje de texto

Descripción generada automáticamente***

***AddToCardCommand.java***

***Captura de pantalla de computadora

Descripción generada automáticamente***

***RemoveFromCartCommand***

***Captura de pantalla de un celular

Descripción generada automáticamente***

***Libro.java***

***Captura de pantalla de computadora

Descripción generada automáticamente***

***ItemCompra.java***

***Captura de pantalla de un celular

Descripción generada automáticamente***

***CarroCompra.java***

***Texto

Descripción generada automáticamente***

***Main.java***

***Captura de pantalla de computadora

Descripción generada automáticamente***

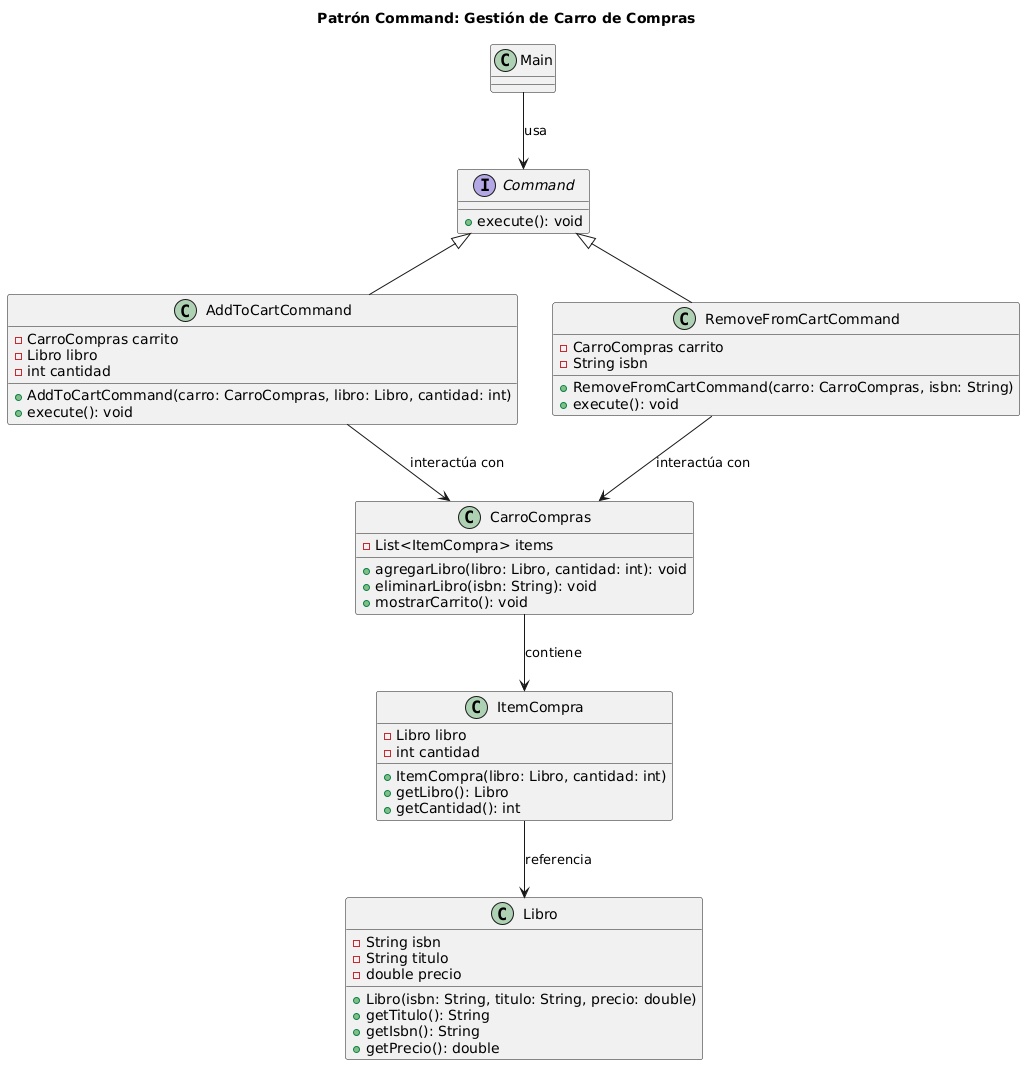
***Texto

Descripción generada automáticamente***

***Explicación del Patrón Command Aplicado***

El patrón Command encapsula las solicitudes como objetos, permitiendo parametrizar acciones, registrar su historial y proporcionar funcionalidad de deshacer/rehacer. En el caso presentado:

1. Actores principales:
   * Cliente (Main): Solicita la ejecución de comandos.
   * Receptor (CarroCompras): Contiene la lógica principal para gestionar libros.
   * Comandos concretos (AddToCartCommand, RemoveFromCartCommand): Encapsulan las acciones que pueden realizarse sobre el receptor.
2. Ventajas del uso del patrón Command en este caso:
   * Flexibilidad: Cada acción está encapsulada, facilitando la adición de nuevas operaciones.
   * Historial: Permite llevar un registro de los comandos ejecutados.
   * Extensibilidad: Podemos agregar comandos como "vaciar carrito" o "mostrar totales" sin modificar la lógica existente.
3. Flujo del programa:
   * El cliente (Main) crea instancias de comandos con parámetros específicos (como el carrito y los libros).
   * Los comandos son ejecutados, invocando métodos en el receptor (CarroCompras).

***Diagrama UML***

***Explicación del UML***

1. Relaciones principales:
   * Command es una interfaz que define el contrato común para los comandos.
   * AddToCartCommand y RemoveFromCartCommand implementan la interfaz Command.
   * El cliente (Main) utiliza los comandos para interactuar con el receptor (CarroCompras).
2. Dependencias:
   * Los comandos (AddToCartCommand, RemoveFromCartCommand) dependen del receptor (CarroCompras).
   * CarroCompras gestiona los elementos del carrito como una lista de ItemCompra.
   * Cada ItemCompra tiene una referencia a un Libro.
3. Beneficio visual:
   * El diagrama organiza claramente los roles de cada clase, mostrando cómo encapsulan sus responsabilidades y colaboran para lograr la funcionalidad.