|  |  |
| --- | --- |
| Patrones de Diseño Java | Udemy  PATRONES DE DISEÑO  Implementación de los patrones de diseño. | Descripción breve  Este documento contiene las especificaciones sobre la implementación de los patrones de diseño solicitados para la elaboración del proyecto integrador de la asignatura de diseño de software.  Autores:   * Emmanuel Chablé Collí. * Eusebio Ajas Santos. * Nicolás Canul Ibarra. |

Contenido

[Patrón de diseño: Decorator. 3](#_Toc42673732)

[Patrón de diseño: Iterator. 4](#_Toc42673733)

[Patrón de diseño: Abstract Factory. 5](#_Toc42673734)

[Patrón de diseño: Singleton. 6](#_Toc42673735)

[Patrón de diseño: State. 7](#_Toc42673736)

[Patrón de diseño: Observer. 9](#_Toc42673737)

# Patrón de diseño: Decorator.

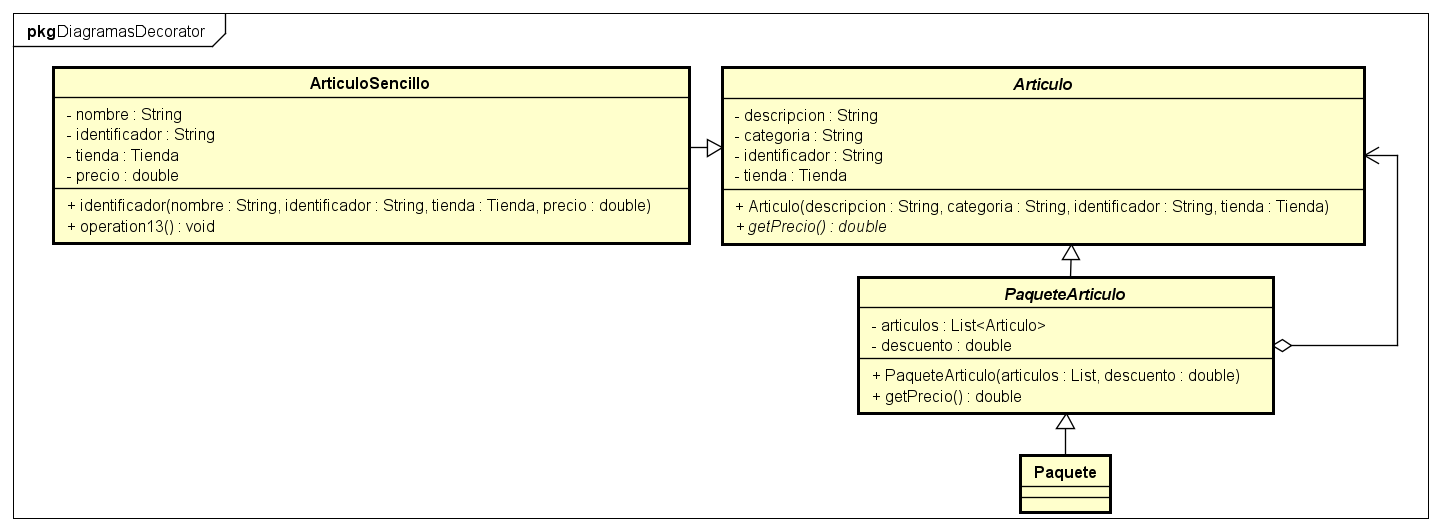
Diagrama de clases:

Ilustración 1. Diagrama de clases del patrón de diseño **Decorator**.

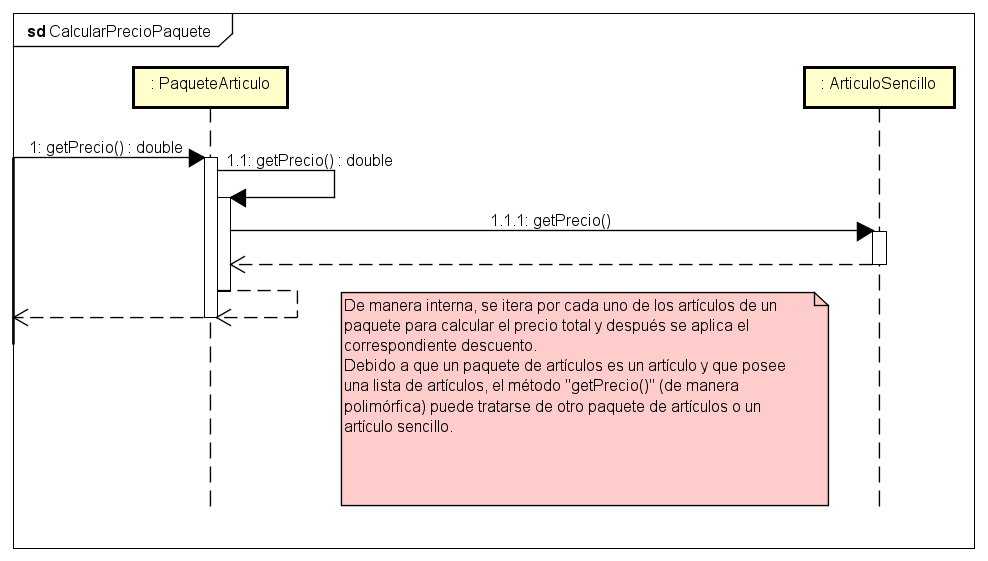
Diagrama de secuencia:

Ilustración 2. Diagrama de secuencia para el cálculo del precio de un paquete de artículos.

Observaciones:

Para esta situación, el uso del patrón **Decorator** nos es muy útil para crear paquetes de artículos e incluso paquetes de paquetes, ya que cada paquete es un artículo. De igual forma, el precio para un paquete se puede calcular de una manera muy sencilla siguiendo el diseño que este patrón nos ofrece.

# Patrón de diseño: Iterator.

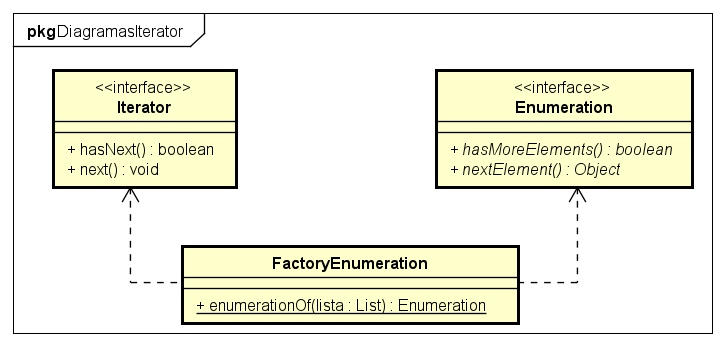
Diagrama de clases:

Ilustración 3. Diagrama de clases para el patrón de diseño **Iterator**.

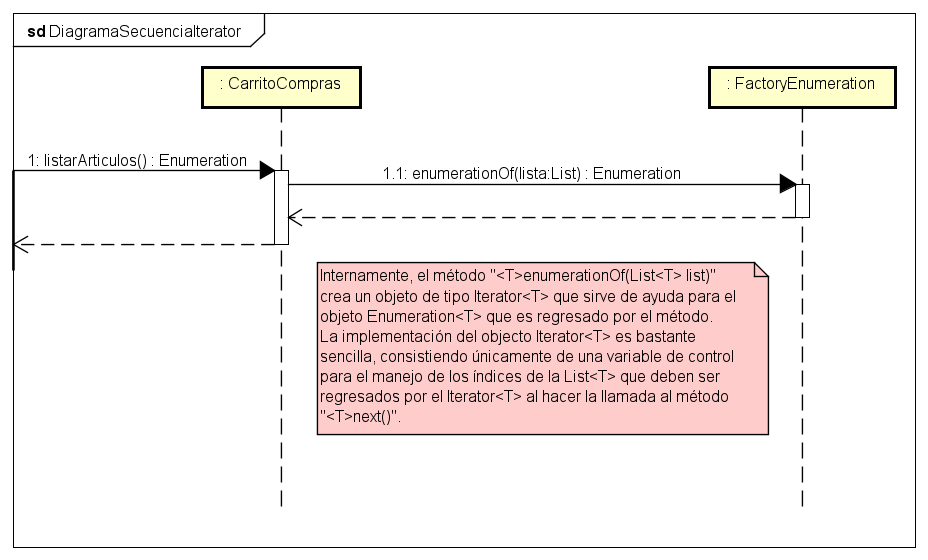
Diagrama de secuencia:

Ilustración 4. Diagrama de secuencia para la creación de objeto de tipo **Enumeration**<T> a partir de una List<T>.

Observaciones:

Los objectos **Enumeration**<T> son utilizados para recorrer una colección de elementos en algún orden no especificado. Para nuestro caso, estos objectos se usan para iterar sobre algún conjunto de datos (clientes o artículos) y posteriormente ir mostrándolos en pantalla (tablas o listas).

# Patrón de diseño: Abstract Factory.

Diagrama de clases:

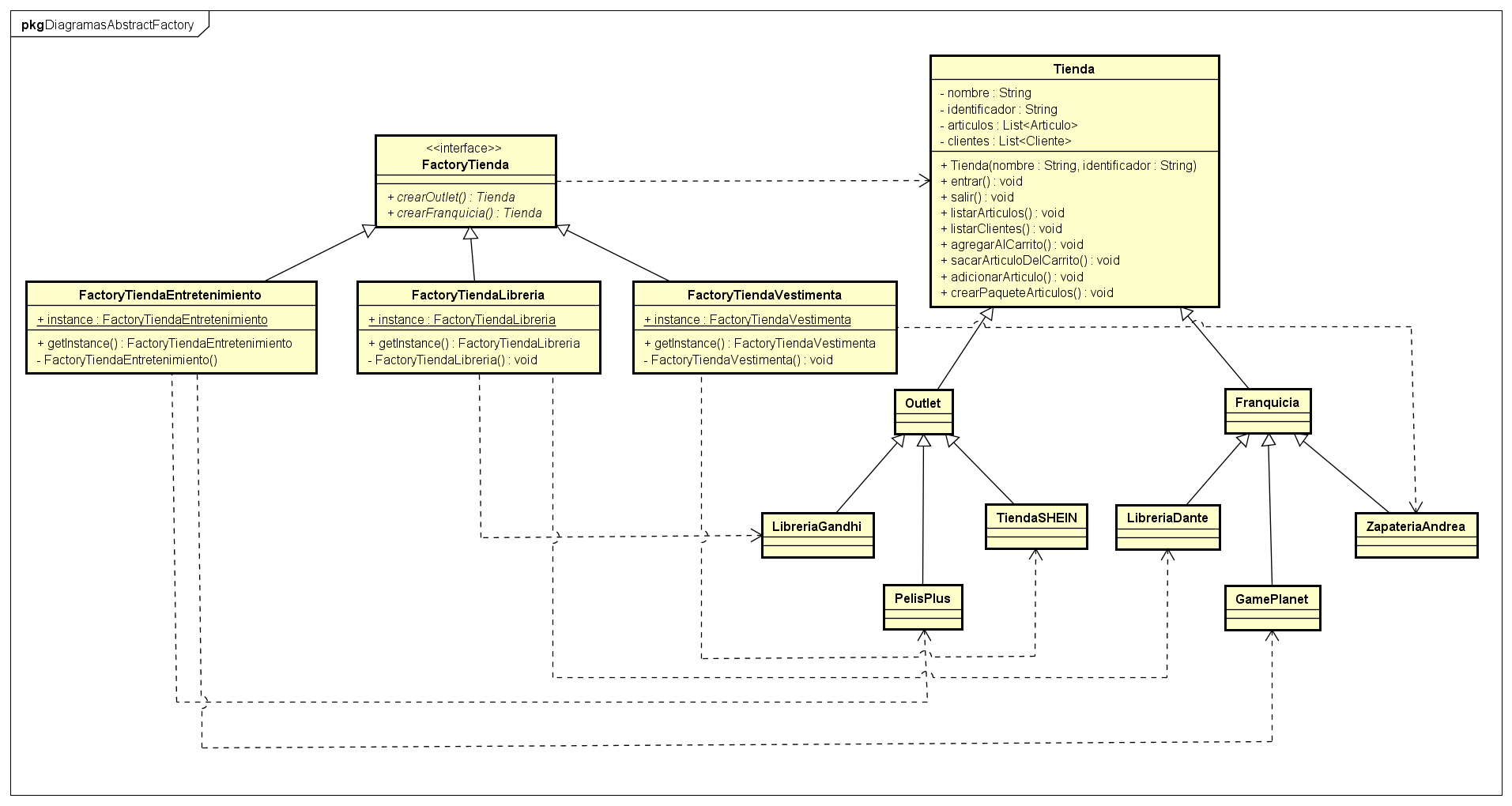


Ilustración . Diagrama de clases del patrón **Abstrac Factory.**

Diagrama de secuencia:

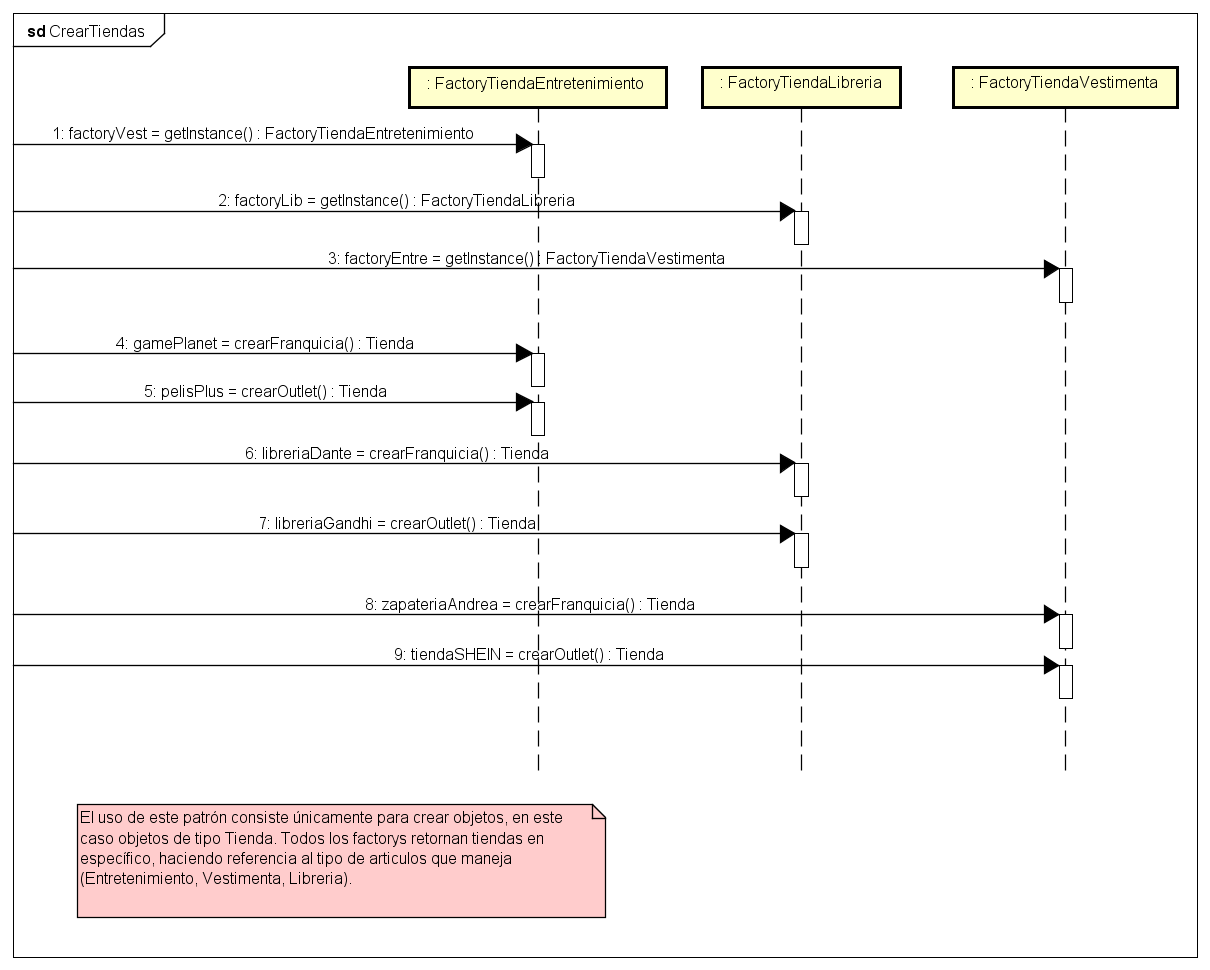


Ilustración . Diagrama de secuencia del patrón Abstrac Factory.

Observaciones:

Este patrón de diseño nos facilita mucho la creación de familias de objectos. Para esta situación en particular en la que tenemos objetos que no dependen entre sí, la creación de este conjunto de tiendas es muy sencilla.

# Patrón de diseño: Singleton.

Diagrama de clases:

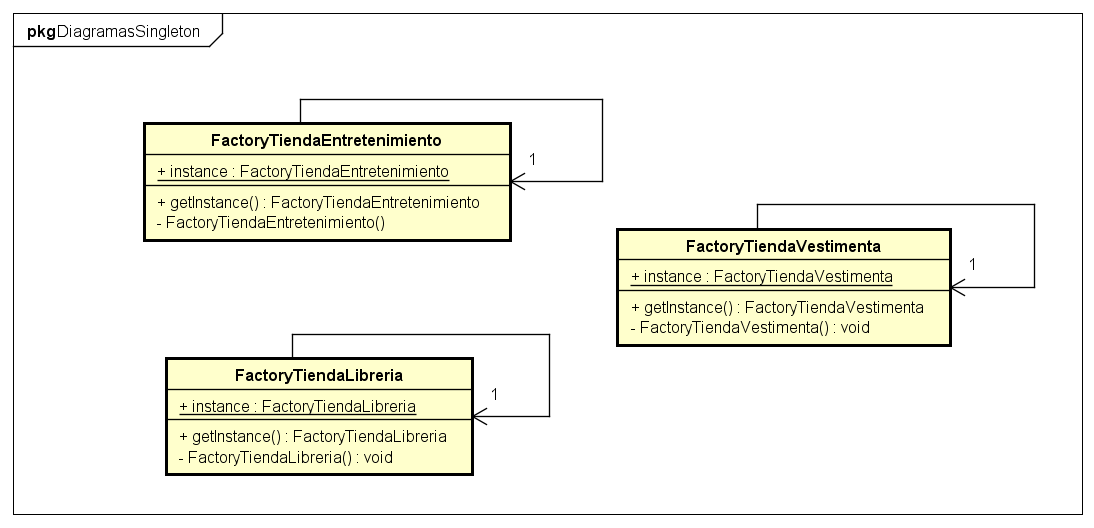


Ilustración . Diagrama de clases **Singleton**

Diagrama de secuencia:

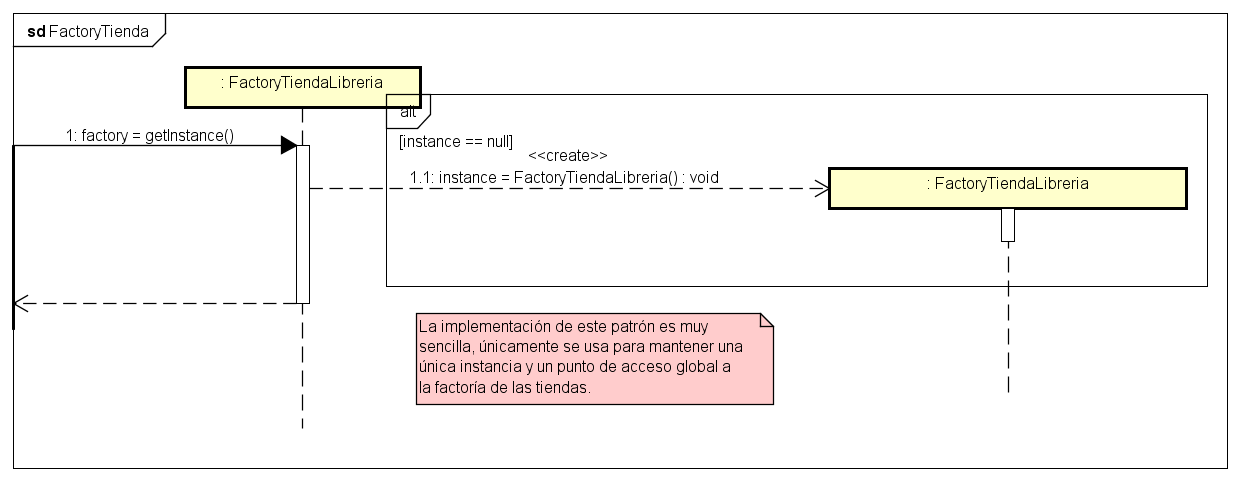


Ilustración . Diagrama de clases Singleton

# Patrón de diseño: State.

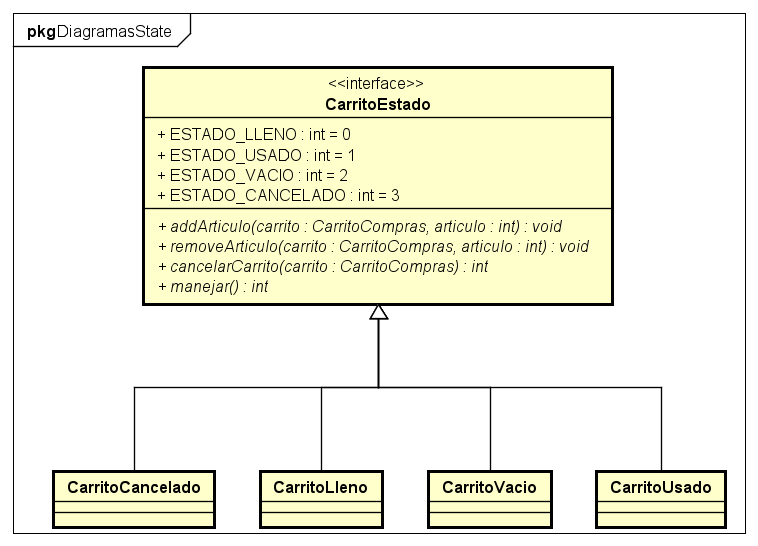
Diagrama de clases:

Ilustración 9. Diagrama de clases para el patrón de diseño **State**.

Diagrama de secuencia:

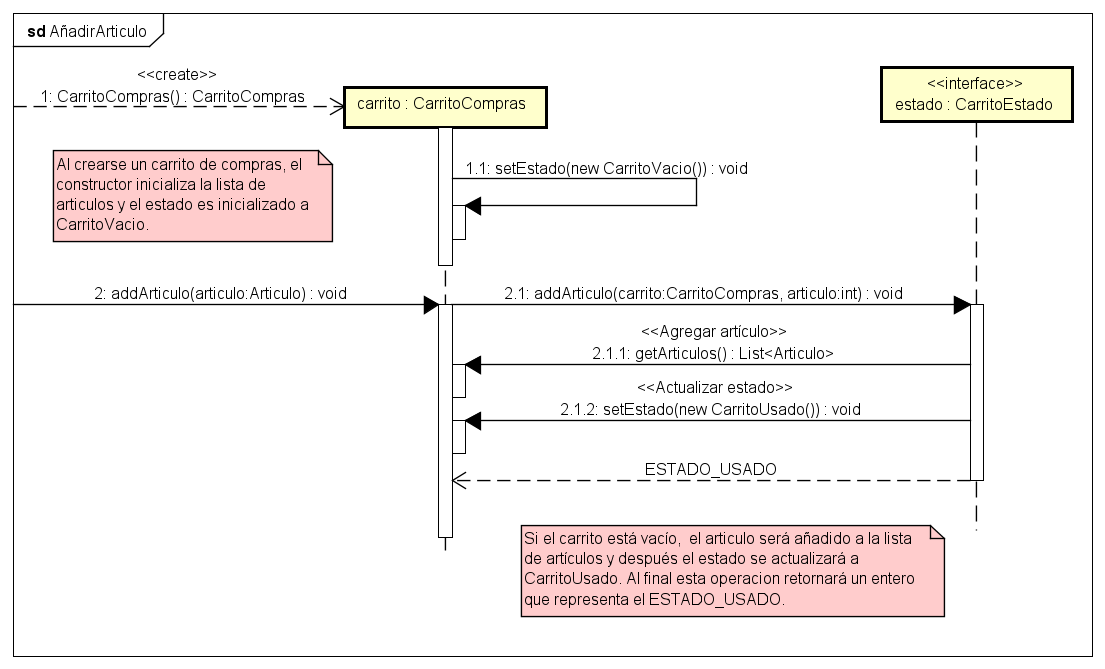


Ilustración 10. Diagrama de secuencia para las acciones que se desencadenan cuando se añade un artículo al carrito de compras.

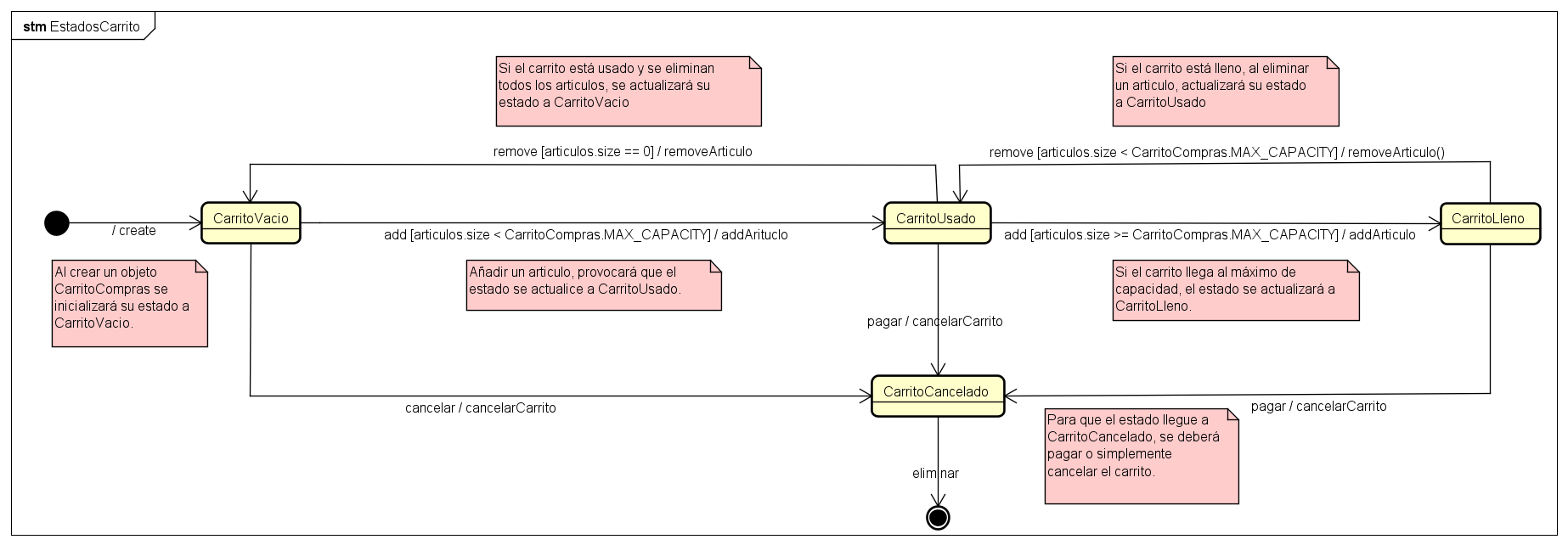
Diagrama de estados:

Ilustración 11. Diagrama de estados del carrito de compras.

Explicación:

Este patrón es utilizado para la clase CarritoCompras el cual nos ayuda a procesar los estados que el carrito tiene cuando se ejecuta una acción. Este patrón también es utilizado para la parte gráfica y, uniéndolo con el patrón **Observer**, nos permite actualizar elementos visuales que representan al carrito.

# Patrón de diseño: Observer.

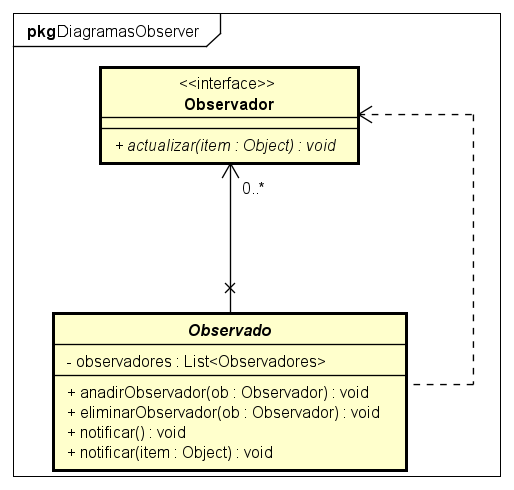
Diagrama de clases:

Ilustración 12. Diagrama de clases para el patrón de diseño **Observer**.

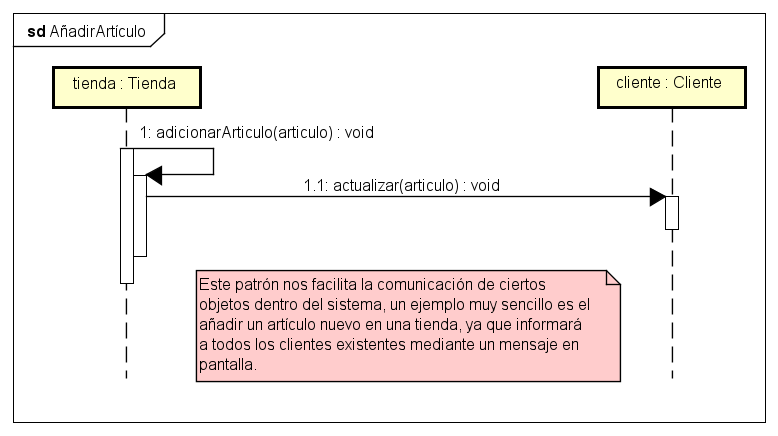
Diagrama de secuencia:

Ilustración 13. Diagrama de clases para un uso concreto del patrón dentro del sistema.

Observaciones:

Este patrón de diseño no solo se usa en este contexto específico del sistema, sino también para mandar notificaciones entre ventanas para responder de cierta manera ante las acciones del usuario; dibujar el estado del carrito en pantalla según los cambios que se hagan dentro de él, o simples acciones internas que se realizan dentro de las tablas para actualizar los datos.