Ejercicio 1: Patrón Template

\*\*Enunciado:\*\*

Crea una clase base abstracta `Bebida` que define un método plantilla `preparar()`. Este método debe llamar a una serie de métodos abstractos que representan los pasos para preparar una bebida (por ejemplo, `hervirAgua()`, `añadirIngredientes()`, `mezclar()`, `servir()`). Luego, crea dos clases derivadas `Cafe` y `Te` que implementen estos métodos abstractos de manera específica para cada bebida.

Ejercicio 2: Patrón Command

\*\*Enunciado:\*\*

Implementa un sistema de control remoto para un televisor. Define una interfaz `Comando` con un método `ejecutar()`. Luego, crea clases concretas que implementen esta interfaz para diferentes acciones del televisor, como `EncenderTelevisor`, `CambiarCanal` y `ApagarTelevisor`. Finalmente, crea una clase `ControlRemoto` que pueda almacenar y ejecutar estos comandos.

Ejercicio 3: Patrón Cadena de Responsabilidad

\*\*Enunciado:\*\*

Desarrolla un sistema de manejo de solicitudes de soporte técnico. Define una interfaz `Manejador` con un método `manejarSolicitud(Solicitud solicitud)`. Luego, crea clases concretas que implementen esta interfaz para diferentes niveles de soporte, como `SoporteBasico`, `SoporteIntermedio` y `SoporteAvanzado`. Cada manejador debe decidir si puede manejar la solicitud o pasarla al siguiente manejador en la cadena.

Ejercicio 4: Patrón Observer

\*\*Enunciado:\*\*

Implementa un sistema de notificación de actualizaciones de un blog. Define una interfaz `Observador` con un método `actualizar(String mensaje)`. Luego, crea una clase `Blog` que mantenga una lista de observadores y notifique a todos los observadores cuando se publique un nuevo artículo. Finalmente, crea clases concretas que implementen la interfaz `Observador` para diferentes tipos de suscriptores, como `SuscriptorEmail` y `SuscriptorSMS`.

---