Patrones de Diseño: State

Juan Diego Conteras Yepes 20211020069

Introducción

El patrón State permite que un objeto cambie su comportamiento cuando su estado interno se modifica, haciendo parecer como si el objeto cambiara su clase

Comportamiento Dinámico

El objeto actúa diferente. Su comportamiento depende del estado actual.

Encapsula Estados

Cada estado es una clase propia. Esto aísla su lógica específica.

Código Limpio

Elimina condicionales grandes. El código es más legible y fácil de mantener.

Escenario del Problema

Para la explicación del patrón presentamos un sistema de impresora. Su comportamiento depende de su estado actual. Cada estado define una respuesta única a los comandos.

Lista para imprimir

Recibe comandos para iniciar trabajos.

Imprimiendo

Solo permite una impresión al tiempo

Sin papel

Solicita recarga; no puede imprimir.

Error de impresión

Bloquea cualquier tipo de operación hasta solucionarse

Abordando el Problema

Sin el patrón State, el manejo de estados suele ser con grandes estructuras condicionales, o con Case o switch a lo largo del código. Esto lleva a un código complejo y difícil de manejar. Cada nueva regla extiende y complica el sistema.

Lógica Centralizada

Todas las transiciones de estado se manejan en un solo lugar.

Condicionales Anidados

Se usan extensos bloques "if-else if" o "switch" para cada estado.

Código Frágil

Añadir nuevos estados o comportamientos es complicado y propenso a errores.

Solución con el Patrón State

Desacoplamiento

Cada estado es una clase separada. Su lógica está encapsulada.

Extensibilidad

Añadir nuevos estados es sencillo. No modifica el código existente.

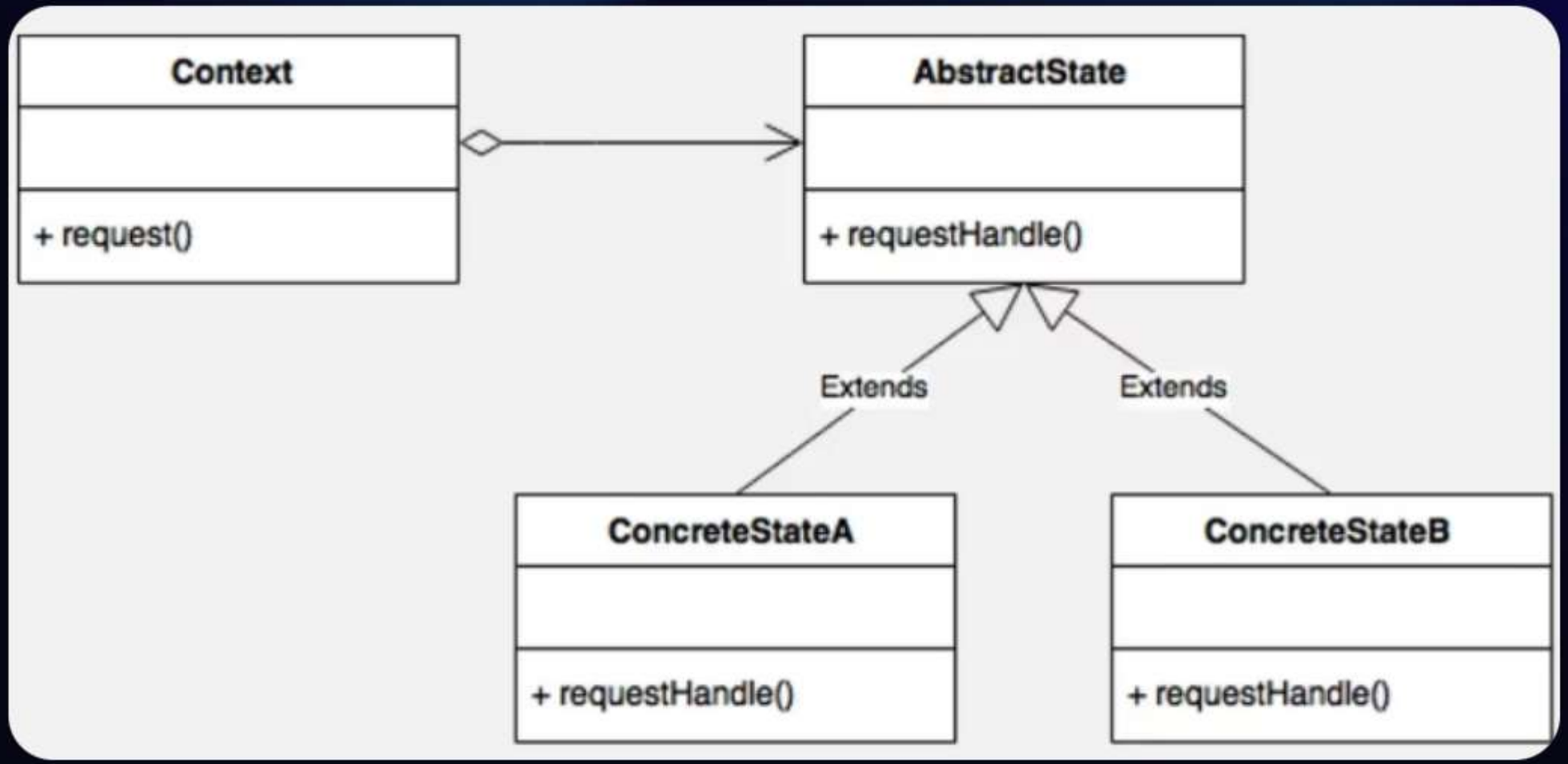
Código Limpio

Elimina grandes estructuras condicionales. Mejora la legibilidad.

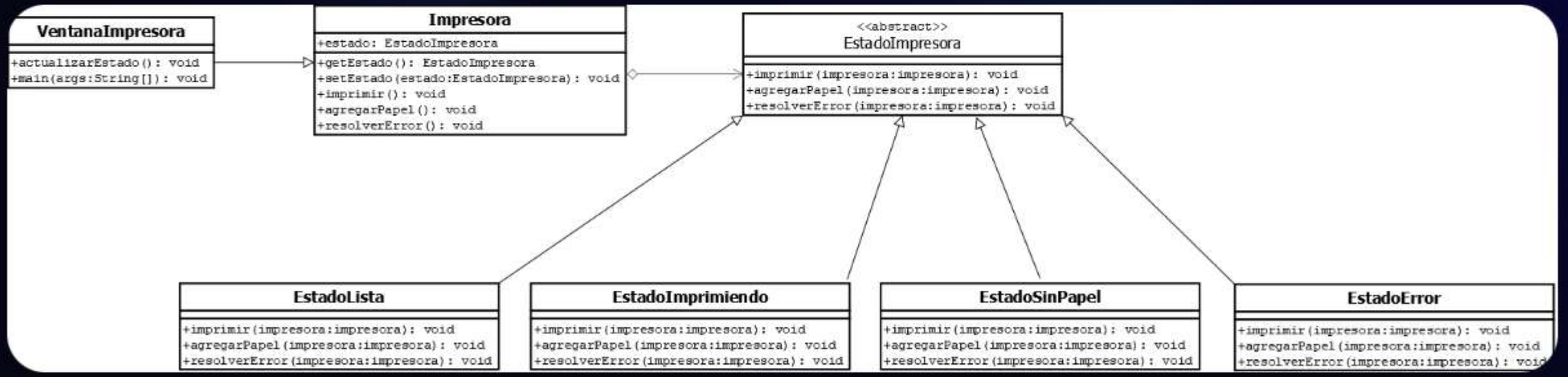
Transiciones Claras

El estado actual define el comportamiento.

UML del Patrón



UML de la Solución



Aplicaciones Prácticas del Patrón State

Máquinas de Estado de TCP/IP

Gestiona las fases de una conexión de red. Cada estado define sus acciones y transiciones.

Comportamiento en Videojuegos

Controla las acciones de personajes o elementos. Por ejemplo, "caminando" o "atacando".

Interfaces de Usuario (UI)

Maneja la interacción con los componentes. Un botón está "activo" o "desactivado".

Manejo de Servidores

Controla los estados de un servidor, revisando si este puede estar ocupado, inciado, saturado, en demanda o en cualquier otro estado

Ascensores

En este caso, el ascensor puede encontrarse en uno de tres estados: Subiendo, Bajando y Detenido.

Referencias

- https://refactoring.guru/es/design-patterns/state
- https://danielggarcia.wordpress.com/2014/05/20/patrones-de-comportamiento-v-patron-state/
- https://devexpert.io/blog/state-patrones-diseno
- https://reactiveprogramming.io/blog/es/patrones-de-diseno/state