2

(VIII)

State

(Patrones de diseño)

Diseño del Software

Grado en Ingeniería Informática del Software

Curso 2017-2018

Escuela de Ingeniería Informática
Universidad de Oviedo

César Acebal

State
(Estado)

- **Patrón de comportamiento (ámbito de objetos)**
- **Propósito:**

> *Permite a un objeto alterar su comportamiento cuando cambia su estado interno. Parecerá como si el objeto hubiera cambiado su clase.*

- **También conocido como:**
  - Estados como objetos

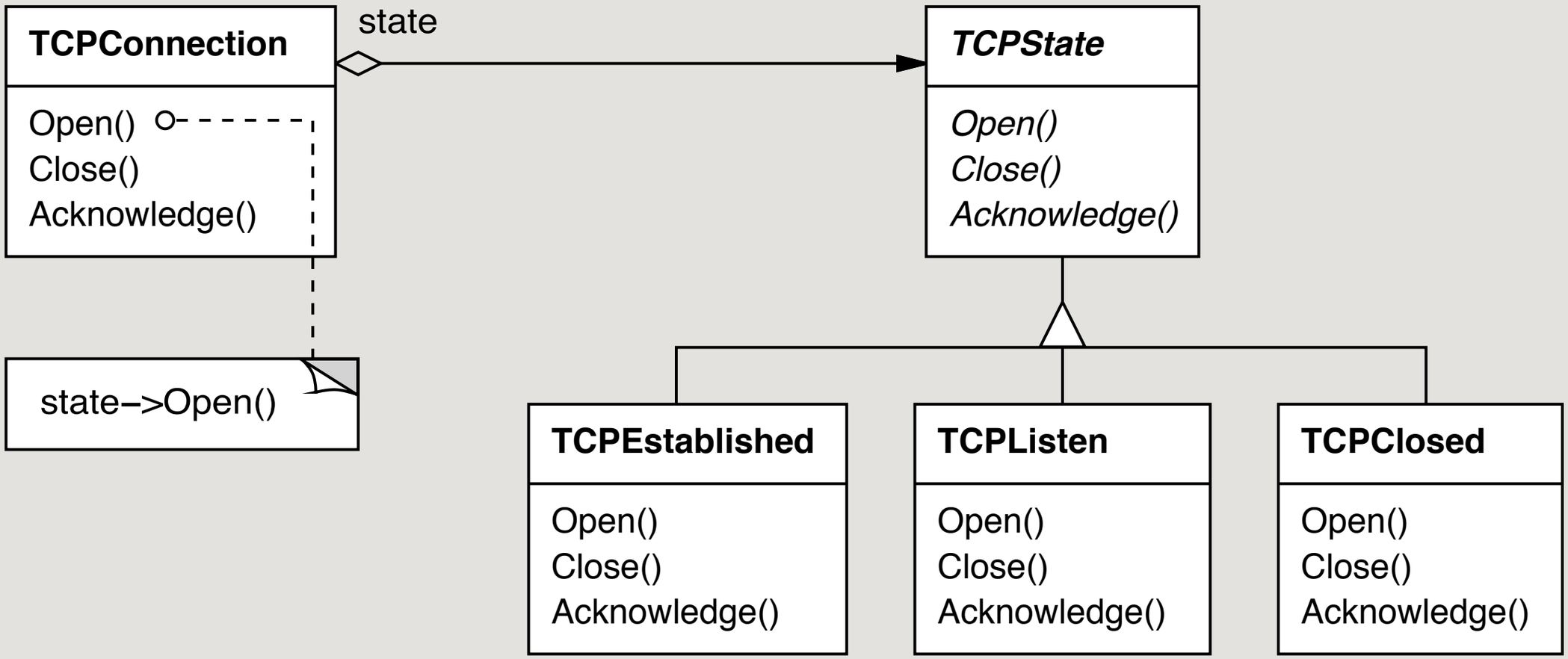
Motivación

- **Una conexión de red es representada en una implementación de TCP como `TCPConnection`**
- **La conexión puede estar en uno de los siguientes estados:**
  - Abierta
  - Escuchando
  - Cerrada

Cuando un objeto TCPConnection recibe peticiones de otros objetos debe responder de modo distinto dependiendo de su estado actual. Por ejemplo, si se solicita establecer una conexión que ya está abierta no hará lo mismo que si la conexión está cerrada.

Y, naturalmente, como siempre que sea posible, queremos eliminar la típica lógica condicional.

Motivación

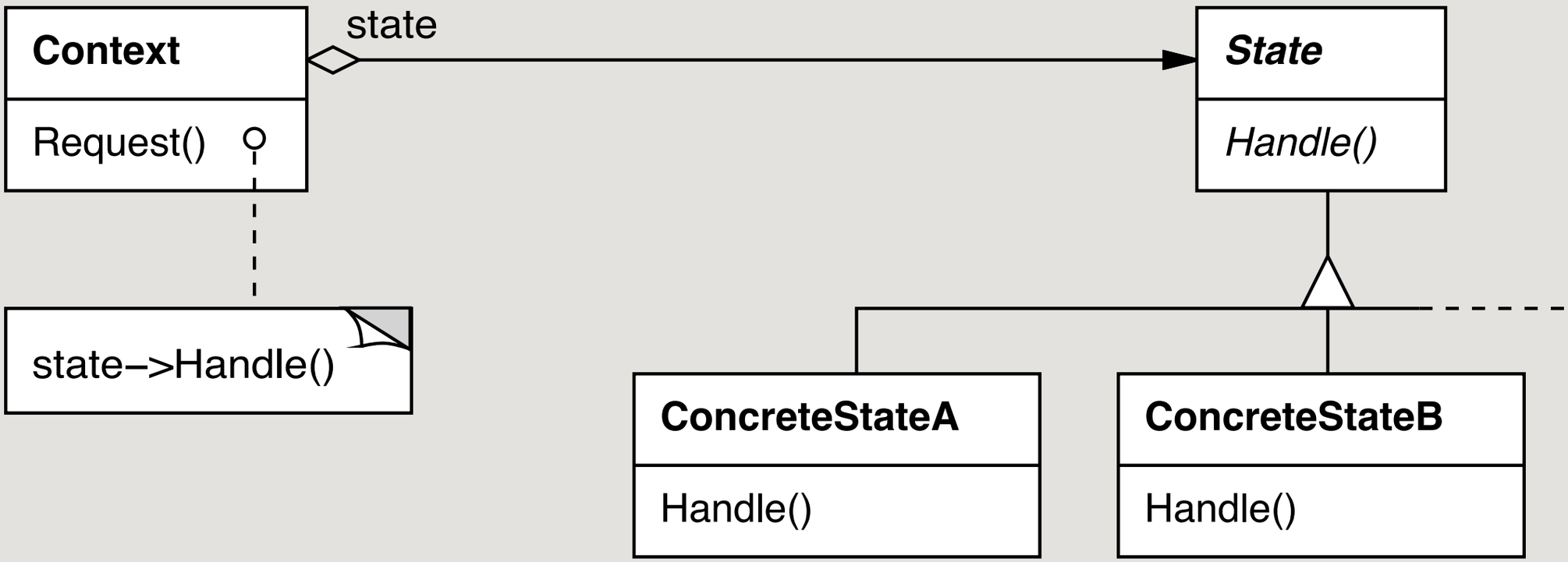

TCPConnection
Open()
Close()
Acknowledge()
state
TCPState
Open()
Close()
Acknowledge()
state->Open()
TCPEstablished
Open()
Close()
Acknowledge()
TCPListen
Open()
Close()
Acknowledge()
TCPClosed
Open()
Close()
Acknowledge()

Aplicabilidad

- **Úsese en los siguientes casos:**
  - El comportamiento de un objeto depende de su estado
    - Y éste puede cambiar en tiempo de ejecución
  - Las operaciones tienen sentencias condicionales anidadas que tratan con los estados
    - Siendo el estado normalmente una constante
    - Este patrón mueve cada rama de la lógica condicional a una clase aparte

      Lo que nos permite tratar al estado del objeto como un objeto de pleno derecho, que puede variar independientemente de otros objetos.

Estructura


Context
Request()
state
State
Handle()
state->Handle()
ConcreteStateA
Handle()
ConcreteStateB
Handle()

Participantes

- **Context (`TCPConnection`)**
  - Define la interfaz que interesa a los clientes
  - Mantiene una referencia a una subclase de estado concreto que representa el estado actual
- **State (`TCPState`)**
  - Define la interfaz para encapsular el comportamiento asociado con el estado del contexto
- **Subclases ConcreteSate (`TCPStablished`, `TCPListen`, `TCPClosed`)**
  - Cada subclase implementa las operaciones anteriores para ese estado concreto

Colaboraciones

- **El contexto delega las operaciones dependientes del estado al objeto que representa el estado actual**
- **El contexto podría pasarse a sí mismo como parámetro**
  - Para que el estado acceda al contexto si es necesario

Colaboraciones

- Una vez que el contexto es inicializado en un determinado estado, los clientes no necesitan tratar directamente con los estados
- O bien el contexto o bien los estados concretos deciden cuándo se pasa de uno a otro estado

Consecuencias

- **Localiza el comportamiento específico del estado y lo aísla en un objeto**
  - Se pueden añadir nuevos estados y transiciones fácilmente simplemente definiendo nuevas subclases de `State`
- **Hace explícitas las transiciones entre estados**

Implementación

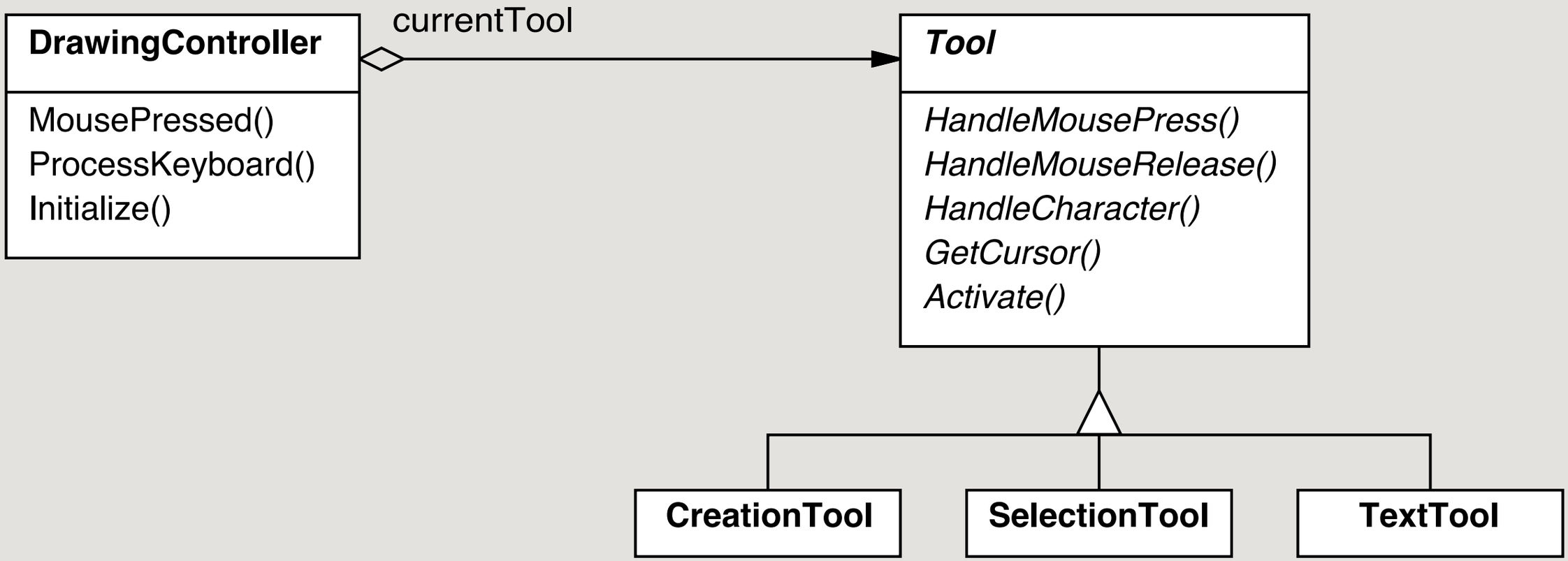
- **¿Quién define las transiciones entre estados?**
  - Si el criterio es siempre el mismo, puede ser el propio contexto
  - Normalmente es más flexible y apropiado que sean las propias subclases de los estados
    - Esto requiere añadir una operación al contexto para cambiar su estado de forma explícita
      - Ventaja: mucha flexibilidad, podemos modificar la lógica o añadir estados nuevos simplemente definiendo nuevas subclases
      - Inconveniente: más dependencia entre las subclases de los estados

Implementación

- **Uso de herencia dinámica**
  - Este patrón no sería necesario en lenguajes que permiten cambiar la clase de un objeto en tiempo de ejecución o que proveen mecanismos para delegar peticiones automáticamente en otros objetos

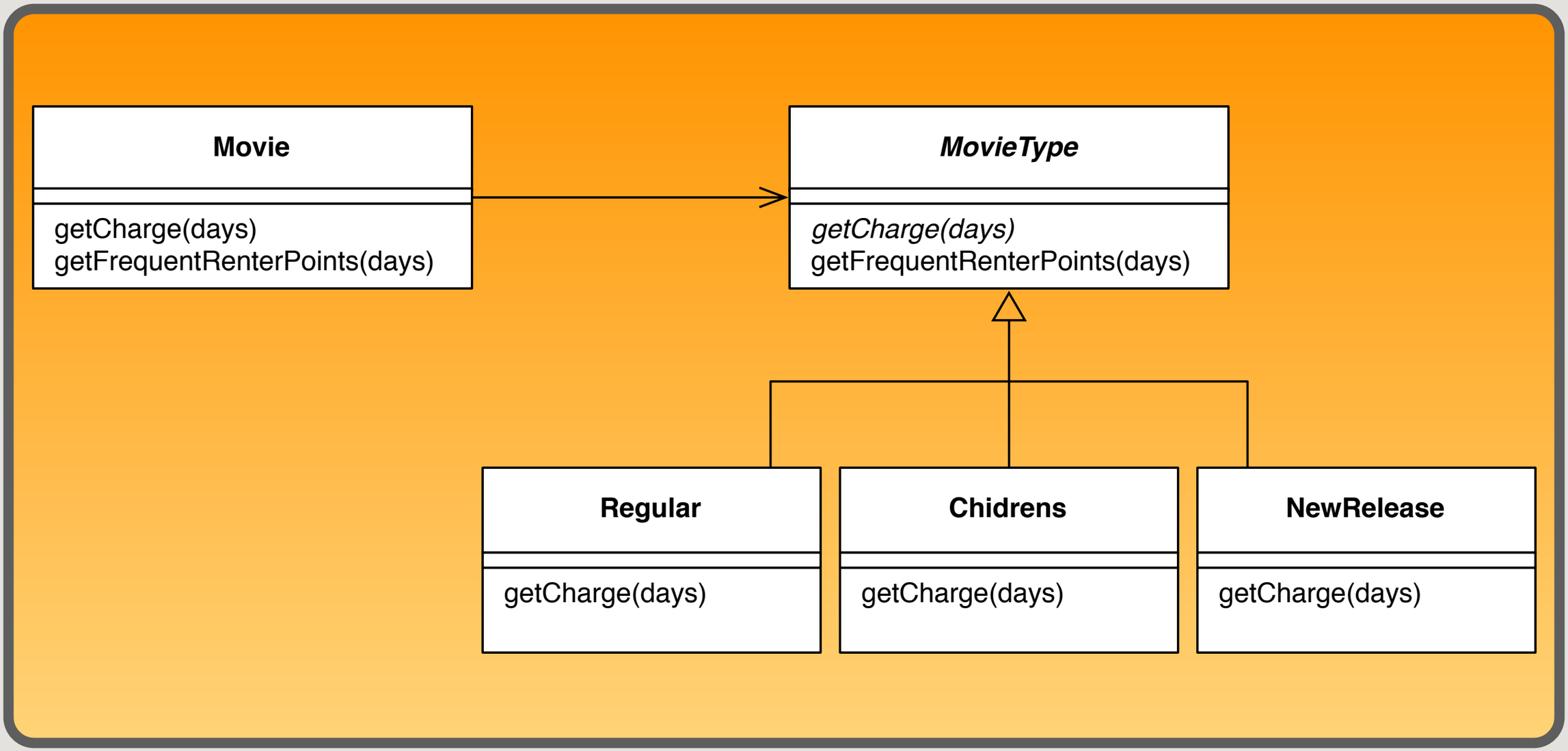
Usos conocidos

- Editor gráfico HotDraw



Otros ejemplos

Nuestro ejemplo del videoclub




Coca-Cola
Trademarks®
ICE COLD

Cuestiones

Cuestiones

- ¿En qué se diferencia del *Strategy*?