Estado (State)

- *Patrón de Comportamiento*
- *Propósito*

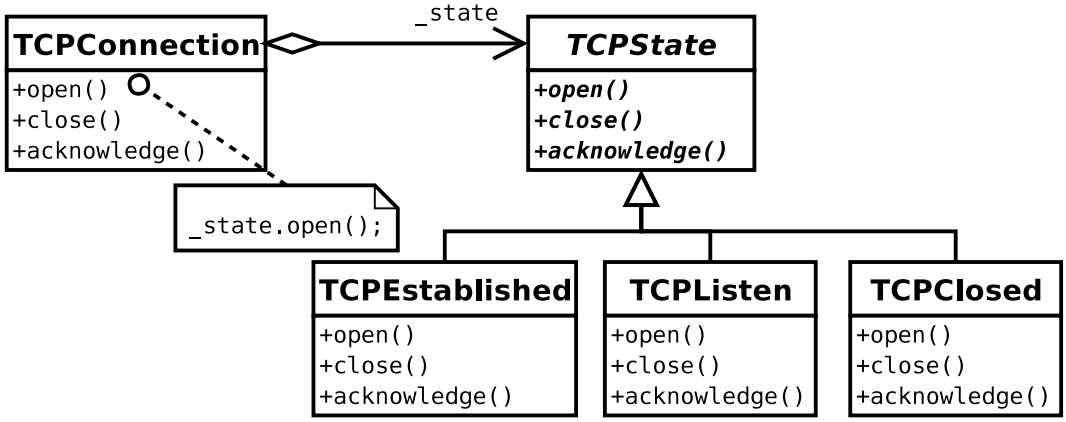
  Permite a un objeto modificar su comportamiento al cambiar su estado interno.

- *Motivación*

  - Clase para representar una conexión de red Un objeto puede estar en diferentes estados: Establecida, Escuchando, o Cerrada.
  - Al recibir una petición, el objeto se comporta de modo distinto en base a su estado.
  - Posibilidades:
    - Extender la clase conexión con subclases que representen la conexión en distintos estados.

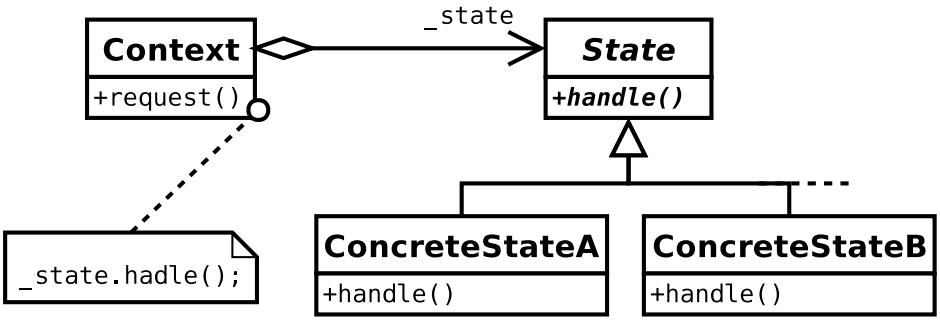
      Problema: Los objetos tienen que cambiar dinámicamente de clase
    - Única clase de conexión y condiciones (switch) dentro de los métodos en base al estado
  - Solución: Introducir una clase abstracta que representa a los estados de la conexión de red.
    - Superclase: define interfaz de los métodos dependientes del estado
    - Subclases: implementan el comportamiento específico del estado

- Motivación (cont.)
  - La clase conexión mantiene una referencia a un objeto estado en el que delega todo el comportamiento específico de dicho estado.



- *Aplicabilidad*
  - El comportamiento de un objeto depende de su estado, y dicho estado puede cambiar en tiempo de ejecución.

- *Estructura*



- *Participantes*
  - Contexto (Context)
    Define la interfaz para los clientes
    Mantiene una instancia de un estado concreto que define el estado actual del objeto contexto
  - Estado (State)
    Define una interfaz para encapsular el comportamiento asociado con un estado particular del contexto
  - Estados Concretos (ConcreteState$_i$)
    Cada subclase implementa el comportamiento asociado con un estado del contexto
- *Colaboraciones*
  - El contexto delega las partes de comportamiento específicos del estado al objeto ConcreteState$_i$
  - El contexto puede pasarse a sí mismo como argumento al objeto estado. De esta forma, el objeto estado puede acceder a información del contexto.
  - La configuración de un contexto la puede realizar un cliente con objetos estado
  - Los clientes utilizan el contexto como interfaz, sin necesidad de manejar objetos estado directamente
  - Tanto el contexto como los estados concretos pueden decidir cambiar el estado actual

- *Consecuencias*
  - Localiza y separa el comportamiento específico de cada estado
    - Facilita añadir nuevos estados y transiciones
    - ... pero es menos compacto (clases ↑)
  - Hace explícitas las transiciones entre estados
  - Bajo ciertas circunstancias, es posible la compartición de objetos estado
- *Implementación*
  - Responsabilidad de efectuar transición entre estados
  - Creación y destrucción de objetos estado
    - Crear y destruir cada vez que es necesario
    - Crear al principio y no destruir
  - Herencia dinámica