STATE

Carlos Anibal Valdez Cordero

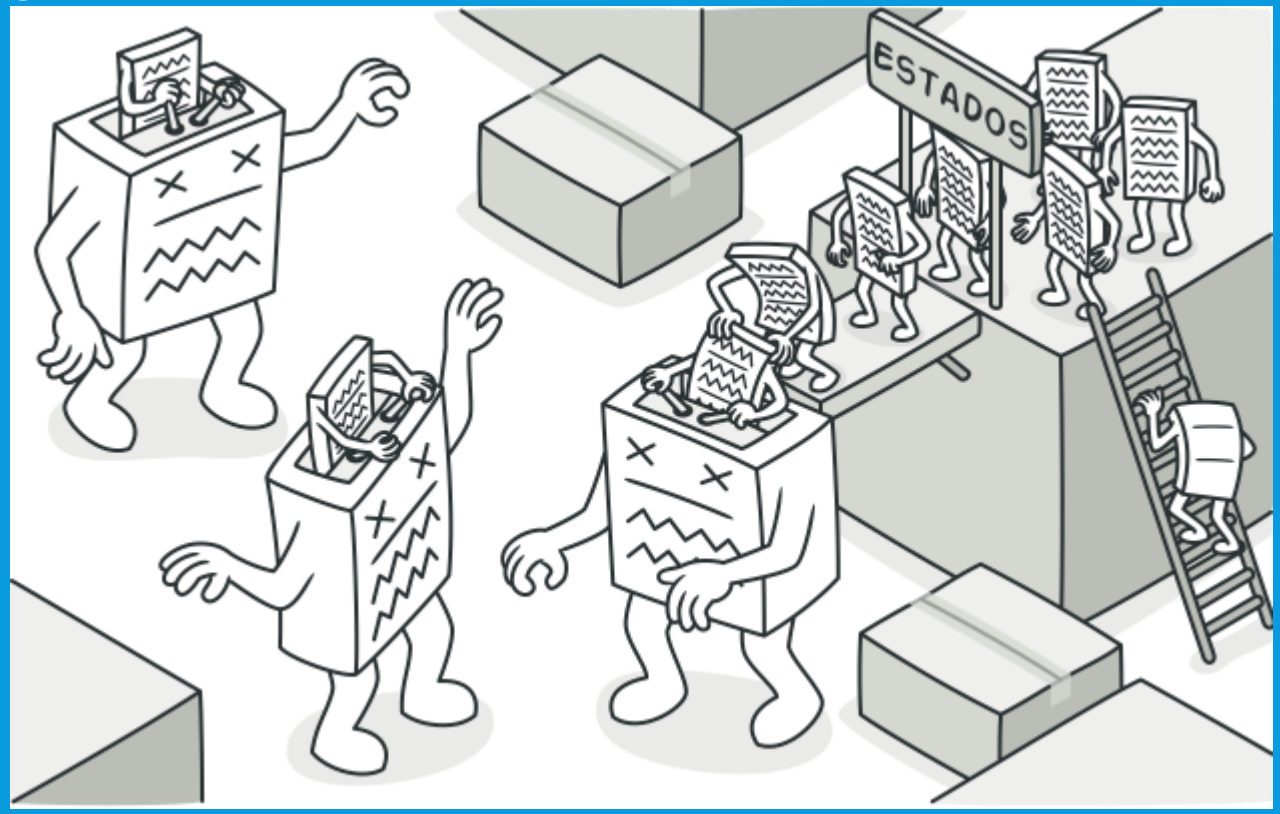
338835

6HW1

Lenguajes de programación IV

PROPÓSITO

State es un patrón de diseño de comportamiento que permite a un objeto alterar su comportamiento cuando su estado interno cambia. Parece como si el objeto cambiara su clase.



PROBLEMA

El patrón State está estrechamente relacionado con el concepto de la *Máquina de estados finitos*

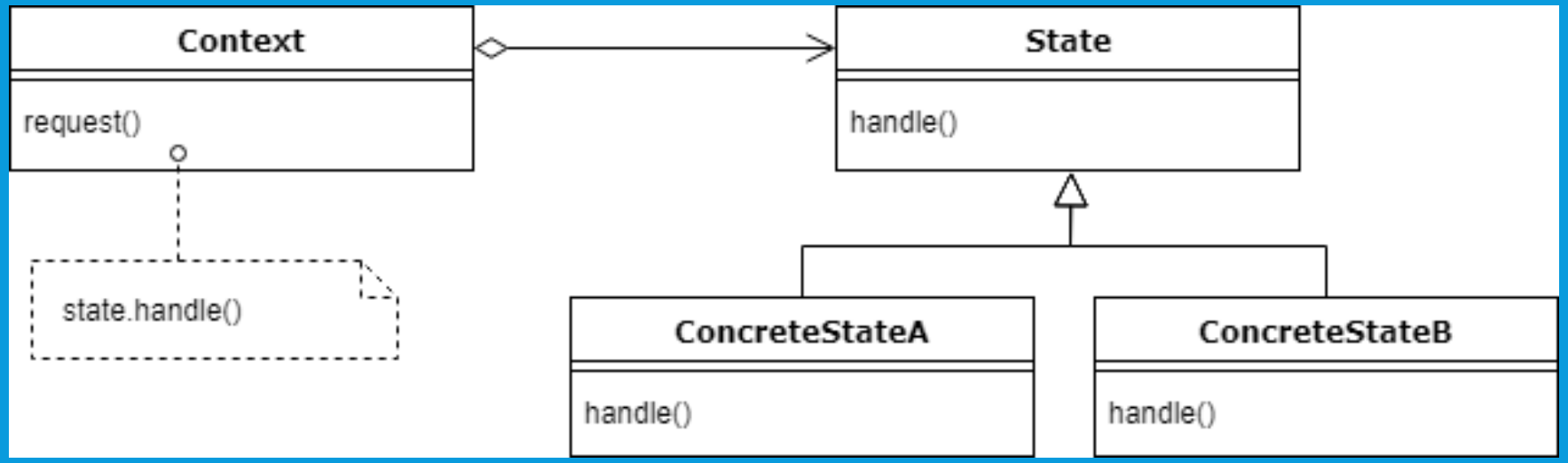
La idea principal es que, en cualquier momento dado, un programa puede encontrarse en un número *finito* de *estados*. Dentro de cada estado único, el programa se comporta de forma diferente y puede cambiar de un estado a otro instantáneamente. Sin embargo, dependiendo de un estado actual, el programa puede cambiar o no a otros estados. Estas normas de cambio llamadas *transiciones* también son finitas y predeterminadas.

COLABORACIONES

El patrón State puede utilizar el patrón Singleton cuando requiera controlar que exista una sola instancia de cada estado. Lo puede utilizar cuando se comparten los objetos como Flyweight existiendo una sola instancia de cada estado y ésta es compartida con más de un objeto.

Flyweight es un patrón de diseño estructural que te permite mantener más objetos dentro de la cantidad disponible de RAM compartiendo las partes comunes del estado entre varios objetos en lugar de mantener toda la información en cada objeto.

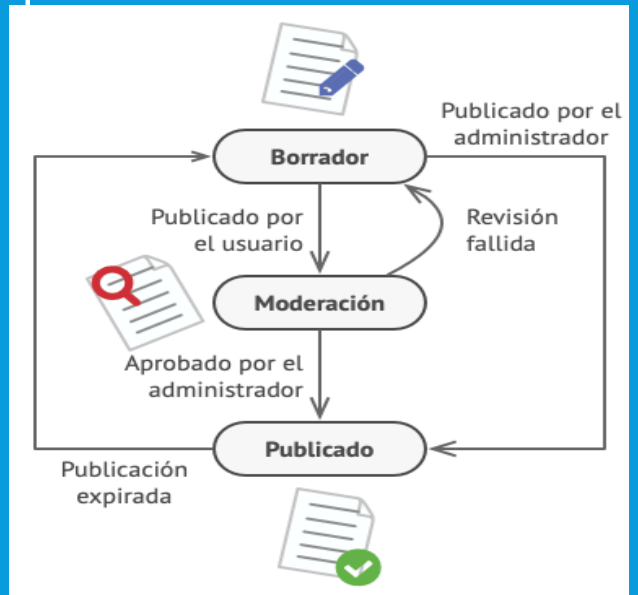
UML


Context
request()
state.handle()
State
handle()
ConcreteStateA
handle()
ConcreteStateB
handle()

EJEMPLO

También puedes aplicar esta solución a los objetos. Imagina que tienes una clase Documento. Un documento puede encontrarse en uno de estos tres estados: Borrador, Moderación y Publicado. El método publicar del documento funciona de forma ligeramente distinta en cada estado:

- En Borrador, mueve el documento a moderación.
- En Moderación, hace público el documento, pero sólo si el usuario actual es un administrador.
- En Publicado, no hace nada en absoluto.



SOLUCIÓN

El patrón State sugiere que crees nuevas clases para todos los estados posibles de un objeto y extraigas todos los comportamientos específicos del estado para colocarlos dentro de esas clases.

En lugar de implementar todos los comportamientos por su cuenta, el objeto original, llamado *contexto*, almacena una referencia a uno de los objetos de estado que representa su estado actual y delega todo el trabajo relacionado con el estado a ese objeto.

ANALOGÍA DEL MUNDO REAL

Los botones e interruptores de tu smartphone se comportan de forma diferente dependiendo del estado actual del dispositivo:

- Cuando el teléfono está desbloqueado, al pulsar botones se ejecutan varias funciones.
- Cuando el teléfono está bloqueado, pulsar un botón desbloquea la pantalla.
- Cuando la batería del teléfono está baja, pulsar un botón muestra la pantalla de carga.