- Esperamos hasta las 19:05 parara que ingresen todos sus compañeros

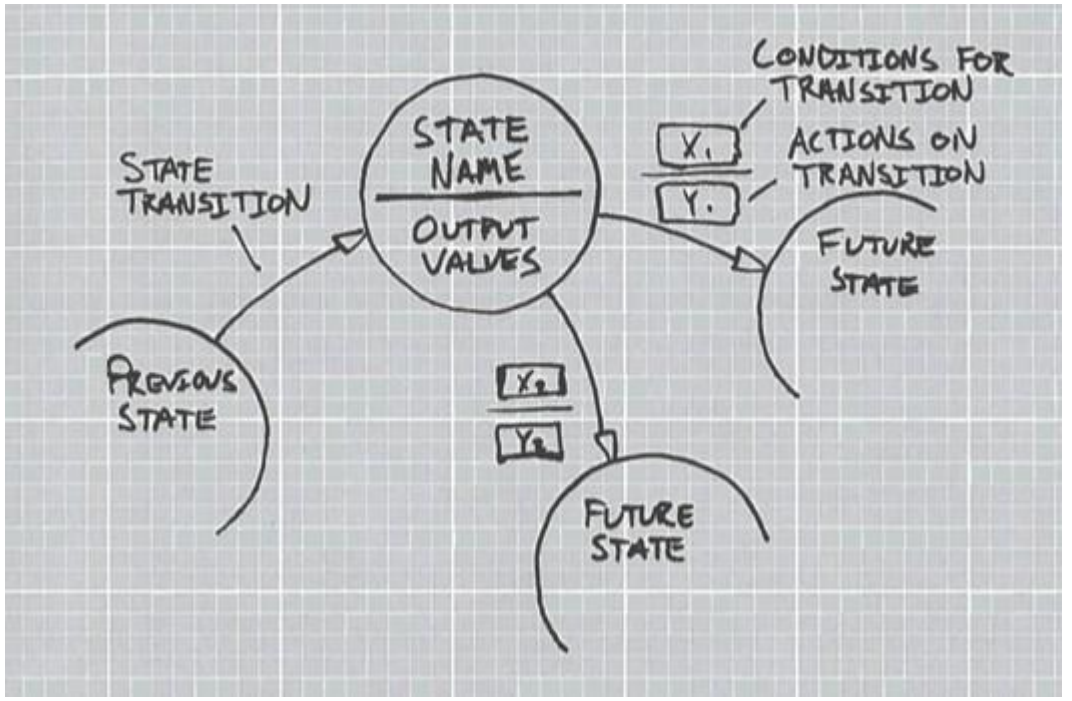
Patrones de Diseño de Software

PATRONES COMPORTAMIENTO - STATE

Patron de diseño State

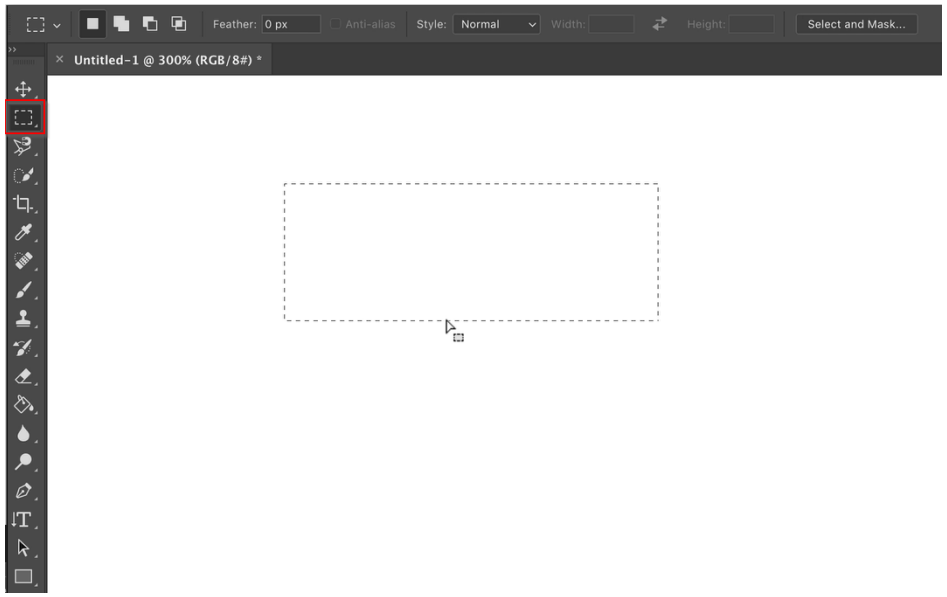
PROPOSITO:

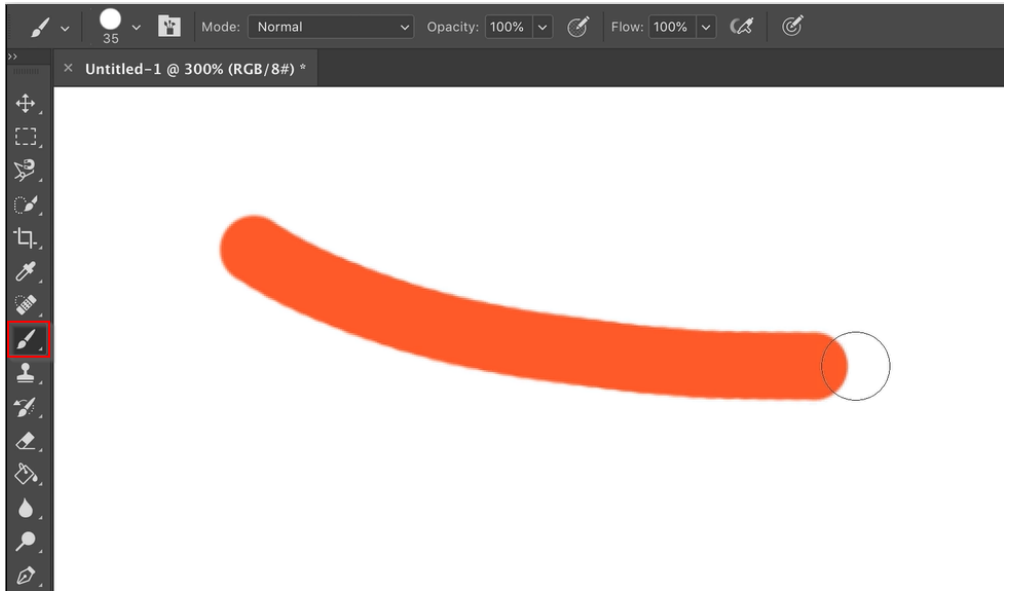
State es un patrón de diseño de comportamiento que permite a un objeto alterar su comportamiento cuando su estado interno cambia. Parece como si el objeto cambiara su clase.

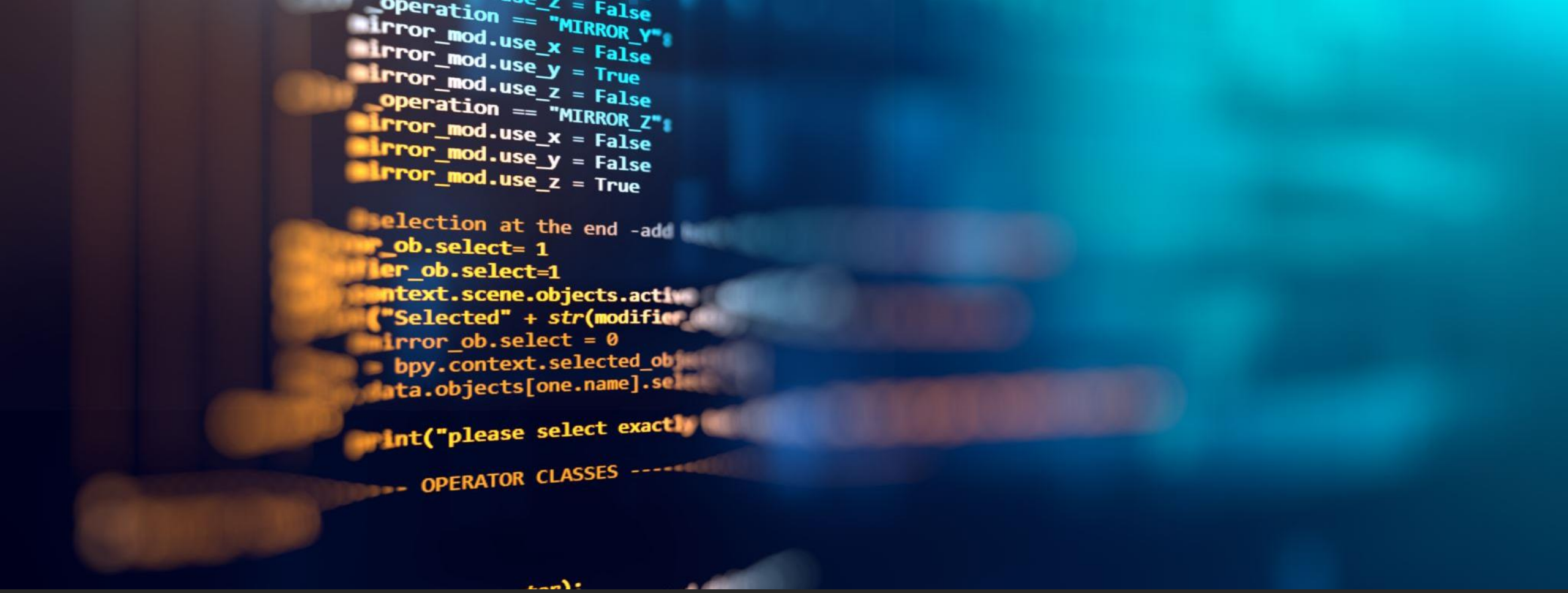


Patrones de diseño – State

Problema: Ejemplo estados cursor en aplicación Grafica





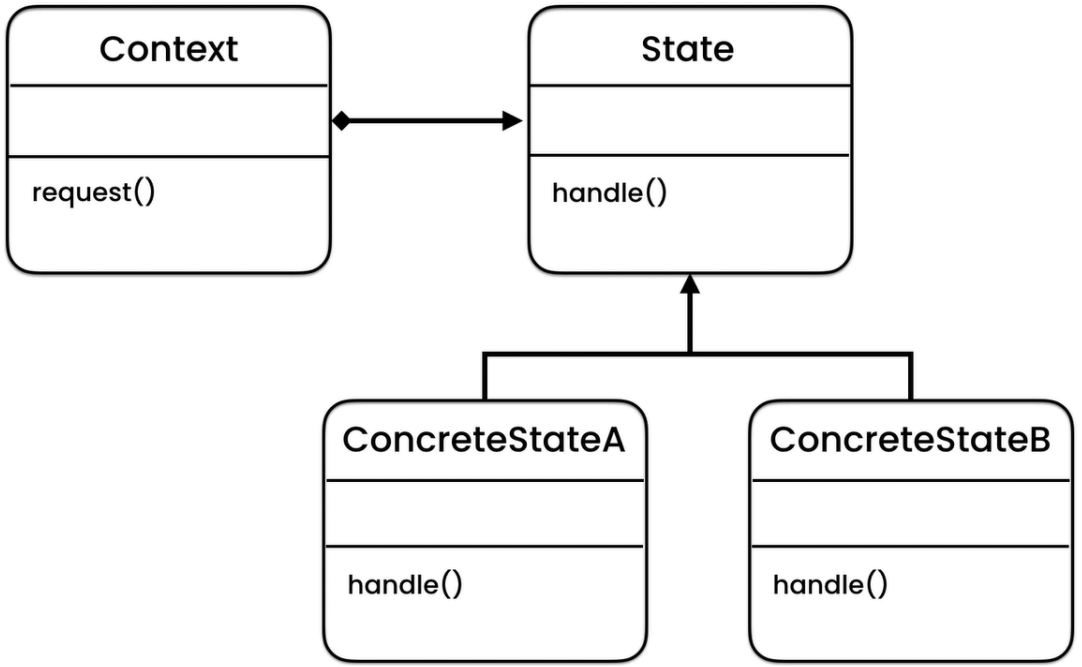


Demo

Revisemos en Código la implementación del problema

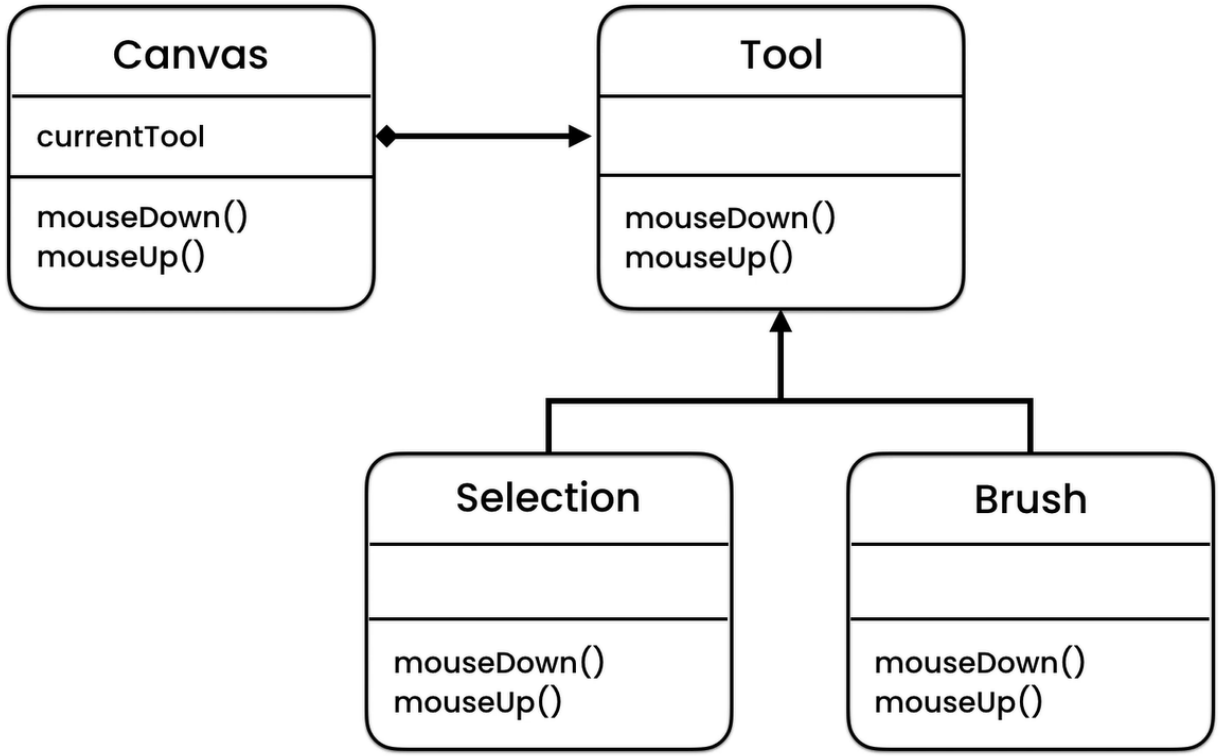
Patrones de diseño – State

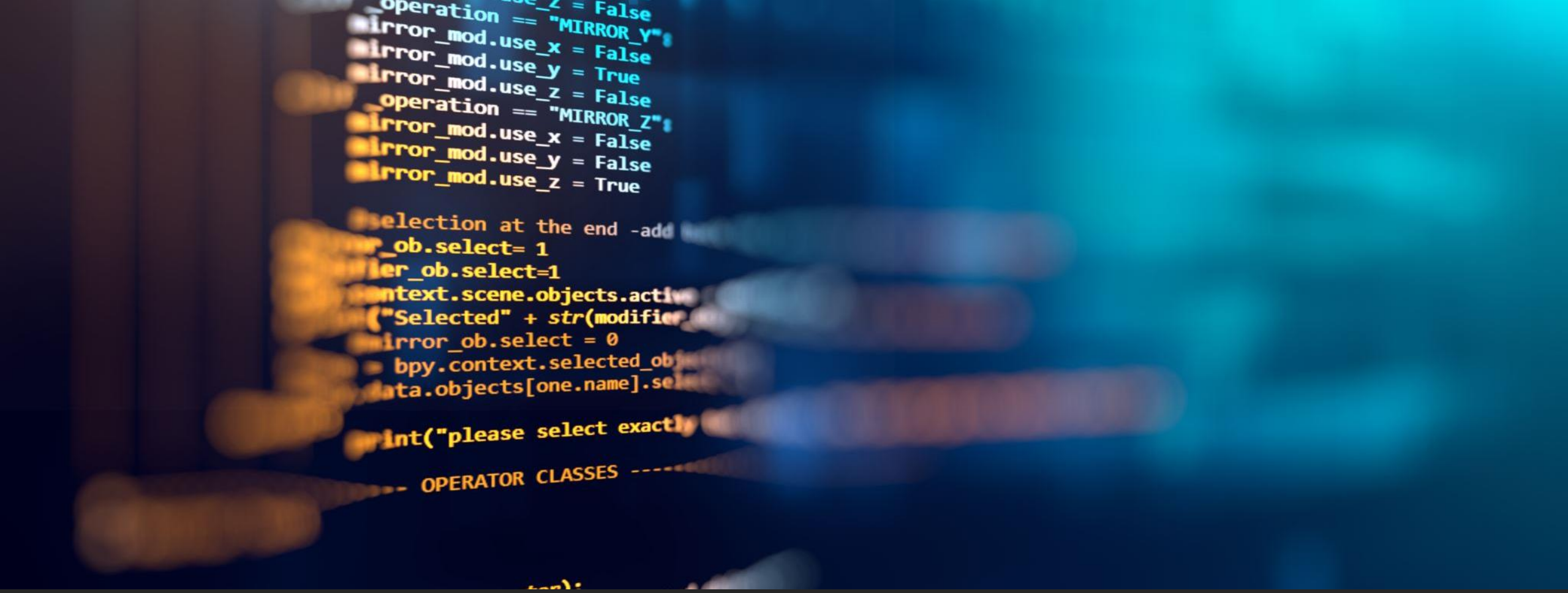
Solución:



Patrones de diseño – State

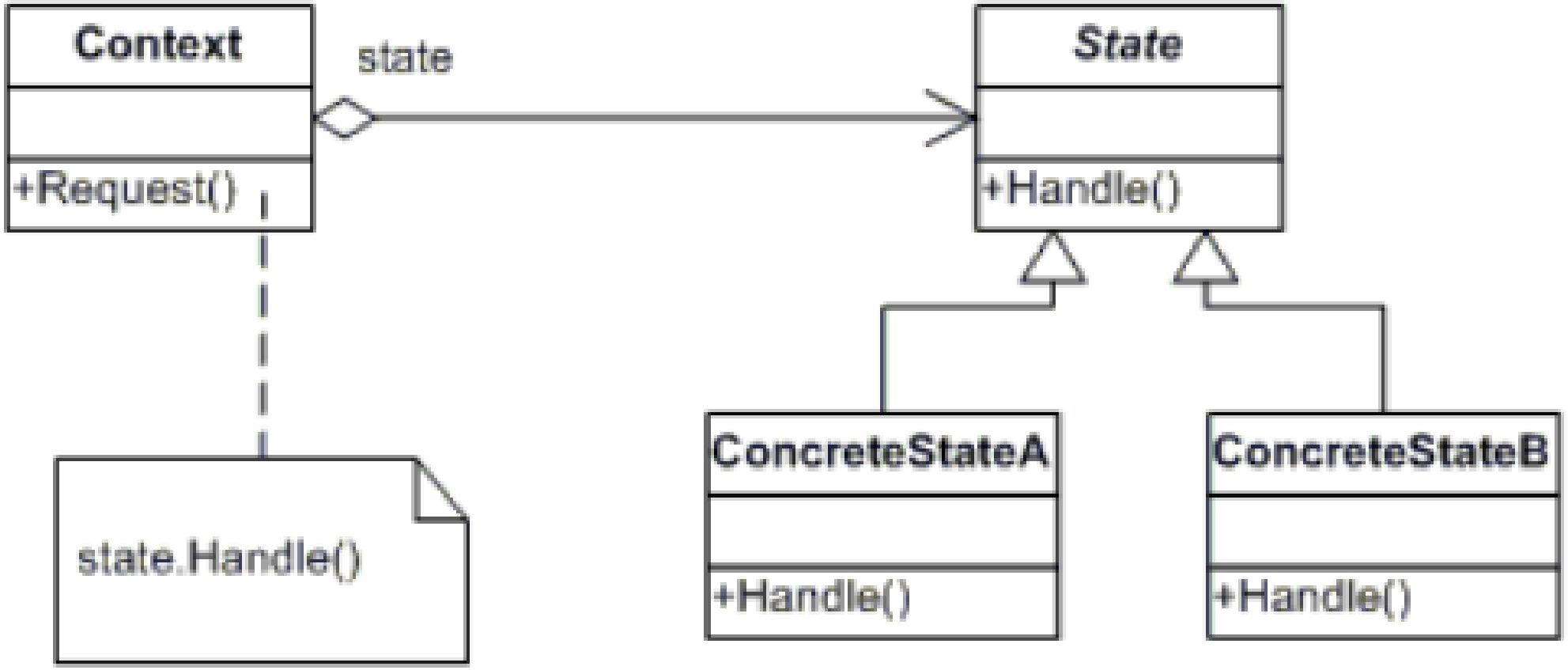
Implementación





Demo

Revisemos en Código la implementación de la solución



Patron de diseño - State

Estructura

Patrones de diseño – State

Ventajas:

- ✓ Principio de responsabilidad única. Organiza el código relacionado con estados particulares en clases separadas.
- ✓ Principio de abierto/cerrado. Introduce nuevos estados sin cambiar clases de estado existentes o la clase contexto.
- ✓ Simplifica el código del contexto eliminando voluminosos condicionales de máquina de estados.

Desventajas:

- × Aplicar el patrón puede resultar excesivo si una máquina de estados sólo tiene unos pocos estados o raramente cambia.

Patrón de diseño State

Cuándo utilizar este patrón?

Utiliza el patrón State cuando tengas un objeto que se comporta de forma diferente dependiendo de su estado actual, el número de estados sea enorme y el código específico del estado cambie con frecuencia.

Utiliza el patrón cuando tengas una clase contaminada con enormes condicionales que alteran el modo en que se comporta la clase de acuerdo con los valores actuales de los campos de la clase.

Utiliza el patrón State cuando tengas mucho código duplicado por estados similares y transiciones de una máquina de estados basada en condiciones.

Patrones de diseño – State

Práctica

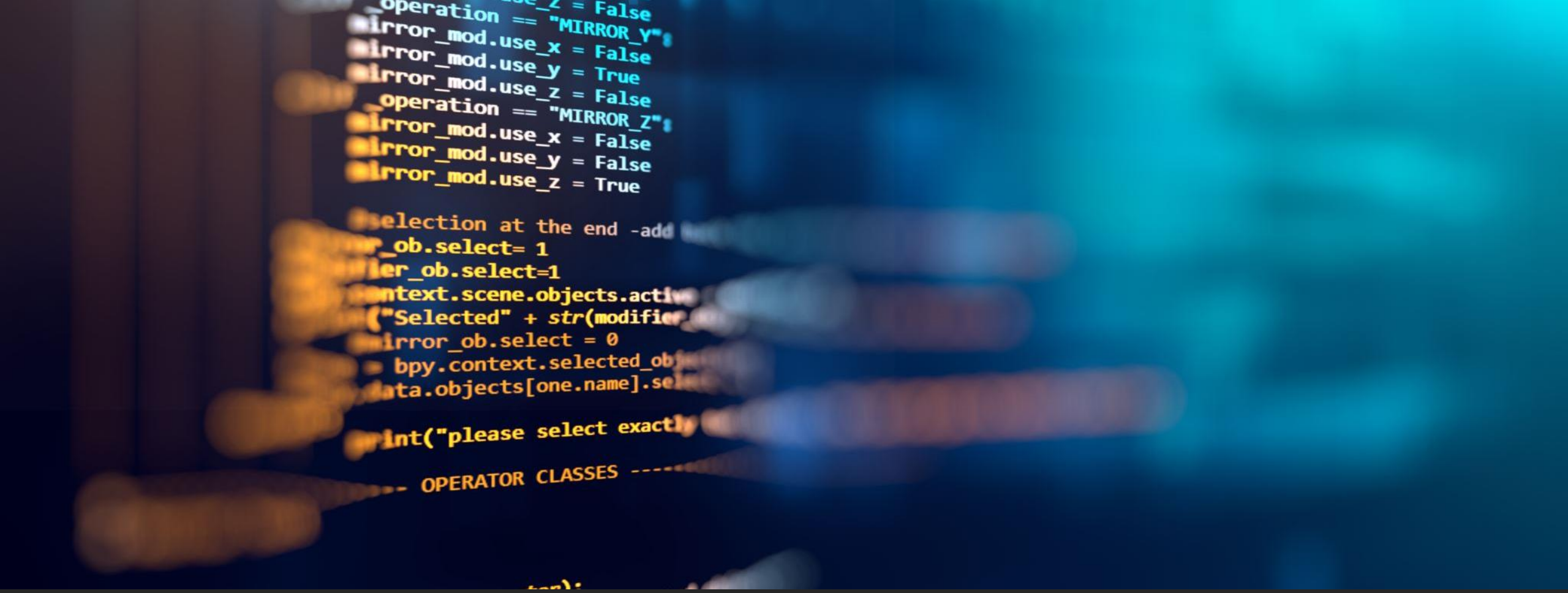
Pensemos en los posibles estados y acciones de un cajero

Acciones:

- InsertaTarjeta
- DevuelveTarjeta
- IntroducePin
- SolicitaDinero

Estados:

- TarjetaIntroducida
- SinTarjeta
- PinIntroducido
- SinDinero



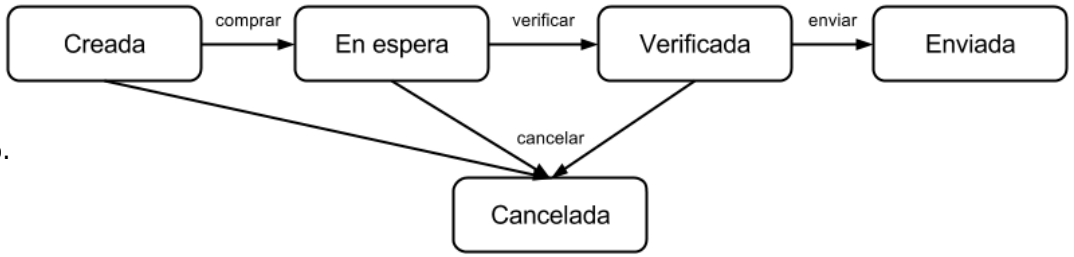
Demo

Solución de la tarea

Patrones de diseño – State

Tarea:

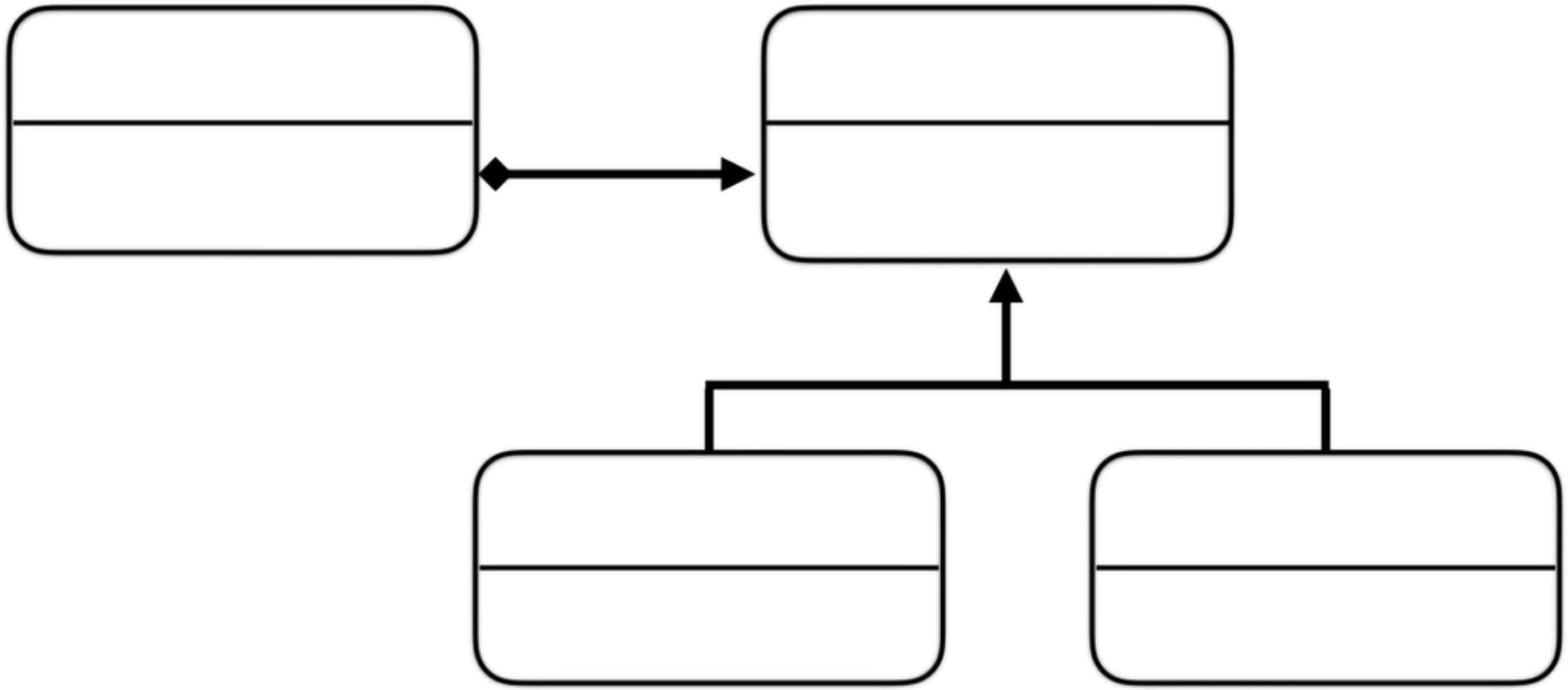
- supongamos que tenemos una entidad Compra que a lo largo de su vida en la aplicación pasa por diferentes estados:
  - **creada**: la compra se acaba de crear.
  - **en espera**: se ha hecho una compra y se está esperando que el pago sea correcto.
  - **verificada**: el pago es correcto y se está esperando a enviar el producto.
  - **cancelada**: la compra se ha cancelado porque el usuario no quiere ya el producto, no hay existencias u otro motivo.
  - **enviada**: el pedido ha sido enviado.
- Y tiene diferentes transiciones como:
  - **comprar**: la compra pasa de creada a en espera de verificarla.
  - **verificar**: la compra pasa de en espera a verificada y esperando a enviarse.
  - **cancelar**: la compra se puede cancelar excepto una vez que ya se ha enviado.
  - **enviar**: la compra se envía al usuario y ya no puede cancelarse.




Gracias

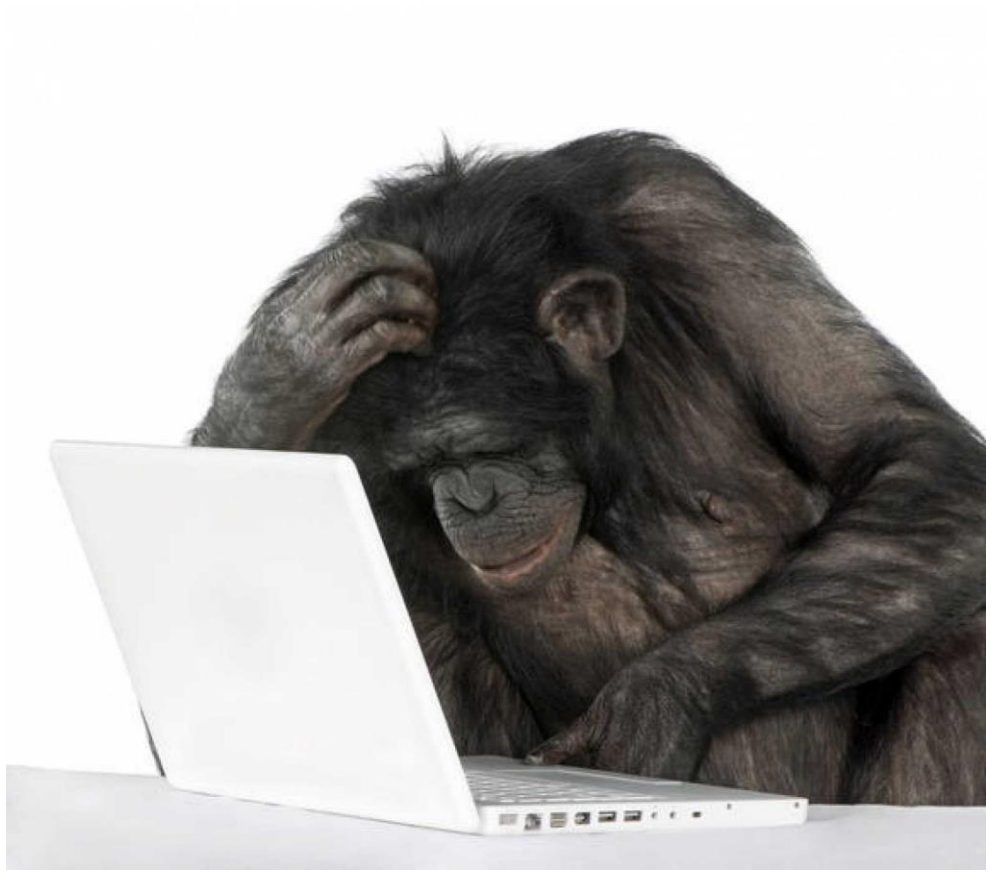
Patrones de diseño

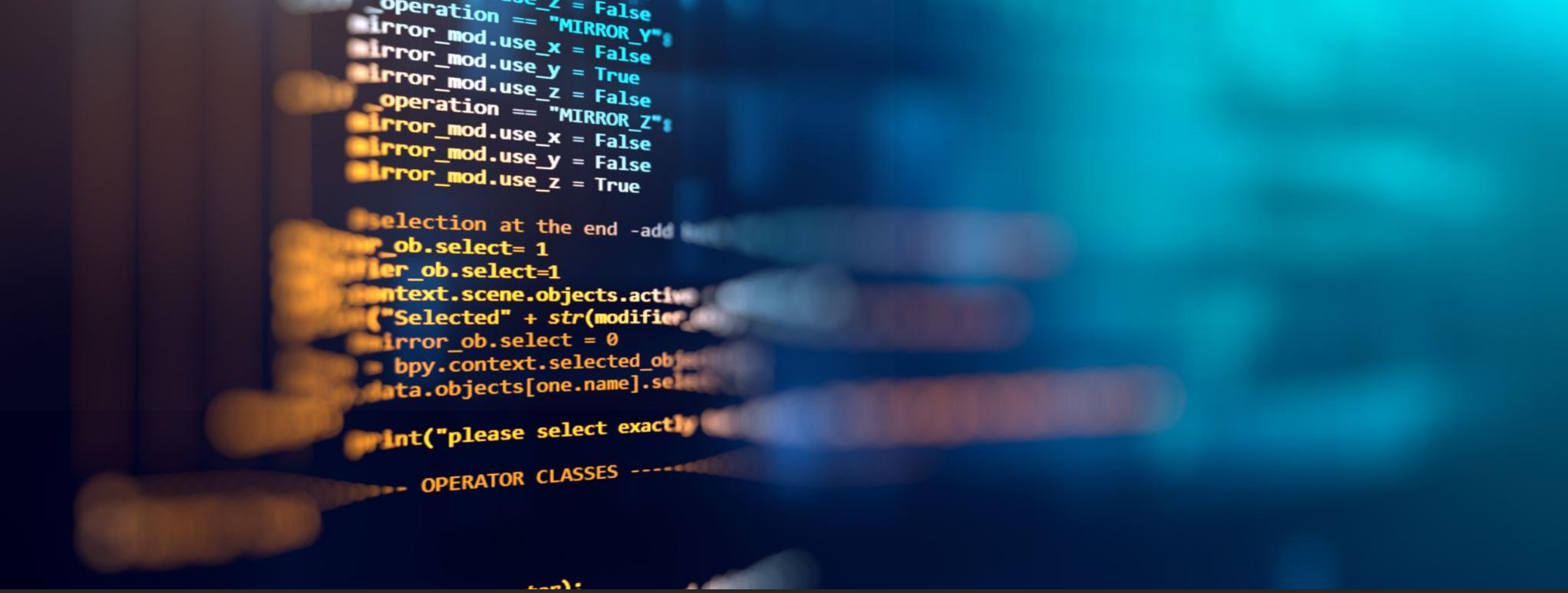
Abuso del uso de los patrones de diseño



Yo soy mejor que tu!

Lo importante es escribir mucho código



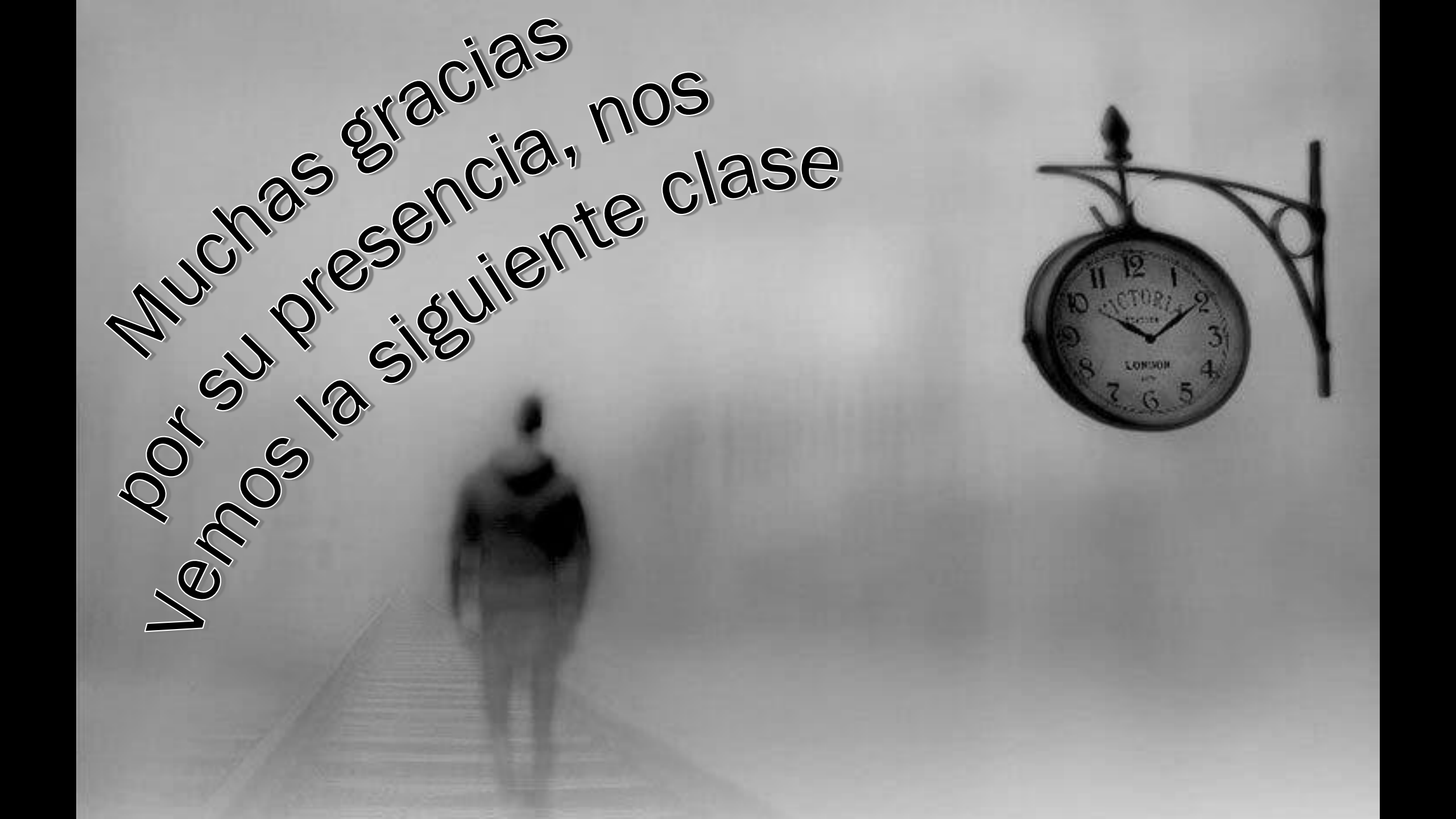


Demo

Ejemplo de abuso en el uso de patrones


30
1
2
3
4
5
6
7
8
9
10
25
30
5
25
26
27
28
29
30
24
25


Gracias


Muchas gracias
por su presencia, nos
Vemos la siguiente clase