Capgemini

Coding Dojo 6

Kata Lunar Rover

TDD – Design Patterns

devonfw

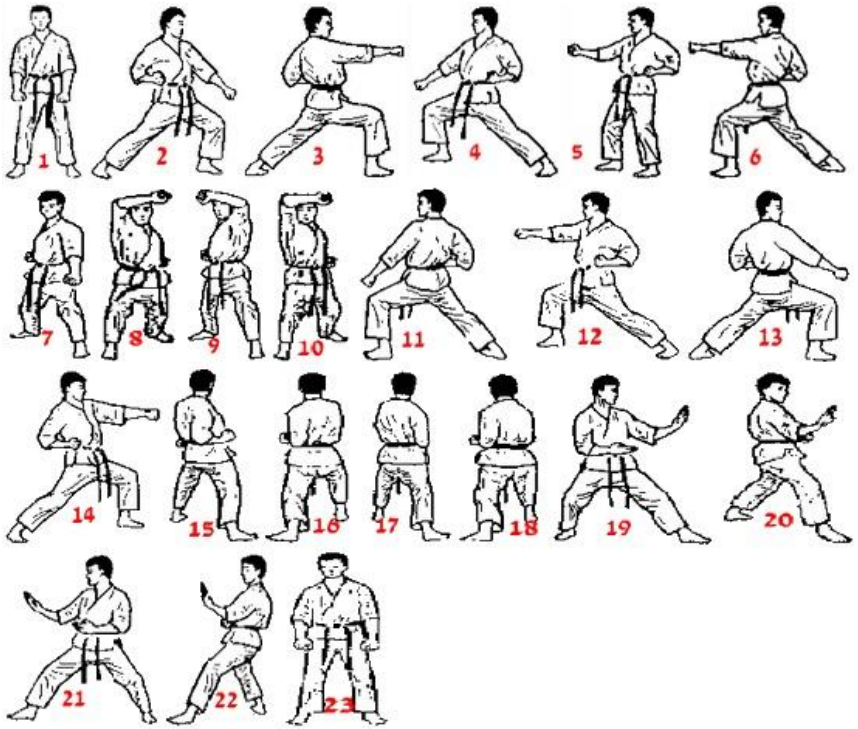
Antes de empezar (Introducción)

Lugar donde se reúne la gente para practicar y entrenar artes marciales.

CODING DOJO

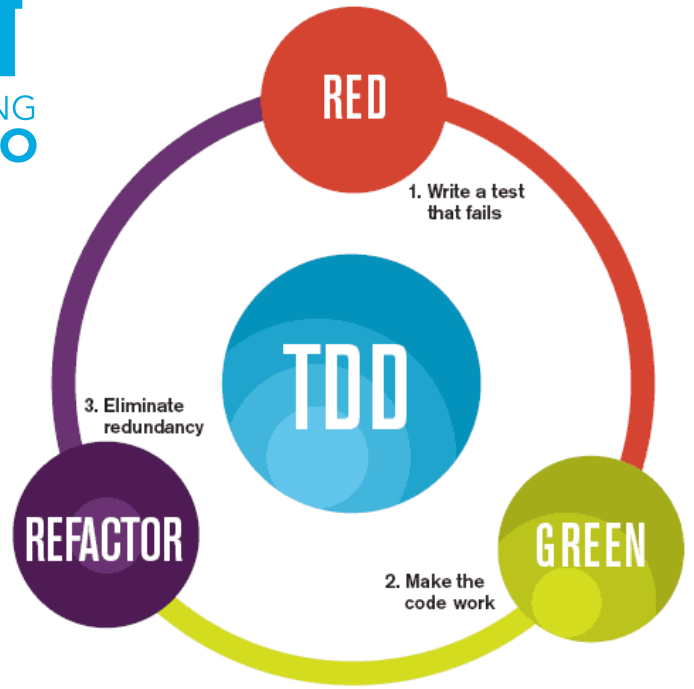
Lugar donde nos reunimos para practicar y entrenar buenas formas de programación.

Antes de empezar (Introducción)



Ejecutan Katas para aprender los movimientos y las técnicas.

CODING DOJO



Usamos TDD

Desarrollamos por consenso

Antes de empezar (Refactor)

"Cambios en el código para hacerlo más fácil de entender y más barato de modificar, **sin alterar su comportamiento observable**"

- Martin Fowler -

Antes de empezar (Refactor)

Basic smells

COMMENTS

MAGIC NUMBER

LONG METHOD

DUPLICATE METHOD

LARGE CLASS

LONG PARAMETER LIST

Design smells

SWITCHS

PRIMITIVE OBSESSION

MESSAGE CHAINS

SPECULATIVE GENERALIZATION

DATA CLUMPS

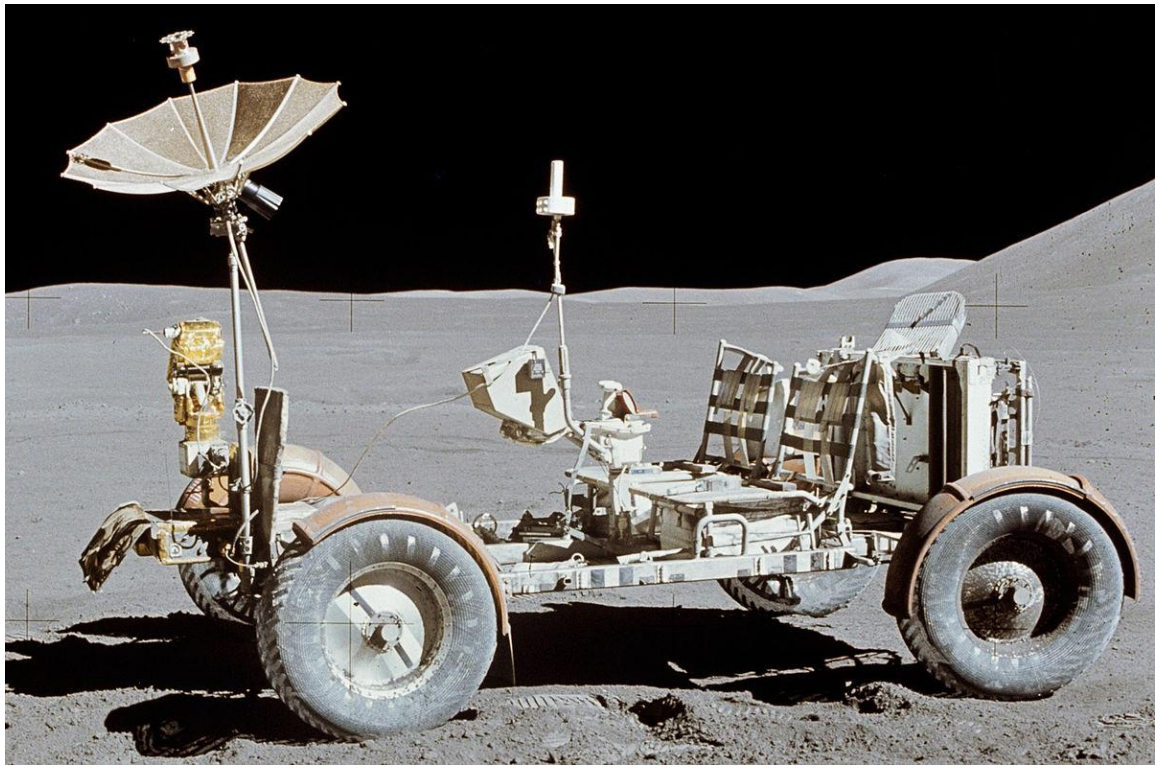
FEATURE ENVY

https://refactoring.guru/refactoring/smells

Práctica

Kata Lunar Rover

Lunar Rover (Enunciado)



En la próxima misión no tripulada de la ESA, se pretende llevar un vehículo Rover de control remoto a la Luna.

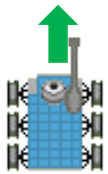
Un Lunar Rover es un vehículo de exploración espacial diseñado para moverse a través de la superficie Lunar.

La Agencia ha contratado al mejor equipo de ingenieros para que diseñen e implementen una API de comunicaciones con el Rover, de tal forma que se pueda controlar su movimiento enviando unos sencillos comandos.

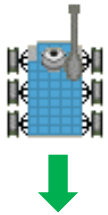
Lunar Rover (Enunciado)

Las reglas son sencillas:

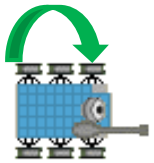
- Trataremos la superficie lunar como una rejilla de 100x100 casillas
- Las orientaciones permitidas del Rover serán (‘N’, ‘E’, ‘S’, ‘W’)
- La posición inicial del Rover será la casilla (0,0) y con orientación ‘N’
- La luna es redonda, cuando lleguemos a la casilla 100, en realidad estaremos en la casilla 0
- El Rover recibe los comandos enviados de uno en uno y los ejecuta en el momento
- Los comandos permitidos para el Rover son “char” para ahorrar ancho de banda y son los siguientes:



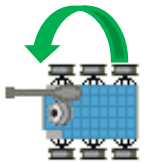
‘F’ → Avance



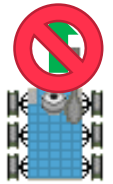
‘B’ → Retroceso



‘R’ → Giro derecha



‘L’ → Giro Izquierda



‘U’ → Deshacer

Basada en la kata Mars Rover: https://technologyconversations.com/2014/10/17/java-tutorial-through-katas-mars-rover/

Lunar Rover (Consejos)

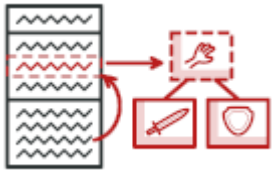
- Se debe intentar realizar el test de uno en uno.
  - Desde lo más sencillo a lo más complejo. Test recomendados:
    - Probar posición inicial para definir la estructura general de la API → (0,0) y 'N'
    - Probar comandos sencillos y comprobar su posición → 'F', 'B', 'R', 'L'
    - Probar los límites del grid
    - Probar combinaciones y el comando 'U'
- Se debe intentar implementar el código mínimo necesario para cada test.
  - Aunque se conozca el problema completo, hay que evitar el "*Speculative generalization*"
- En cada implementación de un test, se debe refactorizar el código (el test también).
  - ¿*me siento cómodo con este código*?
  - ¿*cambiaría la implementación a objetos*?
  - ¿*cambiaría la implementación con herencia*?
  - ¿*es la mejor estructura de datos que se podría utilizar*?

Refactoring

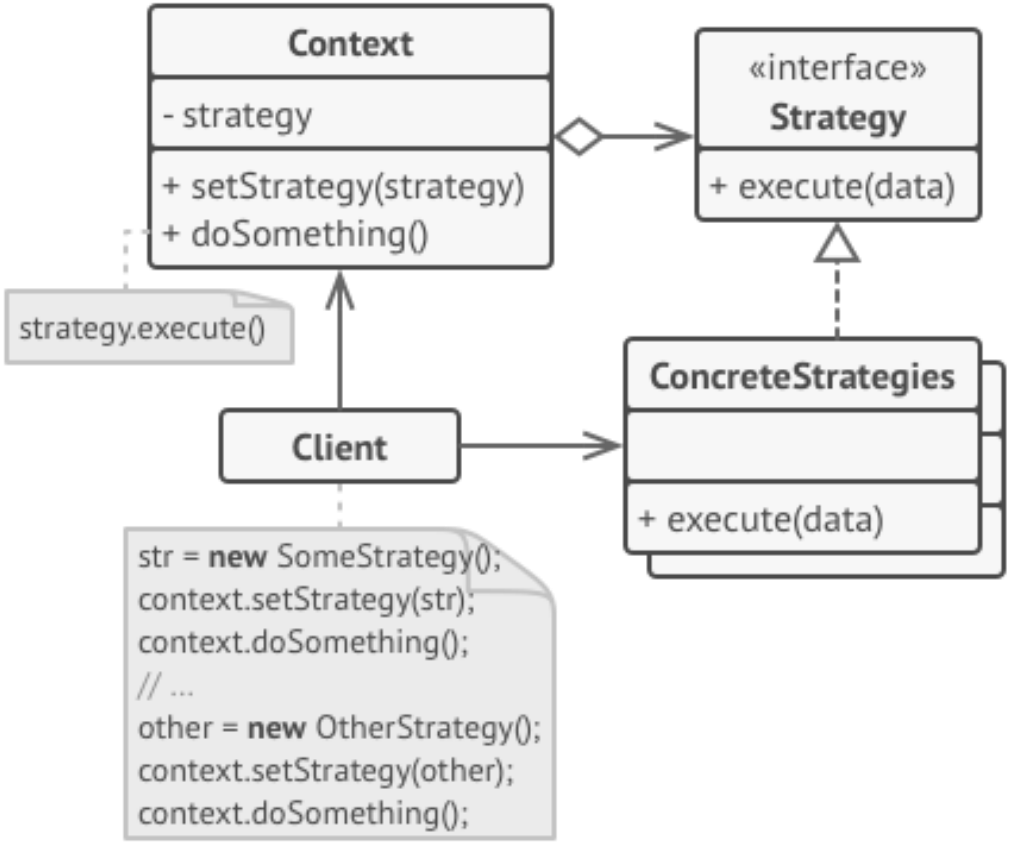
Design Patterns

Advanced Refactoring (Design Patterns)

Strategy



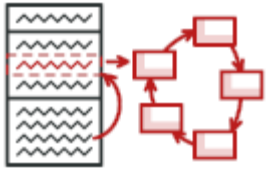
El patrón ‘Strategy’ tiene sentido cuando nos encontramos un escenario en el que para conseguir un objetivo, existen diferentes estrategias para lograrlo.



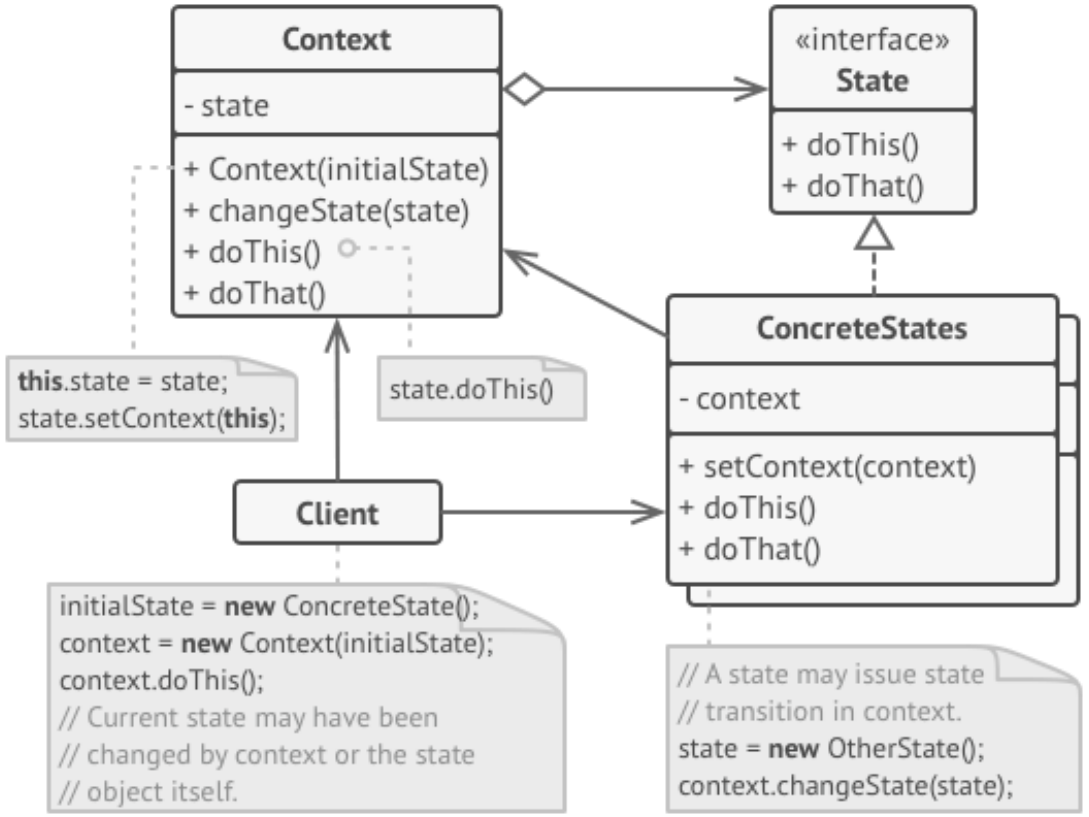
Si aparece una nueva estrategia, tan solo hay que añadir una nueva clase que herede de la genérica e implementar esa estrategia concreta.

Advanced Refactoring (Design Patterns)

State



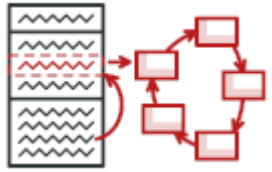
El patrón ‘State’ tiene sentido cuando nos encontramos un escenario en el comportamiento es diferente dependiendo del estado en el que se encuentra.

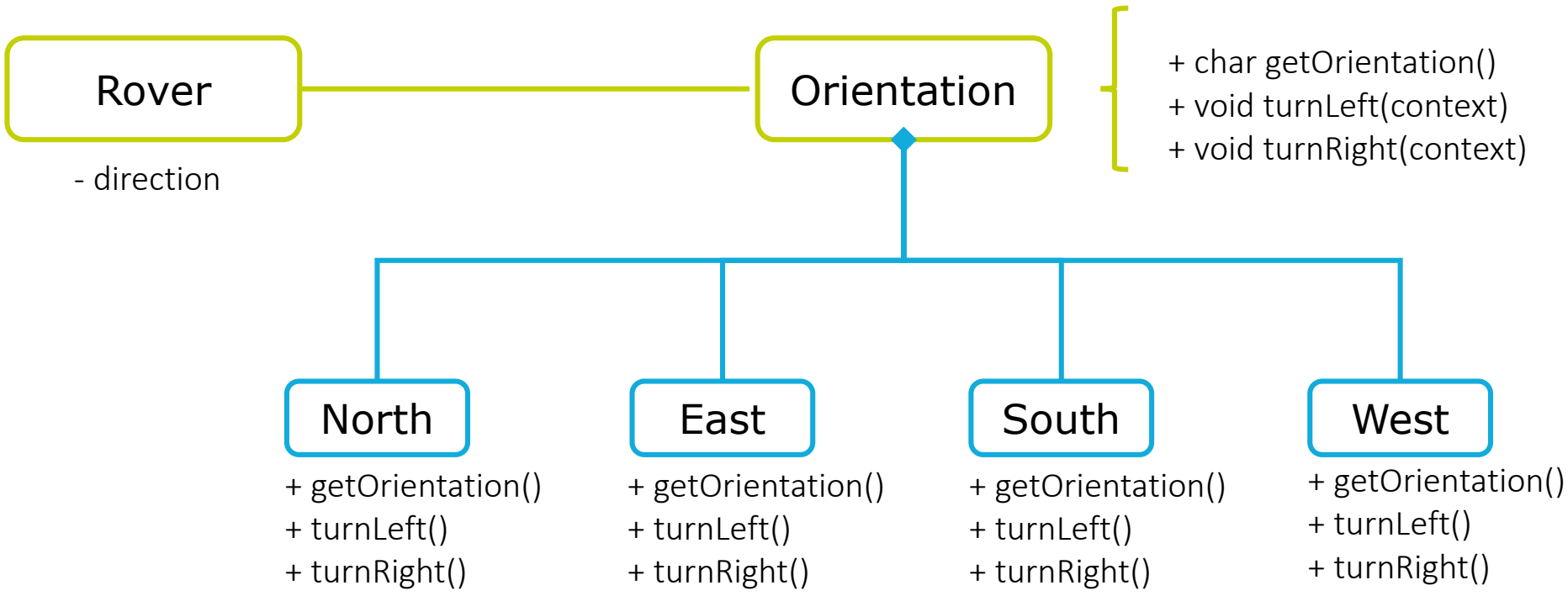


Si aparece un nuevo estado, tan solo hay que añadir una nueva clase que implemente el interface, implementar las acciones de ese nuevo estado y las posibles transiciones.

Advanced Refactoring (Design Patterns)

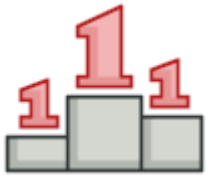
State (Implementación)



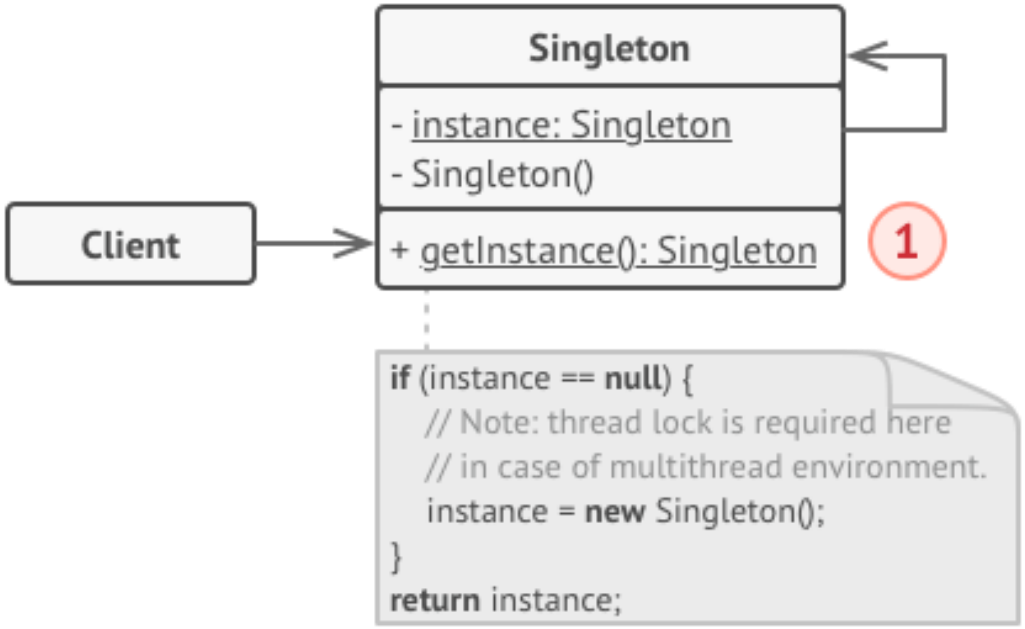


Advanced Refactoring (Design Patterns)

Singleton



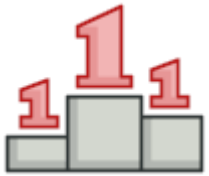
El patrón ‘Singleton’ permite restringir la creación de una clase a un solo objeto. Permite rebajar la creación de objetos y reutilizar los ya existentes, además permite el acceso global a una instancia desde cualquier punto del programa.

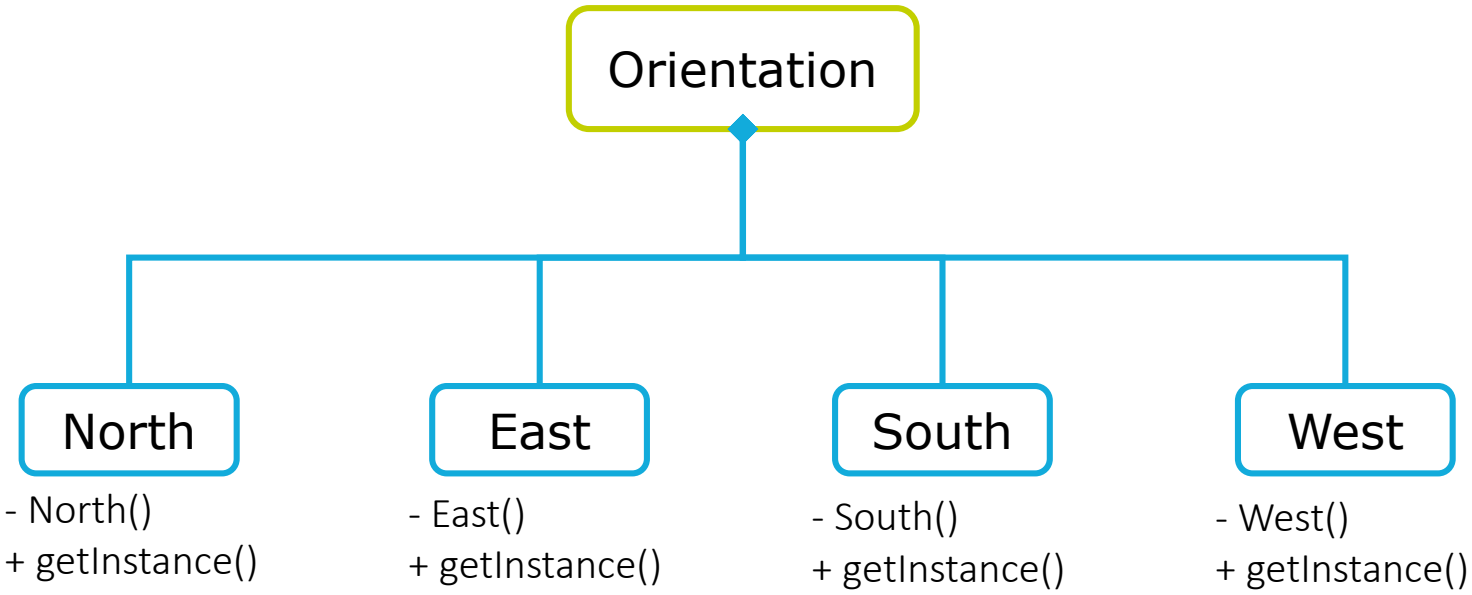


Hay que tener cuidado con este patrón ya que el objeto será único y compartido por todas las clases que hagan uso de él.

Advanced Refactoring (Design Patterns)

Singleton (Implementación)

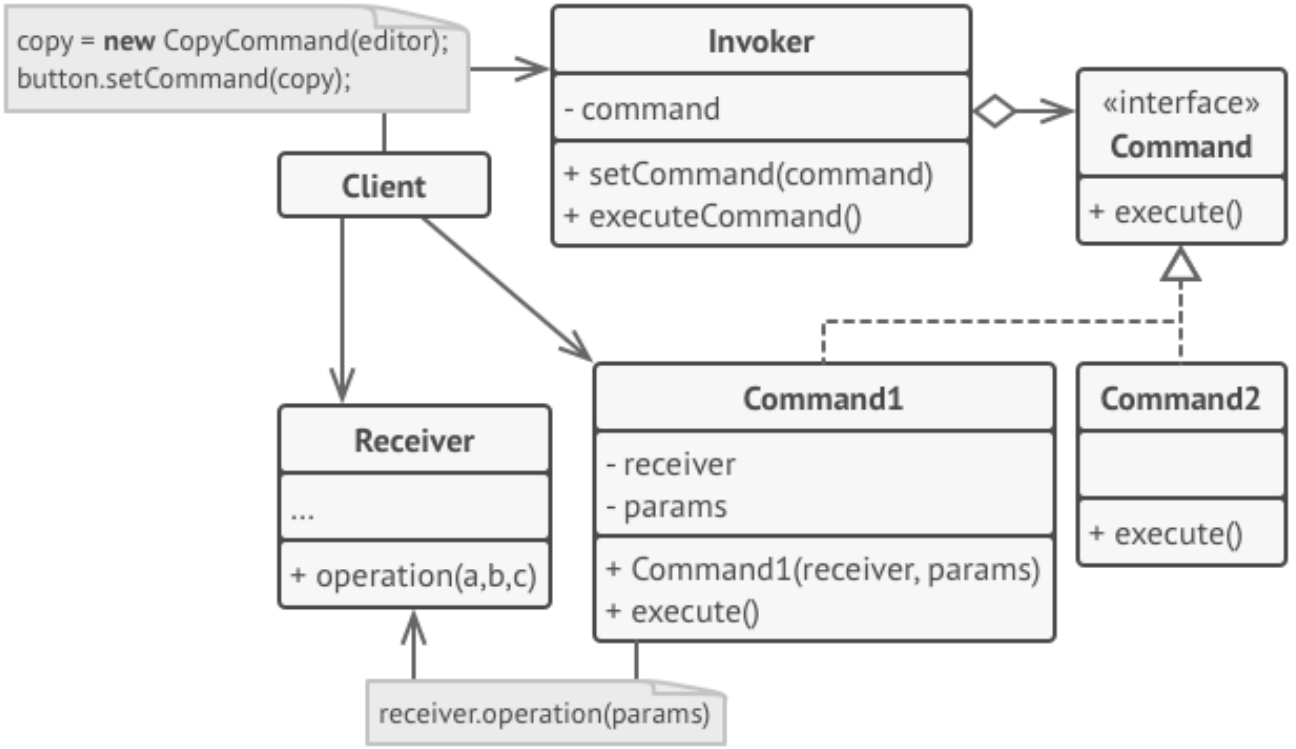




Advanced Refactoring (Design Patterns)

Command

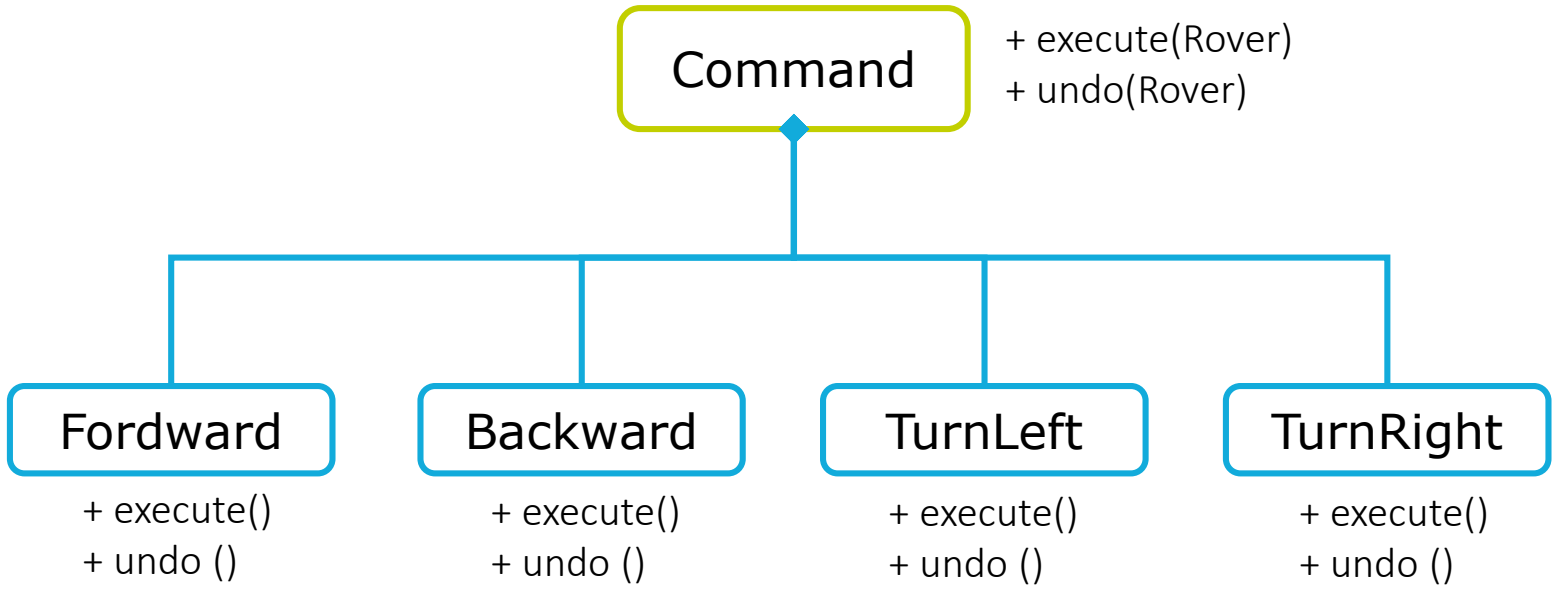
Permite codificar operaciones como objetos independientes.
Es muy útil cuando se quiere realizar un histórico de operaciones por ejemplo colas o se quiere para deshacer operaciones.



Si aparece un nuevo comando, tan solo hay que implementar el comportamiento de este comando y hacer que el cliente lo ejecute cuando sea necesario.

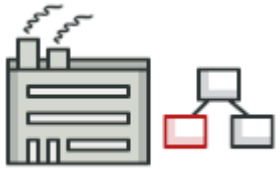
Advanced Refactoring (Design Patterns)

Command (Implementación)

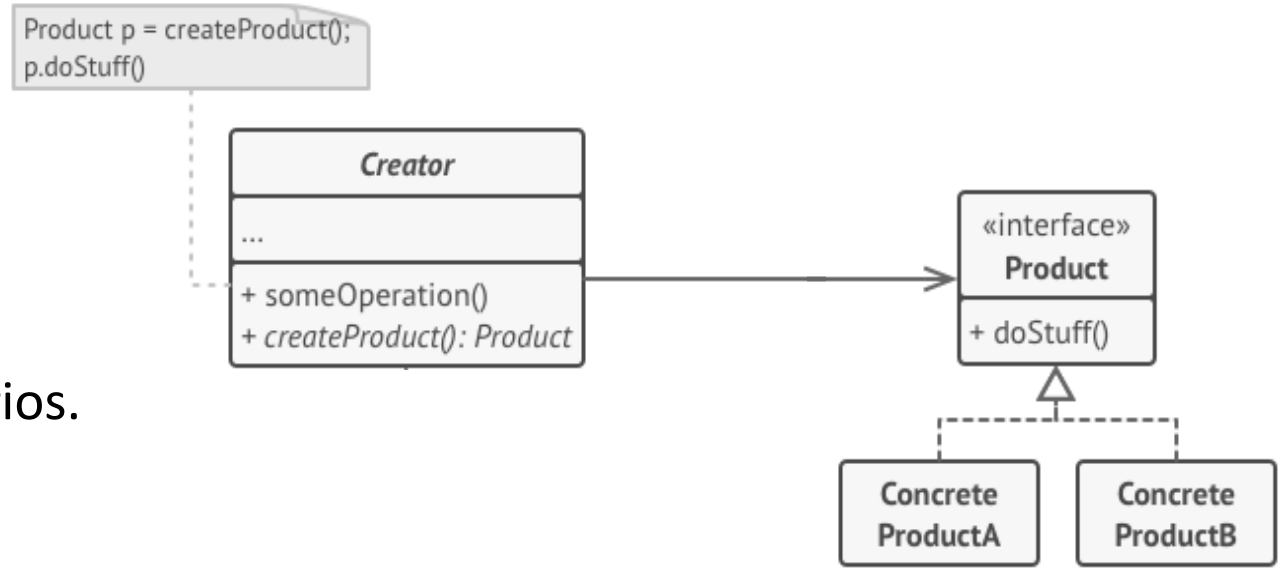


Advanced Refactoring (Design Patterns)

Abstract Factory



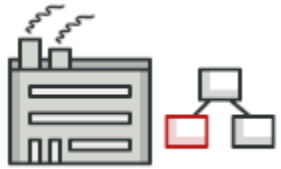
El patrón ‘Factory’ nos permite centralizar la lógica de creación de instancias de un objeto padre en función de una reglas o criterios. De esta forma evitamos tener que replicar código de creación en varios puntos.

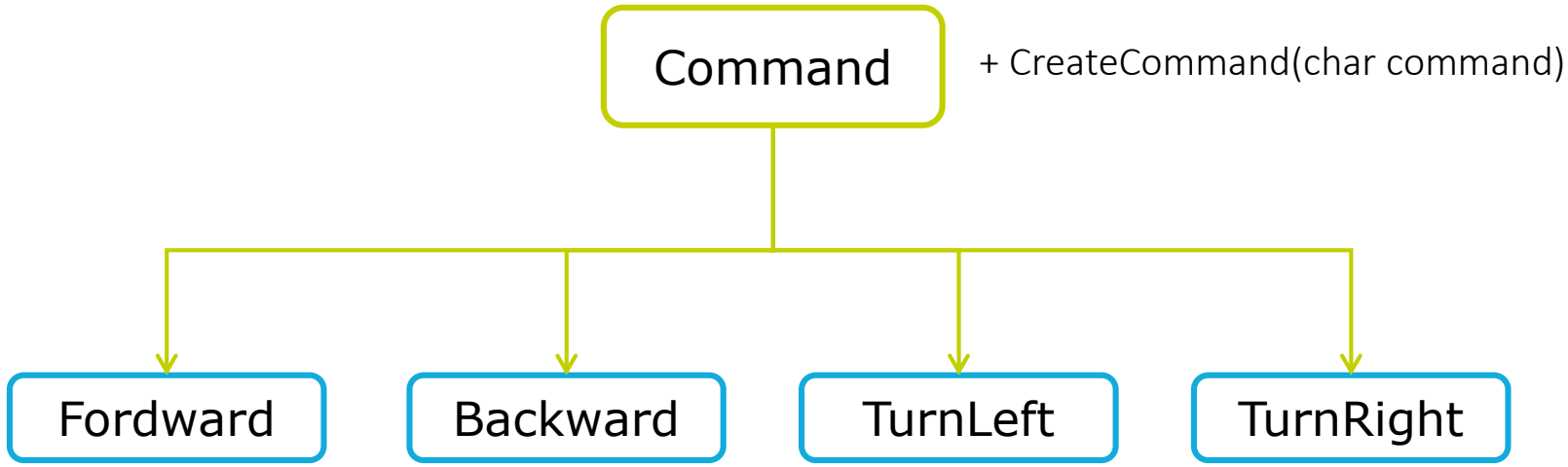


Si aparece una nueva clase concreta, tan solo hay que añadir esa clase con sus reglas, al método de creación.

Advanced Refactoring (Design Patterns)

Abstract Factory (Implementación)





devonfw

People matter, results count.

About Capgemini

With more than 190,000 people, Capgemini is present in over 40 countries and celebrates its 50th Anniversary year in 2017. A global leader in consulting, technology and outsourcing services, the Group reported 2016 global revenues of EUR 12.5 billion. Together with its clients, Capgemini creates and delivers business, technology and digital solutions that fit their needs, enabling them to achieve innovation and competitiveness. A deeply multicultural organization, Capgemini has developed its own way of working, the Collaborative Business Experience™, and draws on Rightshore®, its worldwide delivery model.

Learn more about us at

www.capgemini.com