POLITÉCNICO COLOMBIANO
JAIME ISAZA CADAVID

Patrón de comportamiento

ESTRATEGIA

Realizado por
Sebastian Zapata Castaño

Dirigido por
Hector Manuel Vanegas Solis

Universidad
Politecnico Colombiano Jaime Isaza Cadavid

Enero, 2022

Descripcion general

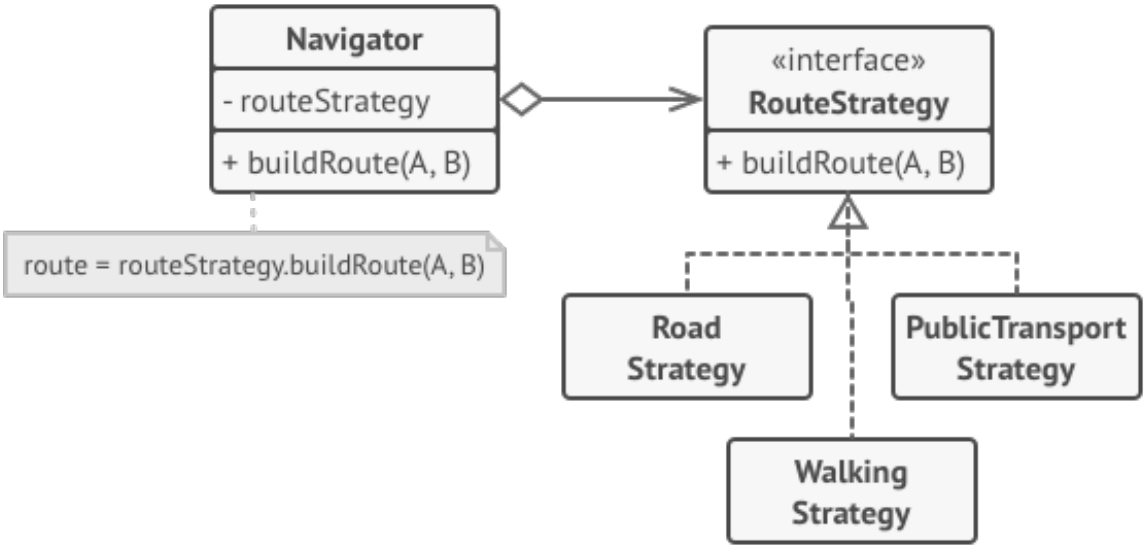
En la programación orientada a objetos, los design patterns, o patrones de diseño, asisten los desarrolladores con estrategias y plantillas de soluciones cuya eficacia ha sido probada. Una vez encontrado el esquema de resolución apropiado, todo lo que queda es hacer ajustes individuales. Actualmente, hay un total de 70 patrones de diseño adaptados a áreas específicas de aplicación. Los patrones strategy se centran en el comportamiento del software. Strategy es un patrón de diseño de comportamiento que le permite definir un grupo de algoritmos poner cada algoritmo en una clase separada y hacer que sus ojetos sean intercamiales.

¿Que soluiona este patrón de diseño?

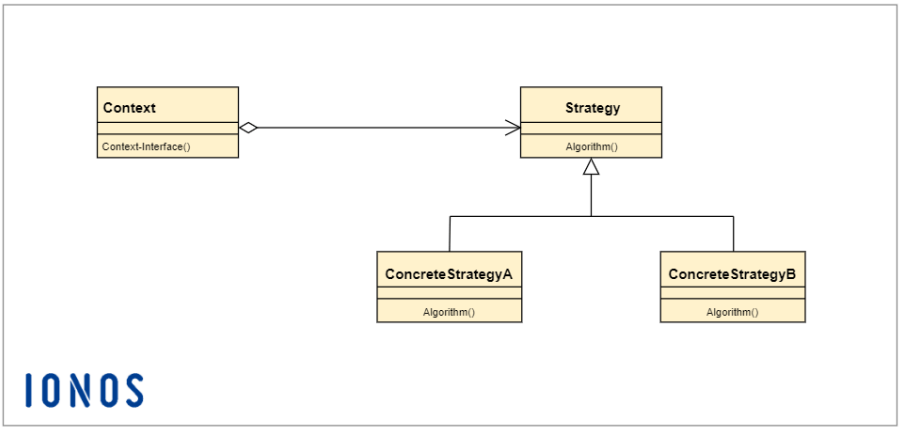
Supongamos que tenemos una aplicacion de rutas, al incio funciona super bien para calcular la ruta si vas en automvil, pero ¿que pasa con los que se van caminando? tendrías que implementar otro metodo que calcule la ruta para estas personas, y despues ocurriría lo mismo para los que se van en bicicleta o en trasnporte público, llegaría a estar muy saturado y con facilidad de errores.

El patrón Strategy sugiere que tomes esa clase que hace algo específico de muchas formas diferentes y extraigas todos esos algoritmos para colocarlos en clases separadas llamadas estrategias. La clase original, llamada contexto, debe tener un campo para almacenar una referencia a una de las estrategias. El contexto delega el trabajo a un objeto de estrategia vinculado en lugar de ejecutarlo por su cuenta. El patrón Strategy sugiere que tomes esa clase que hace algo específico de muchas formas diferentes y extraigas todos esos algoritmos para colocarlos en clases separadas llamadas estrategias.

La clase original, llamada contexto, debe tener un campo para almacenar una referencia a una de las estrategias. El contexto delega el trabajo a un objeto de estrategia vinculado en lugar de ejecutarlo por su cuenta.



Builder como diagrama UML



Ventajas y desventajas de Estrategia

0.0.1. Ventajas del patrón de comportamiento Estrategia

Permite cambiar el comportamiento de la clase de manera dinámica sin poner condiciones dentro de la clase contexto. Permite encapsular comportamiento para reutilizarlo y no repetir código. Permite modificar comportamiento sin tener que modificar las clases de contexto.

0.0.2. Desventajas del patrón de comportamiento Estrategia

La aplicación debe estar al tanto de todas las posibles estrategias para seleccionar la adecuada para cada situación. Todas las estrategias deben implementar la misma interfaz, esto hace que algunas estrategias tengan que implementar métodos que no necesitan para su comportamiento.

Ejemplo en codigo del patron de diseño Builder

Context:

```
public class Context {
    // Valor por defecto (comportamiento por defecto): ConcreteStrategyA
    private Strategy strategy = new ConcreteStrategyA();
    public void execute() {
        //delega el comportamiento a un objeto de estrategia
        strategy.executeAlgorithm();
    }
    public void setStrategy(Strategy strategy) {
        strategy = strategy;
    }
    public Strategy getStrategy() {
        return strategy;
    }
}
```

Strategy, ConcreteStrategyA, ConcreteStrategyB:

```
interface Strategy {
    public void executeAlgorithm();
}
class ConcreteStrategyA implements Strategy {
    public void executeAlgorithm() {
        System.out.println("Concrete Strategy A");
    }
}
class ConcreteStrategyB implements Strategy {
    public void executeAlgorithm() {
        System.out.println("Concrete Strategy B");
    }
}
```

Client:

```
public class Client {
    public static void main(String[] args) {
        //Comportamiento por defecto
        Context context = new Context();
        context.execute();
        //Cambiar el comportamiento
        context.setStrategy(new ConcreteStrategyB());
        context.execute();
    }
}
```

Referencias

- Ortega Mateo, E. (2020, January 2). Patrones de comportamiento - Strategy. Somos PNT - Desarrollamos Software. https://somospnt.com/blog/136-patrones-de-comportam
- Strategy. (n.d.). Refactoring.guru. Retrieved January 27, 2022, from https://refactoring.guru/es/design-patterns/strategy
- Strategy pattern: un patrón de diseño de software para estrategias variables de comportamiento. (n.d.). IONOS Digitalguide. Retrieved January 27, 2022, from https://www.ionos.es/digitalguide/paginas-web/desarrollo-web/strategy-pattern/