



Programação Avançada

Strategy Pattern

Programação Avançada – 2020-21

Bruno Silva, Patrícia Macedo

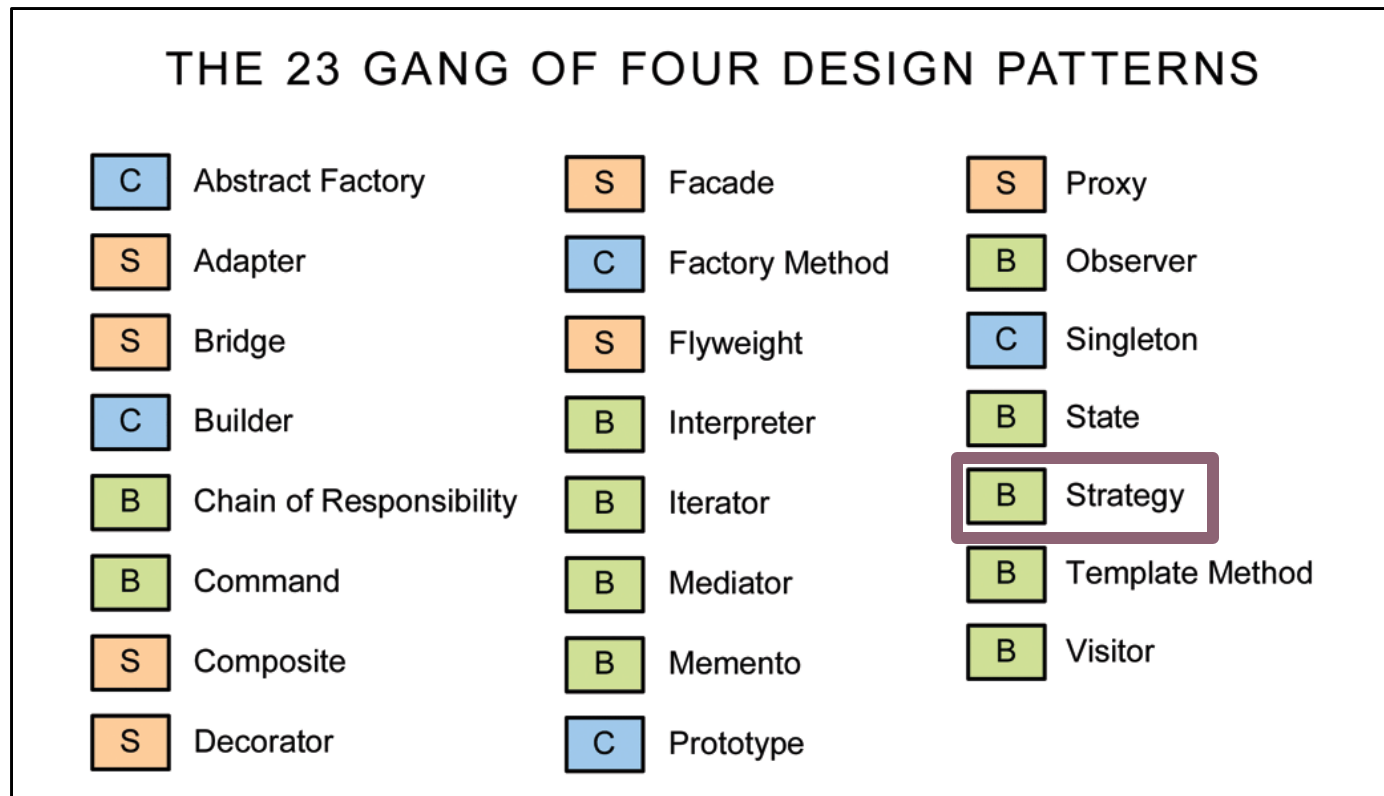
Sumário



- Pattern Strategy
- Exmplo de Aplicação

Strategy Pattern

- Padrão comportamental
- Tal como o nome sugere permite que um objecto **troque de estratégia** de comportamento em tempo de execução.



Strategy (Motivação)

- **Problema Genérico**

- Existe uma classe que o seu **comportamento varia** dependendo das circunstâncias ou do seu tipo.
- Para tal recorremos a uma estrutura **switch case** e em função do tipo, executamos um conjunto de instruções código diferente.
- Como garantir a extensibilidade e manutenibilidade?

Nota:

O uso de enumerados para representar tipos diferentes de classes, vai contra as boas práticas da programação orientada a objetos e é de evitar:

- Mais difícil de manter
- Mais difícil de estender

```
public class X {  
  
    public enum TYPE {TYPE1,TYPE2,TYPE3};  
    private TYPE type;  
    private int id;  
    //...  
  
    public X(int id, TYPE type) {  
        this.id = id;  
        this.type=type;  
        //....  
    }  
  
    public void metodo1()    {  
        switch (type) {  
            case TYPE1:  
                //code1  
                break;  
            case TYPE2:  
                //code2  
                break;  
            case TYPE3:  
                //code3  
                break;  
        }  
    }  
}
```

Strategy (Motivação)

Problema concreto

- Um grupo é constituído por programadores e pretende-se calcular o índice de potencial de um grupo
- A formula de calculo, varia em função do perfil pretendido para o grupo
- É possível o grupo mudar de perfil, ao longo do tempo.
- Como garantir que temos uma aplicação onde é fácil
 - **adicionar/remover novos tipos de perfis**
 - **adicionar novos métodos que variam em função do perfil do grupo**

```
public class Group {  
  
    public enum TYPE {DIVERSITY, SENIOR, MULTYSKILLS}  
    private TYPE type;  
    private String name;  
    private Map<Integer, Programmer> personList;  
  
    public Group(String name, Group.TYPE type) {  
        this.name = name;  
        this.type = type;  
        personList = new HashMap<>();  
    }  
    public float calculateGlobalIndex() {  
        int value = 0;  
        float res = 0.0f;  
        switch (type) {  
  
            case DIVERSITY:  
                /* res= X/Y  
                X= programmers with more than 5 years of experience  
                Y= programmers with less than 5 years of experience */  
  
            case SENIOR:  
                /* res= X/Y  
                X= programmers with more than 10 years of experience  
                Y= group size*/  
  
            case MULTYSKILLS:  
                /* res= X/Y  
                X= programmers with specialist in more than 5 languages  
                Y= group size*/  
  
        }  
        return res;  
    }  
}
```

Strategy (Motivação)

Problema concreto

- Pretende-se calcular o índice global de potencial de um grupo
- A formula de calculo, varia em função do perfil do grupo
- É possível o perfil associado a um grupo mudar ao longo da vida do grupo
- Como garantir que temos uma aplicação onde é fácil
 - A. adicionar/remover novos tipos de perfis**
 - B. adicionar novos métodos que variam em função do perfil do grupo**

Solução 1

- A. Adicionar mais um valor ao enumerado
- B. Adicionar mais um case no método `float calculateGlobalIndex()`

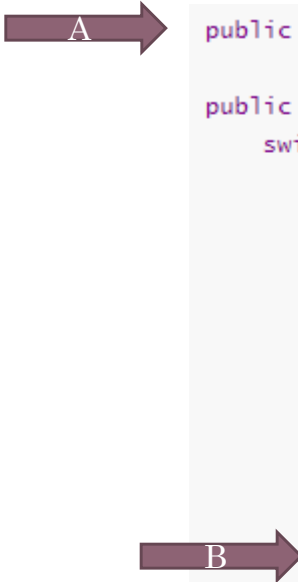
Solução 2 - polimorfismo

- A. Definir a class `Group` como abstrata e o método `float calculateGlobalIndex()` como abstrato
- B. Implementar três subclasses de `Group` uma para cada perfil do grupo

Strategy (Motivação)

Solução 1

- A. Adicionar mais um valor ao enumerado
- B. Adicionar mais um case no método float `calculateGlobalIndex()`



```
public enum TYPE {DIVERSITY, SENIOR, MULTYSKILS, SPECIALIZED};

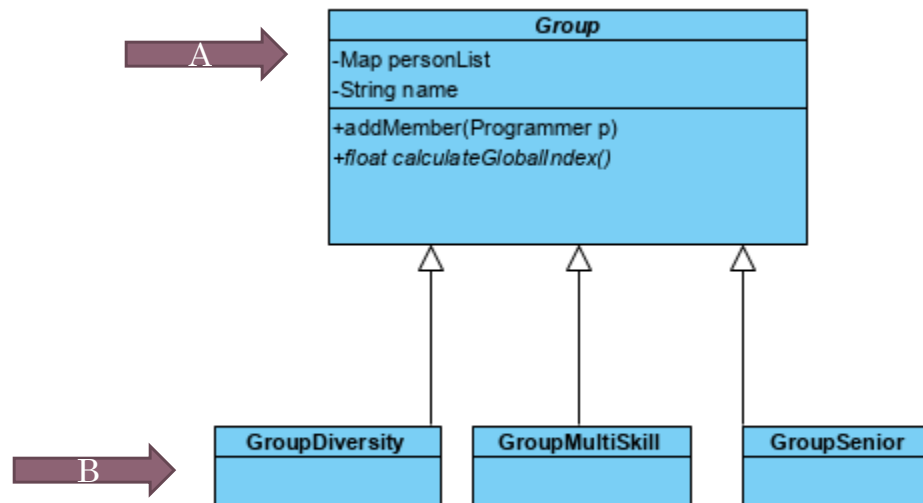
public float calculateGlobalIndex() {
    switch (type) {

        case DIVERSITY:
            //code 1
            break;
        case SENIOR:
            // code 2
            break;
        case MULTYSKILS:
            //code 3
            break;
        case SPECIALIZED:
            //code 4
            break;
    }
    return res;
}
```

Strategy (Motivação)

Solução 2 – polimorfismo

- A. Definir a class Group como abstrata e o método float calculateGlobalIndex() como abstrato.
- B. Implementar três subclasses de Group uma para cada perfil do grupo



Strategy (Motivação)

No caso do Polimorfismo Se o grupo quisesse mudar de perfil em tempo de execução, teria que se instanciar um novo grupo, perdendo-se os dados do anterior.

```
Group gr1 = new GroupDiversity("PA-23");
gr1.addMember(p1,p2,p3,p4);

System.out.printf("\nGrupo %s , GlobalIndex- %f", gr1.toString(),gr1.calculateGlobalIndex());

gr1 = new GroupMultiSkills("PA-23"); ←
System.out.printf("\nGrupo %s , GlobalIndex- %f", gr1.toString(),gr1.calculateGlobalIndex());

gr1 = new GroupStrategy("PA-23"); ←
System.out.printf("\nGrupo %s , GlobalIndex- %f", gr1.toString(),gr1.calculateGlobalIndex());
```

Será que existe Outra Solução melhor?

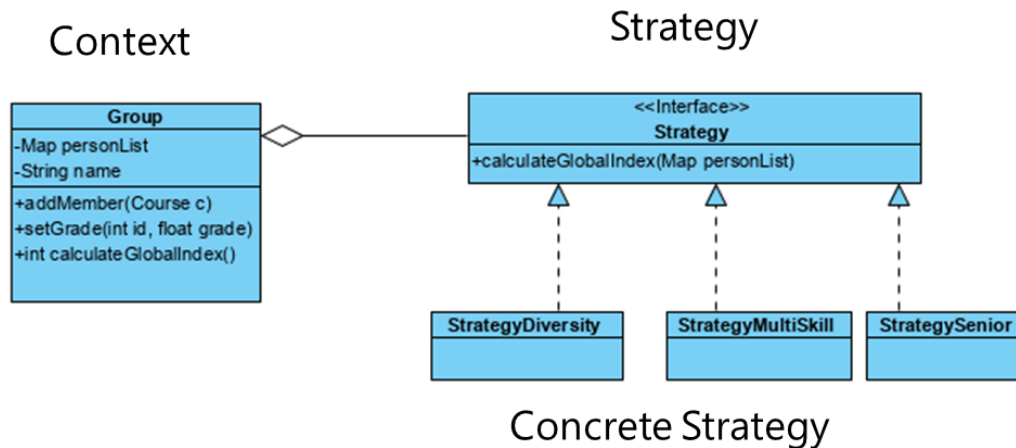
- Sim !! **Aplicar o padrão Strategy**

Solução : Strategy

Strategy: declara uma **interface** comum a todos os algoritmos suportados.

Context: usa essa interface para chamar o algoritmo definido por uma estratégia concreta (ConcreteStrategy).

ConcreteStrategy: Classes **que** implementam o algoritmo segundo uma estratégia específica.



Strategy e Concrete Strategy

```
public interface Strategy {  
  
    public float calculateGlobalIndex(Map<Integer, Programmer> personList);  
  
}
```

```
public class StrategyDiversity implements Strategy {  
    @Override  
    public float calculateGlobalIndex(Map<Integer, Programmer> personList){  
        int countYoung=0, countOld=0;  
        for (Programmer programmer : personList.values()) {  
  
            if(programmer.getYearsOfExperience()>5) countOld++;  
            if(programmer.getYearsOfExperience()<=5) countYoung++;  
        }  
        return countYoung*1.f/countOld;  
    }  
}
```

```
public class StrategySenior implements Strategy {  
    @Override  
    public float calculateGlobalIndex(Map<Integer, Programmer> personList){  
        int countOld=0;  
        for (Programmer programmer : personList.values()) {  
            if(programmer.getYearsOfExperience()>10) countOld++;  
        }  
        return countOld*1.f/personList.size();  
    }  
}
```

Context

```
public class Group {

    private Strategy strategy;
    private String name;
    private Map<Integer, Programmer> personList;

    private static Random random = new Random();

    public Group(String name, Strategy strategy) {
        this.name = name;
        this.strategy=strategy;
        this.personList = new HashMap<>();
    }
    public void setStrategy(Strategy strategy) {
        this.strategy = strategy;
    }
    public float calculateGlobalIndex() {
        return strategy.calculateGlobalIndex(personList);
    }
}
```

1. Define um atributo do tipo **strategy** e é inicializado no construtor.
2. O perfil do grupo (estratégia) pode ser alterado
3. O método **delega** na **estratégia instanciada** o calculo do índice global

Client – (Main)

```
public class MainStrategy {  
  
    public static void main(String[] args) {  
  
        Programmer p1= new Programmer(1, "Ana",5,2);  
        Programmer p2= new Programmer(2, "Rui",15,8);  
        Programmer p3= new Programmer(3, "Paula",22,9);  
        Programmer p4= new Programmer(4, "Luis",5,6);  
        Group gr1 = new Group("PA-23", new StrategyDiversity());  
        gr1.addMember(p1,p2,p3,p4);  
        System.out.printf("\nGrupo %s , GlobalIndex- %f", gr1.toString(),gr1.calculateGlobalIndex());  
        gr1.setStrategy(new StrategyMultiSkill());  
        System.out.printf("\nGrupo %s , GlobalIndex- %f", gr1.toString(),gr1.calculateGlobalIndex());  
        gr1.setStrategy(new StrategySenior());  
        System.out.printf("\nGrupo %s , GlobalIndex- %f", gr1.toString(),gr1.calculateGlobalIndex());  
  
    }  
}
```

- O grupo altera o seu perfil em tempo de execução

Strategy| Exercícios(1)



Crie um projeto a partir do template:

https://github.com/pa-estsetubal-ips-pt/GroupSelection_Template.git

e analise o código disponibilizado, com as soluções completas apresentadas ao longo destes slides.

- Adicione um novo perfil : SPECIALIZED, em o índice é calculado da seguinte formula:

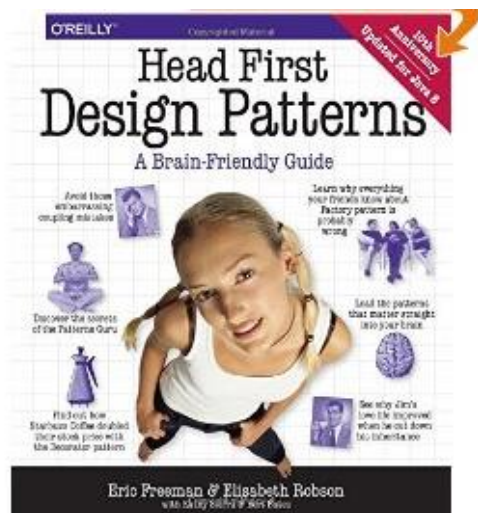
$$X / (Y - X)$$

X número de pessoas com mais de 5 anos de experiência e especializadas no máximo em 3 linguagens

Y número total de pessoas

- Adicione um novo método, para seleção do chefe do grupo que também difere consoante o perfil pretendido
 - SENIOR - o membro com mais anos
 - DIVERSITY - Random
 - MULTISKILLS - o membro com mais linguagens de programação, em caso de empate o mais novo
 - SPECIALIZED - dos membros com mais de 5 anos de experiencia, o que domine menos linguagens de programação. Implemente o código necessário para disponibilizar o método `Programmer selectLeader(Map<Integer,Programmer> personList)` que devolve o membro que será Lider

Rever



Paginas: 10 - 24

- <https://refactoring.guru/design-patterns/strategy>
- <http://www.dofactory.com/net/strategy-design-pattern>