Modèles de conception - Modèle de stratégie

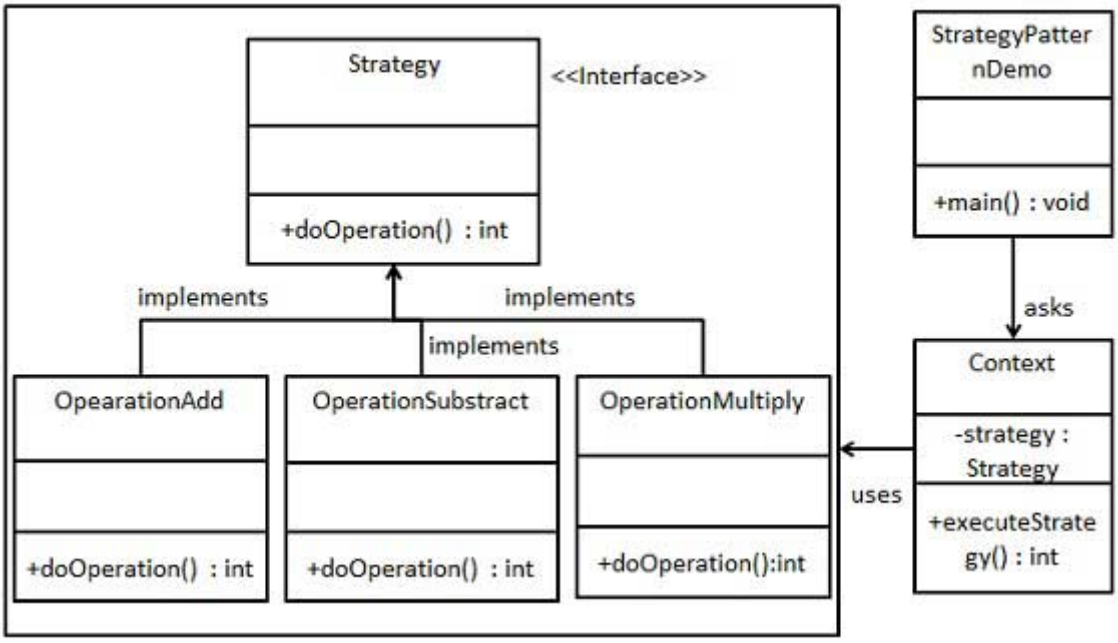
Dans le modèle de stratégie, un comportement de classe ou son algorithme peut être modifié au moment de l'exécution. Ce type de modèle de conception relève du modèle de comportement.

Dans le modèle de stratégie, nous créons des objets qui représentent diverses stratégies et un objet de contexte dont le comportement varie selon son objet de stratégie. L'objet de stratégie modifie l'algorithme d'exécution de l'objet de contexte.

la mise en oeuvre

Nous allons créer une interface *Stratégie* définissant une action et des classes de stratégie concrètes mettant en œuvre l' interface *Stratégie* . *Le contexte* est une classe qui utilise une stratégie.

StrategyPatternDemo , notre classe de démonstration, utilisera le *Contexte* et les objets de stratégie pour démontrer le changement de comportement du Contexte en fonction de la stratégie qu'il déploie ou utilise.



Étape 1

Créez une interface.

Strategy.java

```
public interface Strategy {
   public int doOperation(int num1, int num2);
}
```

Étape 2

Créez des classes concrètes implémentant la même interface.

OperationAdd.java

```
public class OperationAdd implements Strategy{
   @Override
   public int doOperation(int num1, int num2) {
      return num1 + num2;
   }
}
```

OperationSubstract.java

```
public class OperationSubstract implements Strategy{
   @Override
   public int doOperation(int num1, int num2) {
      return num1 - num2;
   }
}
```

OperationMultiply.java

```
public class OperationMultiply implements Strategy{
   @Override
   public int doOperation(int num1, int num2) {
      return num1 * num2;
   }
}
```

Étape 3

Créer une classe de *contexte* .

Context.java

```
public class Context {
   private Strategy strategy;

   public Context(Strategy strategy){
      this.strategy = strategy;
   }

   public int executeStrategy(int num1, int num2){
      return strategy.doOperation(num1, num2);
   }
}
```

Étape 4

Utilisez le *contexte* pour voir le changement de comportement lorsqu'il modifie sa *stratégie* .

StrategyPatternDemo.java

```
public class StrategyPatternDemo {
   public static void main(String[] args) {
      Context context = new Context(new OperationAdd());
```

```
      System.out.println("10 + 5 = " + context.executeStrategy(10, 5));

      context = new Context(new OperationSubstract());
      System.out.println("10 - 5 = " + context.executeStrategy(10, 5));

      context = new Context(new OperationMultiply());
      System.out.println("10 * 5 = " + context.executeStrategy(10, 5));
   }
}
```

Étape 5

Vérifiez la sortie.

```
10 + 5 = 15
10 - 5 = 5
10 * 5 = 50
```