Паршаков Геннадий ПИ-15-1

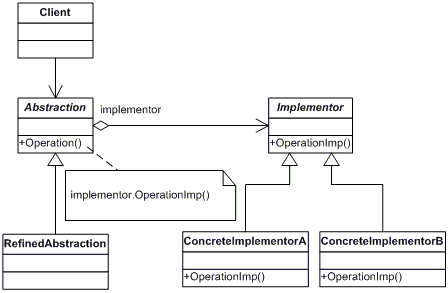
Паттерн «Мост»

# Назначение и цель применения паттерна

Мост (Bridge) - структурный шаблон проектирования, который позволяет отделить абстракцию от реализации таким образом, чтобы и абстракцию, и реализацию можно было изменять независимо друг от друга.

При частом изменении класса преимущества объектно-ориентированного подхода становятся очень полезными, позволяя делать изменения в программе, обладая минимальными сведениями о реализации программы. Шаблон «Мост» является полезным там, где часто меняется не только сам класс, но и то, что он делает.

# UML-диаграмма



* **Abstraction**: определяет базовый интерфейс и хранит ссылку на объект Implementor. Выполнение операций в Abstraction делегируется методам объекта Implementor
* **RefinedAbstraction**: уточненная абстракция, наследуется от Abstraction и расширяет унаследованный интерфейс
* **Implementor**: определяет базовый интерфейс для конкретных реализаций. Как правило, Implementor определяет только примитивные операции. Более сложные операции, которые базируются на примитивных, определяются в Abstraction
* **ConcreteImplementorA** и **ConcreteImplementorB**: конкретные реализации, которые унаследованы от Implementor
* **Client**: использует объекты Abstraction

# Области применения

Данный шаблон используется для:

* независимо изменения интерфейса работы с клиентом и реализацией;
* (или) выбора реализации в процессе работы программы;
* (или) использования одной реализации в нескольких абстракциях;
* (или) уменьшения числа классов, получающихся при использовании наследования.

# Особенности паттерна

Плюсы:

* Позволяет строить платформо-независимые программы.
* Скрывает лишние или опасные детали реализации от клиентского кода.
* Реализует принцип открытости/закрытости.

Минусы:

* Усложняет код программы за счёт дополнительных классов.

# Примеры реализации

class Client

{

    static void Main()

    {

        Abstraction abstraction;

        abstraction = new RefinedAbstraction(new ConcreteImplementorA());

        abstraction.Operation();

        abstraction.Implementor=new ConcreteImplementorB();

        abstraction.Operation();

    }

}

abstract class Abstraction

{

    protected Implementor implementor;

    public Implementor Implementor

    {

        set { implementor = value; }

    }

    public Abstraction(Implementor imp)

    {

        implementor = imp;

    }

    public virtual void Operation()

    {

        implementor.OperationImp();

    }

}

abstract class Implementor

{

    public abstract void OperationImp();

}

class RefinedAbstraction : Abstraction

{

    public RefinedAbstraction(Implementor imp)

        : base(imp)

    {}

    public override void Operation()

    {

    }

}

class ConcreteImplementorA : Implementor

{

    public override void OperationImp()

    {

    }

}

class ConcreteImplementorB : Implementor

{

    public override void OperationImp()

    {

    }

}

# Список источников

1. <https://ru.wikipedia.org/wiki/Мост_(шаблон_проектирования)>
2. <https://refactoring.guru/ru/design-patterns/bridge>
3. <https://metanit.com/sharp/patterns/4.6.php>
4. http://andrey.moveax.ru/post/patterns-oop-structural-bridge