|  |
| --- |
| Пермский филиал федерального государственного автономного образовательного учреждения высшего образования  «Национальный исследовательский университет  «Высшая школа экономики»  *Факультет экономики, менеджмента и бизнес-информатики* |
|  |
| Демина Дарья Сергеевна  **СТРУКТУРНЫЙ ПАТТЕРН «МОСТ»**  Лабораторная работа  по направлению подготовки *09.03.04 Программная инженерия*  образовательная программа «Программная инженерия»   |  |  | | --- | --- | |  | Руководитель  Доцент кафедры информационных технологий в бизнесе.  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  А.В. Кычкин | |

Пермь, 2018 год

# Назначение и цель применения паттерна Bridge

Мост – структурный паттерн, который используется для разделения одного или нескольких классов на две отдельные иерархии – абстракцию и реализацию, для изменения каждой из них независимо друг от друга.

Абстракция – графический интерфейс программы. Он не делает работу самостоятельно, а делегирует ее «Реализации».

Реализация – API к которому интерфейс обращается по реакции на действия пользователя.

Преимущества:

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

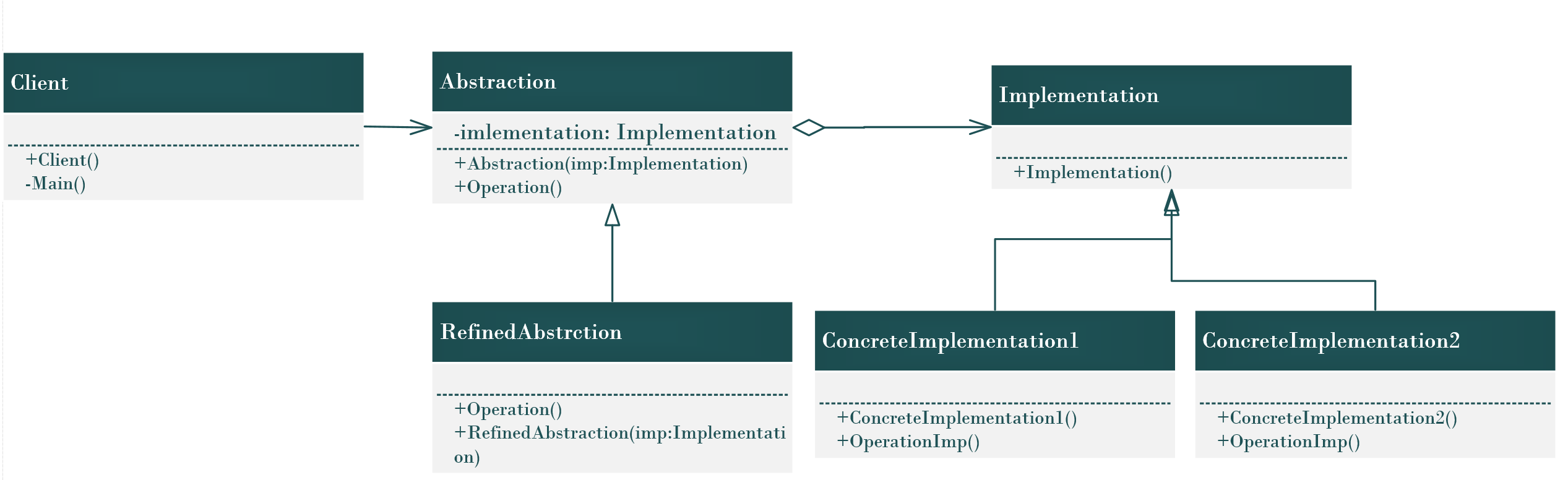
Недостатки:

* Усложняет ко за счет дополнительных классов.

Цели применения:

1. Разделение класса, который содержит несколько различных реализаций.
2. Расширение класса в двух независимых плоскостях.
3. Изменение реализации во время выполнения программы (не основная цель применения данного паттерна).

# UML-диаграмма



Client – работает только с объектами абстракции. Не имеет прямого доступа к объектам реализации.

Abstraction - определяет базовый интерфейс и содержит управляющую логику. Делегирует выполнение операций связному объекту реализации.

RefinedAbstraction – уточненная абстракция, расширяет унаследованный интерфейс.

Implementation – задает общий интерфейс для всех реализаций. Здесь находятся базовые операции, на которых строятся сложные операции абстракции.

ConcreteImplementation1, ConcreteImplementation2 – конкретные реализации, содержат платформо-зависимый код.

# Области применения

Архитектура [Java](https://ru.wikipedia.org/wiki/Java) [AWT](https://ru.wikipedia.org/wiki/Abstract_Window_Toolkit) полностью основана на этом шаблоне — иерархия java.awt.xxx для хэндлов и sun.awt.xxx для реализаций.Особенности паттерна

Abstract Window Toolkit (AWT) — это исходная платформо-независимая оконная библиотека графического интерфейса языка [Java](https://ru.wikipedia.org/wiki/Java). Сейчас AWT является частью [Java Foundation Classes](https://ru.wikipedia.org/wiki/Java_Foundation_Classes) (JFC) — стандартного [API](https://ru.wikipedia.org/wiki/Application_programming_interface) для реализации [графического интерфейса](https://ru.wikipedia.org/wiki/GUI) в Java-программе.

Также AWT является стандартной библиотекой GUI для некоторых профилей [Java ME](https://ru.wikipedia.org/wiki/Java_Platform,_Micro_Edition). Например, профили [Connected Device Configuration](https://ru.wikipedia.org/w/index.php?title=Connected_Device_Configuration&action=edit&redlink=1) требуют от среды исполнения Java на [мобильных телефонах](https://ru.wikipedia.org/wiki/%D0%9C%D0%BE%D0%B1%D0%B8%D0%BB%D1%8C%D0%BD%D1%8B%D0%B9_%D1%82%D0%B5%D0%BB%D0%B5%D1%84%D0%BE%D0%BD) поддержки AWT.

# Особенности паттерна

Особенность паттерна заключается в том, что его общая реализация состоит в объявлении классов абстракций и классов реализаций в отдельных параллельных иерархиях классов.

# Пример реализации

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace BridgePattern

{

class Program

{

static void Main(string[] args)

{

//Начальная инициализация с первым вариантом реализации

RefinedAbstracion refinedAbstraction = new RefinedAbstracion(new ConcretteImplementorA());

refinedAbstraction.Operation();

//Изменяем используемый вариант реализации

refinedAbstraction.Implementor = new ConcretteImplementorB();

refinedAbstraction.Operation();

Console.ReadKey();

}

}

//базовый класс абстракции

abstract class Abstraction

{

private Implementor \_imp;

public Implementor Implementor { set { \_imp = value; } }

public Abstraction(Implementor implementor)

{

\_imp = implementor;

}

public virtual void Operation()

{

\_imp.OperationImp();

}

}

//класс уточненной абстракции

class RefinedAbstracion : Abstraction

{

public RefinedAbstracion(Implementor implementor) : base(implementor)

{ }

public override void Operation()

{

base.Operation();

}

}

//Интерфейс реализации

interface Implementor

{

void OperationImp();

}

class ConcretteImplementorA : Implementor

{

public void OperationImp()

{

Console.WriteLine("Первый вариант реализации");

int a = 3, b = 9, c;

c = a / b;

Console.WriteLine(c);

}

}

class ConcretteImplementorB : Implementor

{

public void OperationImp()

{

Console.WriteLine("Второй вариант реализации");

double a=3, b=9,c;

c = a / b;

Console.WriteLine(c);

}

}

}

# Список источников

* + 1. Мост (Bridge) [Электронный ресурс] //METANIT.COM – сайт о программировании: [сайт]. [2012-2017]. URL: https://metanit.com/sharp/patterns/4.6.php (дата обращения: 06.02.2018).
    2. Мост (шаблон проектирования) [Электронный ресурс] //Википедия: [сайт]. [2017]. URL: https://ru.wikipedia.org/wiki/Мост\_(шаблон\_проектирования) (дата обращения: 06.02.2018).
    3. Мост [Электронный ресурс] //refactoring guru: [сайт]. [2014-2018]. URL: https://refactoring.guru/ru/design-patterns/bridge (дата обращения: 06.02.2018).