Министерство образования и науки Российской Федерации

Севастопольский государственный университет

Кафедра ИС

Отчет

По дисциплине: “Рефакторинг программного обеспечения”

Лабораторная работа №5

“Исследование способов применения структурных паттернов

проектирования при рефакторинге ПО”

Выполнил:

ст.гр. ИС/б-17-2

Черняев Н.Г.

Проверил:

Строганов В.А.

Севастополь

2020

1 ЦЕЛЬ РАБОТЫ

Исследовать возможность использования структурных паттернов проектирования. Получить практические навыки применения структурных паттернов при объектно-ориентированном проектировании и рефакторинге ПО.

2 ХОД РАБОТЫ

Предположим, что существует приложение для торговли на бирже. Наше приложение скачивает биржевые данные из несколько источников в формате XML, а затем рисует красивые графики. Пример кода приложения представлен ниже:

class Program

{

    void Main()

    {

        string dataXML = getXMLData();

        IGraph graph = new Graph();

        graph.DrawGraph(dataXML);

    }

    void getXMLData()

    {

*// parse XML data*

    }

}

interface IGraph

{

    void DrawGraph();

}

class Graph: IGraph

{

    public void DrawGraph(string dataXML)

    {

*// draw graph from XML data*

    }

}

В какой-то момент мы решаем улучшить приложение, применим стороннюю библиотеку для отрисовки красивых графиков. Пример стороннего класса представлен ниже:

class SuperGraph

{

    public void DrawGraphFromJSONData()

    {

*// draw graph from JSON data*

    }

}

Но проблема в том, что новая сторонняя библиотека не поддерживает формат данных XML. Мы могли бы переписать данную библиотеку, чтобы она поддерживала нужный формат, но это может нарушить работу существующего кода. Либо, бывают случаи, когда разработчику не доступен исходный код библиотеки.

Тогда можно воспользоваться паттерном “Адаптер” и создать класс-адаптер, который будет оборачивать объект класса SuperGraph и перед вызовом необходимого метода преобразовывал входные XML данные в формат JSON.

Результат применения паттерна “Адаптер” представлен ниже:

class Program

{

    void Main()

    {

        string dataXML = getXMLData();

        IGraph graph = new Graph();

        graph.DrawGraph(dataXML);

    }

    void getXMLData()

    {

*// parse XML data*

    }

}

interface IGraph

{

    void DrawGraph();

}

class Graph: IGraph

{

    private SuperGraph superGraph = new SuperGraph();

    public void DrawGraph(string dataXML)

    {

        string dataJSON = ConvertToJSON(dataXML); *// pseudo method*

        superGraph.DrawGraphFromJSONData();

    }

}

class SuperGraph

{

    public void DrawGraphFromJSONData()

    {

*// draw graph from JSON data*

    }

}

Либо второй вариант класса-адаптера через наследование:

…

class Graph: SuperGraph, IGraph

{

    public void DrawGraph(string dataXML)

    {

        string dataJSON = ConvertToJSON(dataXML); *// pseudo method*

*this*.DrawGraphFromJSONData();

    }

}

…

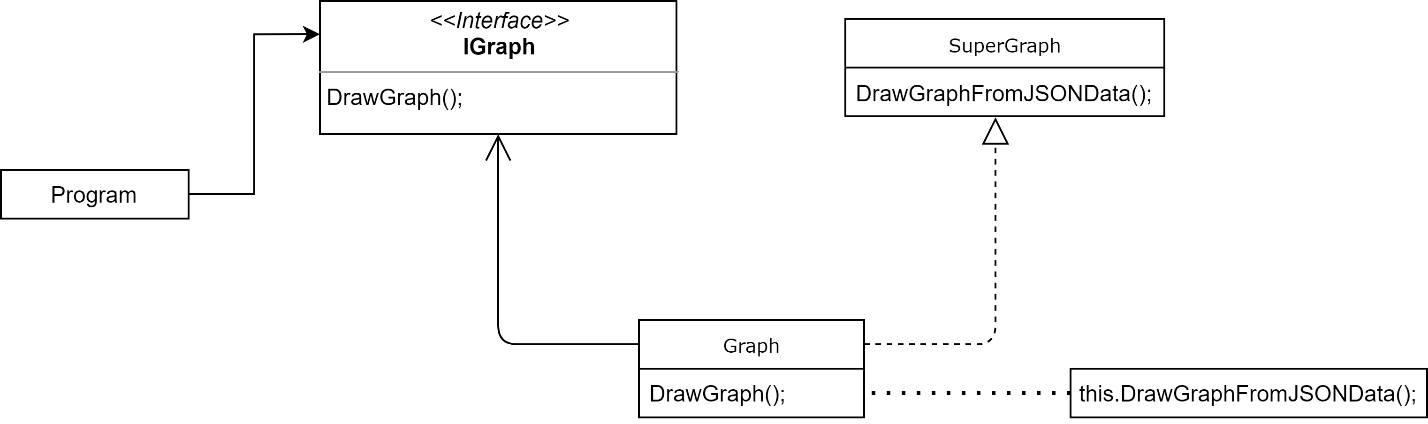


Рисунок 1 – Структура адаптера класса

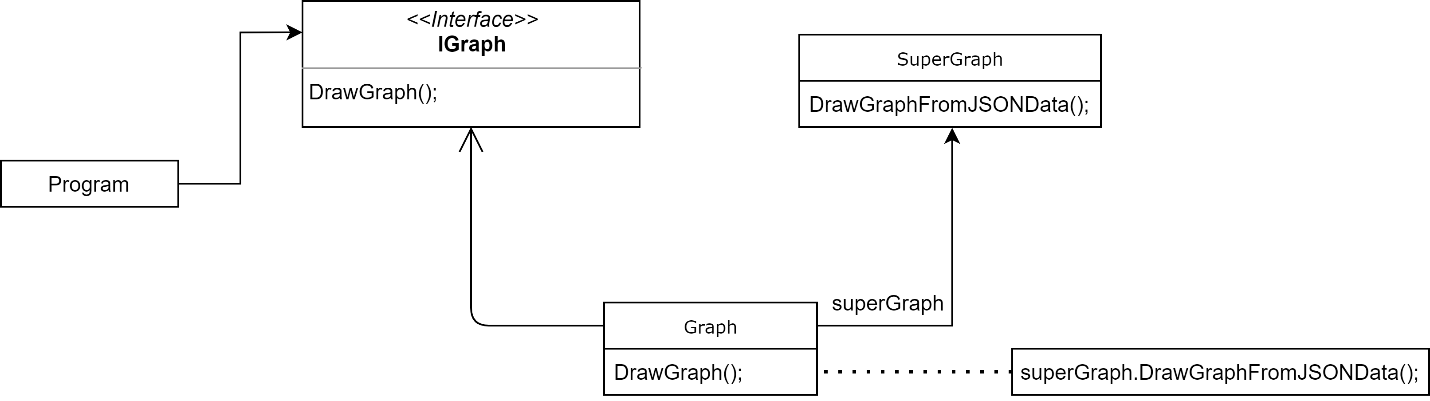


Рисунок 2 – Структура адаптера объекта

ВЫВОДЫ

В ходе выполнения лабораторной работы была исследована возможность использования структурных паттернов проектирования. Получены практические навыки применения структурных паттернов при объектно-ориентированном проектировании и рефакторинге ПО.

Был применен паттерн “Адаптер”, благодаря которому была интегрирована сторонняя библиотека без нарушения реализации интерфейса и модификации самой библиотеки. Был создан класс-адаптер, в котором производится конвертация входных XML данных в JSON формат, а затем вызывается метод адаптируемого класса.