Министерство образования Республики Беларусь

Учреждение образования «Белорусский государственный университет

информатики и радиоэлектроники»

Факультет компьютерных систем и сетей

Кафедра Информатики

Дисциплина «Программирование»

**ОТЧЕТ**

к лабораторной работе №9

на тему:

**«Множественное Наследование»**

БГУИР 6-05-0612-02 67

|  |
| --- |
| Выполнил студент группы 353503  КОХАН Артём Игоревич |
|  |
| (дата, подпись студента) |
| Проверил ассистент каф. Информатики  РОМАНЮК Максим Валерьевич |
|  |
| (дата, подпись преподавателя) |

Минск 2024

# 1 Индивидуальное задание

**Задание 1. Вариант 1.** Предметная область: Телевизор.

Для заданной предметной области реализовать следующие задачи:

– описать семейство классов, имеющих общий функционал (), при этом в каждом классе присутствует дополнительно свой функционал. Набор дополнительных функций в разных классах может быть произвольным;

– дополнительный функционал описать в виде набора интерфейсов;

– одна из общих функций должна быть реализована по-своему в каждом классе;

– одна из общих функций должна быть реализована в других классах (например, изменение скорости, использование оружия, доставка груза). При этом должно быть несколько вариантов реализации (несколько классов), например, персонажам игры доступны разные инструменты – каждый инструмент может использоваться разными персонажами. Конкретный вариант реализации выбирается при создании объекта (применить шаблон проектирования «Мост» («Bridge»));

– для создания объектов использовать шаблон проектирования «Абстрактная фабрика» или «Построитель» (Builder);

– в классе Program создать коллекцию разных объектов (см. п.1). Затем для каждого элемента коллекции вызвать все методы, доступные для данного объекта.

# 2 Выполнение работы

Перед выполнением работы следует разработать диаграмму классов для наглядного выполнения поставленной задачи (см. рисунок 1).



Рисунок 1 – Диаграмма классов

Стоит отметить, что это не вся диаграмма класса, отдельно на диаграмме выделим шаблон проектирования Builder (см. рисунок 2).

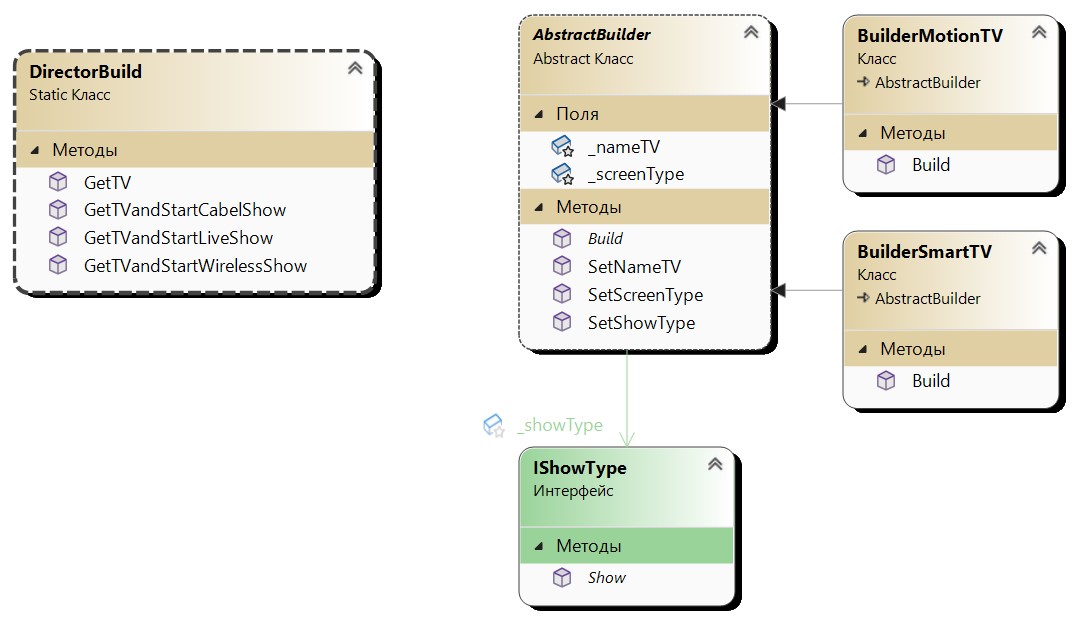


Рисунок 2 – Шаблон проектирования Builder на диаграммe классов

Для выполнения задания в проект была добавлена папка Classes и Interfaces для удобного взаимодействия с программой (cм. рисунок 3).

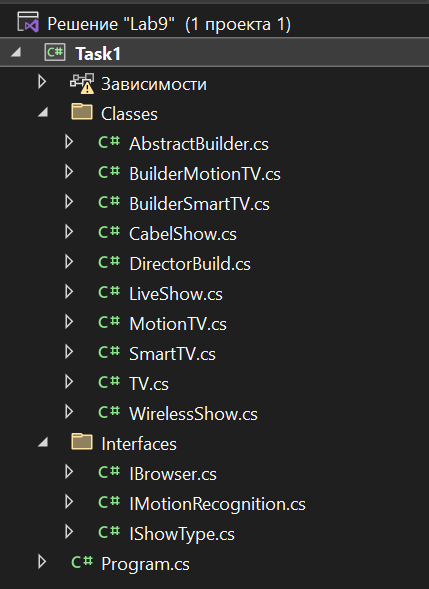


Рисунок 3 – Обозреватель решений

Рассмотрим реализацию интерфейса IShowType с методом Show, который говорит о том, что показывает телевизор, классы которые реализуют интерфейс мы рассмотрим позже.

namespace Task1.Interfaces

{

internal interface IShowType

{

string Show();

}

}

Класс CabelShow реализует показ кабельного телевидения, класс LiveShow реализует показ прямой трансляции, класс WirelessShow реализует показ переданного экрана.

namespace Task1.Classes

{

internal class CabelShow : IShowType

{

public string Show() =>

"Транслирует кабельное телевидение";

}

internal class LiveShow : IShowType

{

public string Show() =>

"Показывает прямую трансляцию";

}

internal class WirelessShow : IShowType

{

public string Show() =>

"Транслирует экран переданного устройста";

}

}

Рассмотрим реализацию абстрактного класса TV, тут имеются свойства и методы, которые будут реализованы в наследниках. Метод GetInfo будет выводить информацию об объекте, а метода Show говорит о том, что показывает экран телевизора.

namespace Task1.Classes

{

internal abstract class TV

{

public string? nameTV { get; set; }

public string? screenType { get; set; }

public IShowType? showType { get; set; } = null;

public abstract string GetInfo();

public abstract string Show();

}

}

Рассмотрим два интерфейса, которые потом будут реализованы в наследниках класса TV. Метод BrowseInternet будет показывать, то что телевизор взаимодействует с браузером, метод OutputRecognizeMotionOnScreen показывает, что телевизор выводит различные движения на экран.

namespace Task1.Interfaces

{

internal interface IBrowser

{

string BrowseInternet();

}

internal interface IMotionRecognition

{

string OutputRecognizeMotionOnScreen();

}

}

Рассмотрим одного из наследников класса TV, который будет реализовывать интерфейс IBrowser. В классе имеется дополнительный метод OpenYouTybe, который реализован, только в классе SmartTV. Следует отметить, что метод Show, может выводить и прочерк, это означает, что телевизор ничего не показывает.

namespace Task1.Classes

{

internal class SmartTV : TV, IBrowser

{

public SmartTV(string nameTV, string screenType)

{

this.nameTV = nameTV;

this.screenType = screenType;

}

public string OpenYouTybe() =>

"Открытие YouTybe...\nYouTybe успешно открыт";

public string BrowseInternet() =>

"Выполняется взаимодействие с браузером";

public override string GetInfo() =>

$"SmartTV, {nameTV}, {screenType}\nТелевизор подключен к интернету";

public override string Show() =>

(showType == null) ? "-" : showType.Show();

}

}

Рассмотрим ещё одного из наследников класса TV, который будет реализовывать интерфейс IMotionRecognition. В классе, также, имеется дополнительный метод, который реализован, только в классе MotionTV. Дополнительный метод реализован только в классе MotionTV, так как только в режиме, когда телевизор считывает ваши движения, можно играть в игру, которая напрямую связана с этим.

namespace Task1.Classes

{

internal class MotionTV : TV, IMotionRecognition

{

public MotionTV(string nameTV, string screenType)

{

this.nameTV = nameTV;

this.screenType = screenType;

}

public string PlayToTheRecognitionGame() =>

"Запущена игра, где нужно выполнять различные движения за время";

public string OutputRecognizeMotionOnScreen() =>

"Телевизор выводит ваши движения на экран";

public override string GetInfo() =>

$"MotionTV, {nameTV}, {screenType}\nТелевизор распознаёт движения";

public override string Show() =>

(showType == null) ? "-" : showType.Show();

}

}

Основная логика программы была уже описано, но ещё не был описан набор классов, которые реализует шаблон проектирования Builder. Рассмотрим абстрактный класс AbstractBuilder. Класс абстрактный, так как TV имеет наследников и нам нужно реализовывать метод Build в зависимости от выбранного наследника для создания объекта.

namespace Task1.Classes

{

internal abstract class AbstractBuilder

{

private protected string? \_nameTV;

private protected string? \_screenType;

private protected IShowType? \_showType = null;

public AbstractBuilder SetNameTV(string nameTV) {

\_nameTV = nameTV;

return this;

}

public AbstractBuilder SetScreenType(string screenType) {

\_screenType = screenType;

return this;

}

public AbstractBuilder SetShowType(IShowType showType) {

\_showType = showType;

return this;

}

public abstract TV Build();

}

}

Один из наследников класса TV – это SmartTV. Выполним построение объекта этого класса в наследнике AbstractBuilder. Всё также поле типа IShowType может принимать значение типа null.

namespace Task1.Classes

{

internal class BuilderSmartTV : AbstractBuilder

{

public override TV Build()

{

SmartTV smartTV = new SmartTV(nameTV: \_nameTV, screenType: \_screenType);

if (\_showType != null)

{

smartTV.showType = \_showType;

}

return smartTV;

}

}

}

Ещё один из наследников класса TV – это MotionTV. Тут такой же принцип построения, как и в классе BuilderSmartTV.

namespace Task1.Classes

{

internal class BuilderMotionTV : AbstractBuilder

{

public override TV Build()

{

MotionTV motionTV = new MotionTV(nameTV: \_nameTV, screenType: \_screenType);

if (\_showType != null)

{

motionTV.showType = \_showType;

}

return motionTV;

}

}

Для создания объектов в классе Program.cs мы будем обращаться к статическому классу DirectorBuilder, который создаёт различные объекты телевизоров. Также отмечу, что есть один статический метод GetTV, который создаёт объект, где объект типа IShowType имеет значение null.

namespace Task1.Classes

{

internal static class DirectorBuild

{

public static TV GetTVandStartCabelShow(string nameTV, string screenType, AbstractBuilder builder)

{

return builder.SetNameTV(nameTV)

.SetScreenType(screenType)

.SetShowType(new CabelShow())

.Build();

}

public static TV GetTVandStartLiveShow(string nameTV, string screenType, AbstractBuilder builder)

{

return builder.SetNameTV(nameTV)

.SetScreenType(screenType)

.SetShowType(new LiveShow())

.Build();

}

public static TV GetTVandStartWirelessShow(string nameTV, string screenType, AbstractBuilder builder)

{

return builder.SetNameTV(nameTV)

.SetScreenType(screenType)

.SetShowType(new WirelessShow())

.Build();

}

public static TV GetTV(string nameTV, string screenType, AbstractBuilder builder)

{

return builder.SetNameTV(nameTV)

.SetScreenType(screenType)

.Build();

}

}

}

Результат работы классов описан в классе Program, чтобы наглядно разобраться в структуре созданных классов и интерфейсов.

namespace Task1

{

file class Program

{

static void Main(string[] args)

{

List<TV> listOfTV = new List<TV>();

listOfTV.AddRange(new TV[]

{

DirectorBuild.GetTVandStartCabelShow("LG", "AMOLED", new BuilderSmartTV()),

DirectorBuild.GetTVandStartLiveShow("Samsung", "OLED", new BuilderSmartTV()),

DirectorBuild.GetTVandStartWirelessShow("Horizont", "IPS", new BuilderMotionTV()),

DirectorBuild.GetTV("Витязь", "ЖК", new BuilderSmartTV())

}) ;

foreach (var tv in listOfTV)

{

if (tv is SmartTV smartTV)

{

Console.WriteLine($"Get info: {smartTV.GetInfo()}");

Console.WriteLine($"Show: {smartTV.Show()}");

Console.WriteLine($"Browse internet: {smartTV.BrowseInternet()}");

Console.WriteLine($"Open YouTybe: {smartTV.OpenYouTybe()}\n");

}

else if (tv is MotionTV motionTV)

{

Console.WriteLine($"Get info: {motionTV.GetInfo()}");

Console.WriteLine($"Show: {motionTV.Show()}");

Console.WriteLine($"Output recognize motion on screen: {motionTV.OutputRecognizeMotionOnScreen()}");

Console.WriteLine($"Play to the recognition game: {motionTV.PlayToTheRecognitionGame()}\n");

}

}

}

}

}

Результат работы программы продемонстрирован ниже (см. рисунок 4).

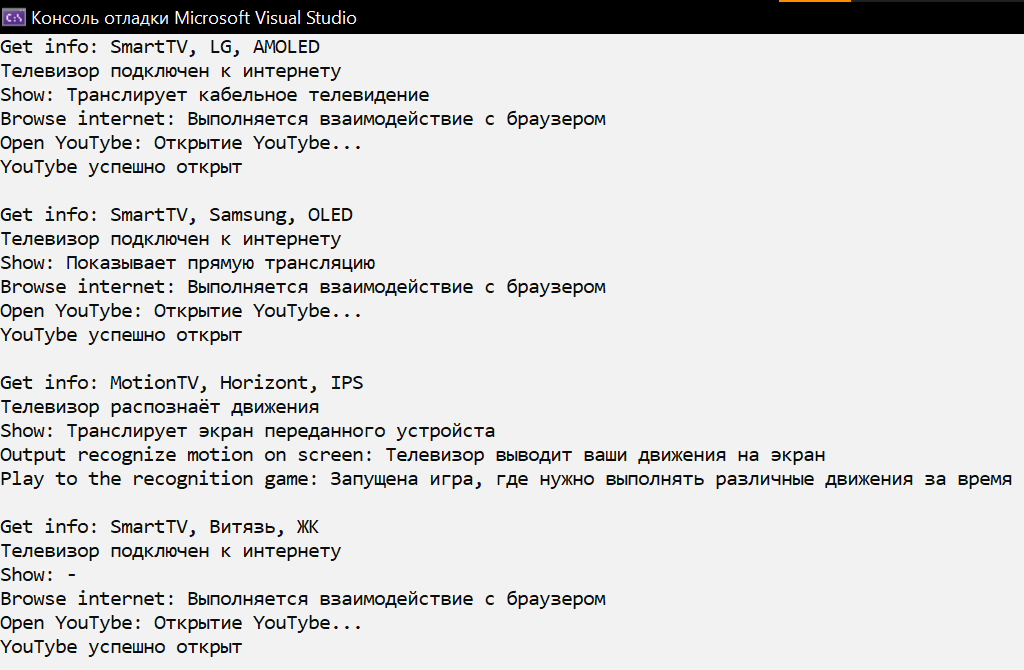


Рисунок 4 – Результат работы программы

# Вывод

В ходе лабораторной работы были получены навыки проектирования приложения, состоящих из нескольких взаимосвязанных классов и интерфейсов. Создания объектов при помощи шаблона проектирования Builder. Освоены основные навыки разделения абстракции от реализации при помощи шаблона проектирования Bringe.