Часть 1. Наследование

Цель работы: изучить наследование в языке C#.

Выполнение: диаграмма классов изображена на рисунке 1, объектная декомпозиция изображена на рисунке 2

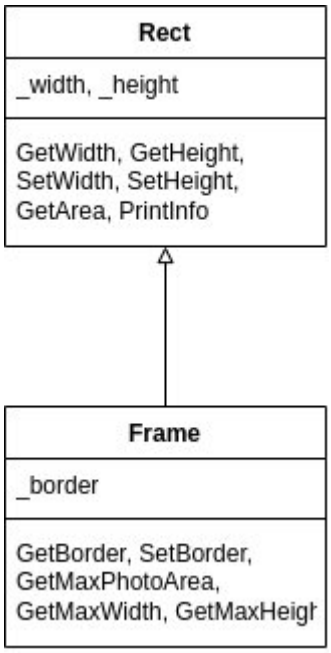

Рисунок 1 — Диаграмма классов

Рисунок 2 — Объектная декомпозиция

Код программы изображён на рисунках 3-4.

```
namespace _01;

2 usages  1 inheritor
public class Rect
{
    private int _width;
    private int _height;
    2 usages
    public Rect(int width, int height)
    {
        SetWidth(width);
        SetHeight(height);
    }
    5 usages
    public int GetWidth(){...}
    5 usages
    public int GetHeight(){...}
    3 usages
    public void SetWidth(int width)
    {
        _width = int.Max(0, width);
    }
    3 usages
    public void SetHeight(int height)
    {
        _height = int.Max(0, height);
    }
    4 usages
    public int GetArea()
    {
        return _width * _height;
    }
    3 usages
    public void PrintInfo()
    {
        Console.WriteLine($"Rect {_width}x{_height}, area={GetArea()}");
    }
}
```

Рисунок 3 — Класс Rect

```
namespace _01;

1 usage
public class Frame : Rect
{
    private int _border;
    1 usage
    public Frame(int width, int height, int border) : base(width, height)
    {
        SetBorder(border);
    }
    1 usage
    public int GetBorder(){...}
    2 usages
    public void SetBorder(int border)
    {
        _border = int.Max(0, border);
    }
    2 usages
    public int GetMaxPhotoArea()
    {
        return (GetWidth() + _border) * (GetHeight() + _border);
    }
    1 usage
    public int GetMaxWidth()
    {
        return GetWidth() + _border;
    }
    1 usage
    public int GetMaxHeight()
    {
        return GetHeight() + _border;
    }
    3 usages
    public new void PrintInfo()
    {
        base.PrintInfo();
        Console.WriteLine($"Max photo area: {GetMaxPhotoArea()}");
    }
}
```

Рисунок 4 — Класс Frame

Тестирование программы: тестирующая программа изображена на рисунке 5. Результат работы программы изображён на рисунке 6.

```
namespace _01;

internal static class Program
{
    1 usage
    private static void TestRect()
    {
        Console.WriteLine("=====Checking rectangle=====");
        var rect = new Rect( width: 640,  height: 480);
        rect.PrintInfo();
        Console.WriteLine($"Area: {rect.GetWidth() * rect.GetHeight()}");
        Console.WriteLine($"Area: {rect.GetArea()}");
        Console.WriteLine("width * 4 && height * 2");
        rect.SetWidth(rect.GetWidth() * 4);
        rect.SetHeight(rect.GetHeight() * 2);
        rect.PrintInfo();
        Console.WriteLine($"New area: {rect.GetArea()}");
    }

    1 usage
    private static void TestFrame()
    {
        Console.WriteLine("=====Checking frame=====");
        var frame = new Frame( width: 20,  height: 5,  border: 2);
        frame.PrintInfo();
        Console.WriteLine($"Area: {frame.GetArea()}");
        Console.WriteLine($"Max photo size: {frame.GetMaxWidth()}x{frame.GetMaxHeight()}");
        Console.WriteLine($"Max photo area: {frame.GetMaxPhotoArea()}");
        Console.WriteLine("width * 2 && height * 2");
        frame.SetWidth(frame.GetWidth() * 2);
        frame.SetHeight(frame.GetHeight() * 2);
        frame.PrintInfo();
        Console.WriteLine("border / 2");
        frame.SetBorder(frame.GetBorder() / 2);
        frame.PrintInfo();
    }

    public static void Main()
    {
        TestRect();
        TestFrame();
    }
}
```

Рисунок 5 — Тестирующая программа

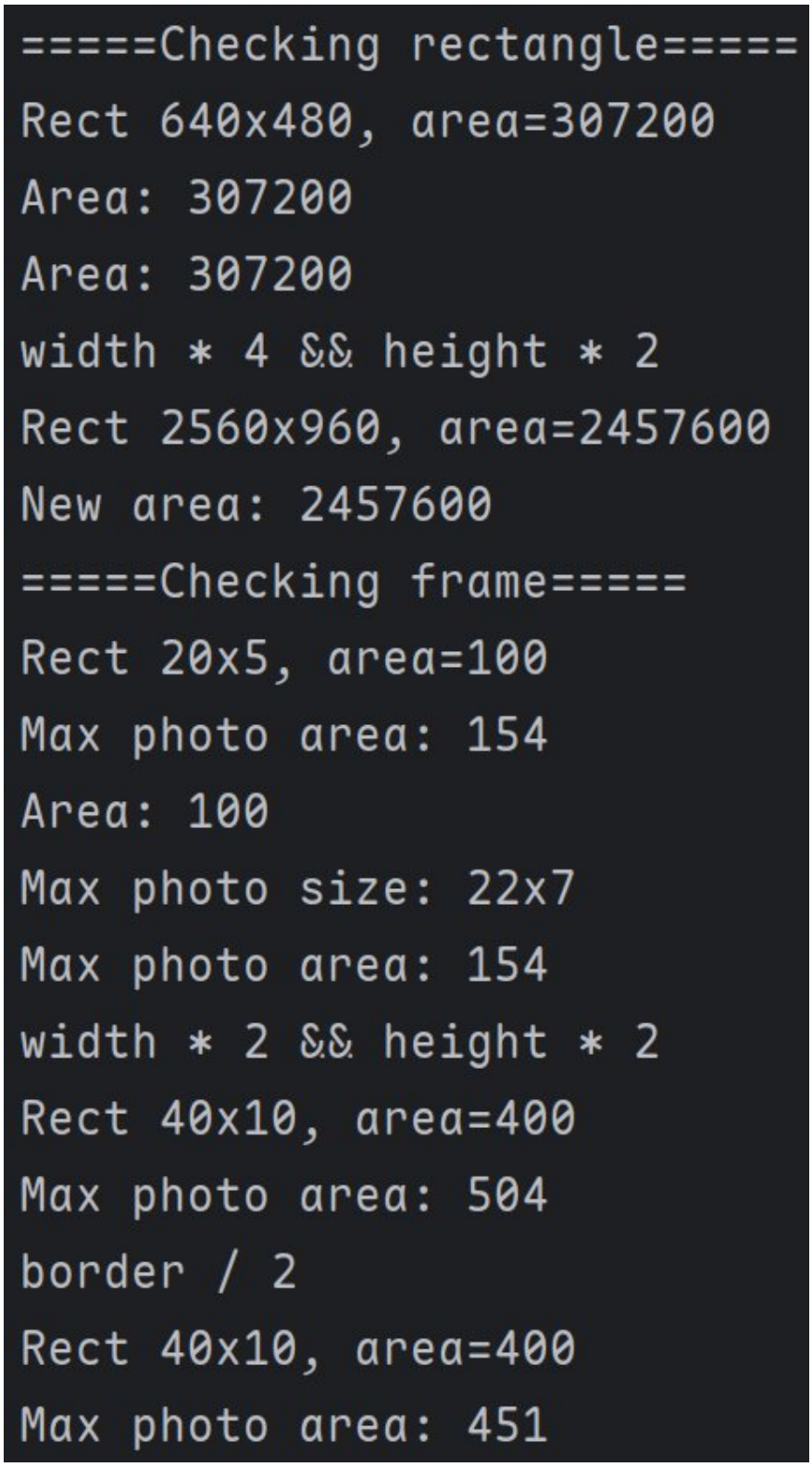

Рисунок 6 — Результат работы программы

Вывод: был изучен механизм “насследование” в языке C#

Часть 2. Композиция.

Цель работы: изучить наследование в C#

Выполнение: диаграмма классов изображена на рисунке 7. Объектная декомпозиция изображена на рисунке 8.

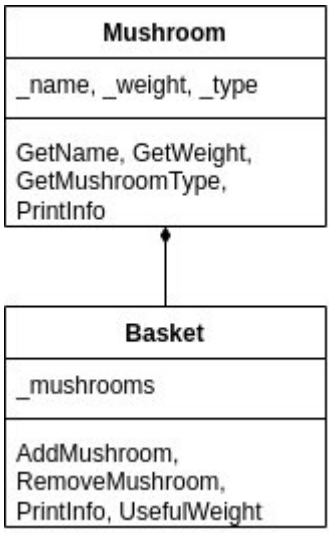

Рисунок 7 — Диаграмма классов

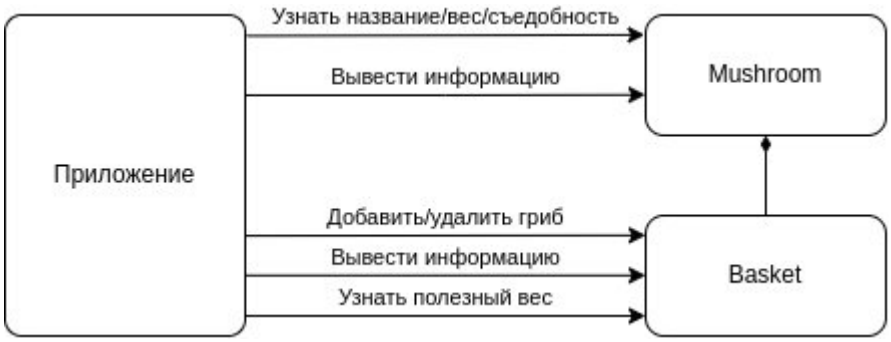

Рисунок 8 — Объектная декомпозиция

Основной код программы изображен на рисунках 9 - 11

```
public enum MushroomType
{
    Edible, Inedible, ConditionallyEdible
}
```

Рисунок 9 — Виды грибов

```
public class Mushroom
{
    private string _name;
    private int _weight;
    private MushroomType _type;
    3 usages
    public Mushroom(string name, int weight, MushroomType type)
    {
        _name = name;
        _weight = int.Max(0, weight);
        _type = type;
    }
    1 usage
    public string GetName(){...}
    1 usage
    public int GetWeight(){...}
    1 usage
    public MushroomType GetMushroomType(){...}
    3 usages
    public void PrintInfo()
    {
        Console.WriteLine($"Гриб {_name}, вес: {_weight}");
        switch (_type)
        {
            case MushroomType.Edible:
                Console.WriteLine("Съедобный");
                break;
            case MushroomType.Inedible:
                Console.WriteLine("Несъедобный");
                break;
            case MushroomType.ConditionallyEdible:
                Console.WriteLine("Условно съедобный");
                break;
        }
    }
}
```

Рисунок 10 — Код класса Mushroom

```
public class Basket
{
    private ArrayList _mushrooms = new();
    1 usage
    public Basket(Mushroom[] mushrooms)
    {
        _mushrooms.AddRange(mushrooms);
    }
    1 usage
    public void AddMushroom(Mushroom mushroom)
    {
        _mushrooms.Add(mushroom);
    }
    1 usage
    public void RemoveMushroom(int index)
    {
        if (index ≥ 0 && index < _mushrooms.Count)
            _mushrooms.RemoveAt(index);
    }
    3 usages
    public void PrintInfo()
    {
        foreach (Mushroom mushroom in _mushrooms)
        {
            mushroom.PrintInfo();
        }
    }
    3 usages
    public int UsefulWeight()
    {
        var result = 0;
        foreach (Mushroom mushroom in _mushrooms)
        {
            if (mushroom.GetMushroomType() ≠ MushroomType.Inedible)
                result += mushroom.GetWeight();
        }
        return result;
    }
}
```

Рисунок 11 — Код класса Basket

Тестирование программы: код тестирующей программы изображён на рисунке 12. Результат работы изображен на рисунке 13.

```
internal static class Program
{
    private static void Main()
    {
        Console.WriteLine("======Грибы======");
        var mushroom1 = new Mushroom( name: "Белый гриб", weight: 100, MushroomType.Edible);
        var mushroom2 = new Mushroom( name: "Поганка", weight: 60, MushroomType.Inedible);
        mushroom1.PrintInfo();
        mushroom2.PrintInfo();

        Console.WriteLine("======Корзинка======");
        var basket = new Basket( mushrooms: [mushroom1, mushroom2]);
        basket.PrintInfo();
        Console.WriteLine($"Полезный вес: {basket.UsefulWeight()}");
        Console.WriteLine("Удаление белого гриба...");
        basket.RemoveMushroom( index: 0);
        basket.PrintInfo();
        Console.WriteLine($"Полезный вес: {basket.UsefulWeight()}");
        var newMushroom = new Mushroom( name: "Подберёзовик", weight: 30, MushroomType.Edible);
        Console.WriteLine($"Новый гриб: {newMushroom.GetName()}");
        basket.AddMushroom(newMushroom);
        basket.PrintInfo();
        Console.WriteLine($"Полезный вес: {basket.UsefulWeight()}");
    }
}
```

Рисунок 12 — Тестирующая программа

```
======Грибы======
Гриб Белый гриб, вес: 100
Съедобный
Гриб Поганка, вес: 60
Несъедобный
======Корзинка======
Гриб Белый гриб, вес: 100
Съедобный
Гриб Поганка, вес: 60
Несъедобный
Полезный вес: 100
Удаление белого гриба...
Гриб Поганка, вес: 60
Несъедобный
Полезный вес: 0
Новый гриб: Подберёзовик
Гриб Поганка, вес: 60
Несъедобный
Гриб Подберёзовик, вес: 30
Съедобный
Полезный вес: 30
```

Рисунок 13 — Результат работы программы

Вывод: был изучен механизм “композиция” в языке C#