

Міністерство освіти і науки України  
Національний технічний університет України  
Київський Політехнічний Інститут імені Ігоря Сікорського  
Кафедра ОТ

Лабораторна робота № 7 з дисципліни "Основи Web-програмування"

Виконав: студент 2-го курсу

Група: ІП-64 ФІОТ

Вінницький В'ячеслав

Заліковка: №6402

**Київ-2018**

## Постановка задачі до комп'ютерного практикуму № 7

При виконанні комп'ютерного практикуму слід реалізувати наступні задачі:

- a) Перезавантажити віртуальний метод `bool Equals (object obj)`, таким чином, щоб об'єкти були рівними, якщо рівні всі дані об'єктів. Для кожного з класів самостійно визначити, які атрибути використовуються для порівняння;
- b) Визначити операції `==` та `!=`. При цьому врахувати, що визначення операцій повинно бути погоджено з перезавантаженим методом `Equals`, тобто критерії, за якими перевіряється рівність об'єктів в методі `Equals`, повинні використовуватися і при перевірці рівності об'єктів в операціях `==` та `!=`;
- c) Перевизначити віртуальний метод `int GetHashCode()`. Класи базової бібліотеки, що викликають метод `GetHashCode()` з призначеного користувальницького типу, припускають, що рівним об'єктів відповідають рівні значення хеш-кодів. Тому в разі, коли під рівністю об'єктів розуміється збіг даних (а не посилань), реалізація методу `GetHashCode()` повинна для об'єктів з однаковими даними повертати рівні значення хеш-кодів.
- d) Визначити метод `object DeepCopy()` для створення повної копії об'єкта. Визначені в деяких класах базової бібліотеки методи `Clone()` та `Copy()` створюють обмежену (shallow) копію об'єкта - при копіюванні об'єкта копії створюються тільки для полів структурних типів, для полів, на які посилаються типи, копіюються тільки посилання. В результаті в обмеженій копії об'єкта поля-посилання вказують на ті ж об'єкти, що і в вихідному об'єкті. Метод `DeepCopy()` повинен створити повні копії всіх об'єктів, посилання на які містять поля типу. Після створення повна копія не залежить від вихідного об'єкта - зміна будь-якого поля або властивості вихідного об'єкта не повинно призводити до зміни копії. При реалізації методу `DeepCopy()` в класі, який має поле типу `System.Collections.ArrayList`, слід мати на увазі, що визначені в класі `ArrayList` конструктор `ArrayList (ICollection)` і метод `Clone()` при створенні копії колекції, що складається з елементів, на які посилаються типи, копіюють тільки посилання. Метод `DeepCopy ()` повинен створити як копії елементів колекції `ArrayList`, так і повні копії об'єктів, на які посилаються елементи колекції. Для типів, що містять колекції, реалізація методу `DeepCopy()` спрощується, якщо в типах елементів колекцій також визначити метод `DeepCopy ()`.
- e) Перезавантажити віртуальний метод `string ToString()` для формування строки з інформацією про всі елементи списку
- f) Підготувати демонстраційний приклад, в котрому будуть використані всі розроблені методи
- g) Підготувати звіт з результатами виконаної роботи.

При виконанні комп'ютерного практикуму слід реалізувати наступні задачі:

- a) Визначити клас, котрий містить типізовану колекцію та котрий за допомогою подій інформує про зміни в колекції.  
Колекція складається з об'єктів силових типів. Колекція змінюється при видаленні/додаванні елементів або при зміні одного з вхідних в колекцію посилань, наприклад, коли одному з посилань присвоюється нове значення. В цьому випадку у відповідних методах або властивості класу кидаються події.  
При зміні даних об'єктів, посилання на які входять в колекцію, значення самих посилань не змінюються. Цей тип змін не породжує подій.  
Для подій, що сповіщають про зміни в колекції, визначається свій делегат. Події реєструються в спеціальних класах-слухачах.
- b) Реалізувати обробку помилок, при цьому необхідно перевизначити за допомогою наслідування наступні події:
- 1) StackOverflowException
  - 2) ArrayTypeMismatchException
  - 3) DivideByZeroException
  - 4) IndexOutOfRangeException
  - 5) InvalidCastException
  - 6) OutOfMemoryException
  - 7) OverflowException
- c) Підготувати демонстраційний приклад, в котрому будуть використані всі розроблені методи
- d) Підготувати звіт з результатами виконаної роботи.

### **Варіант індивідуального завдання:**

#### **Вариант 2**

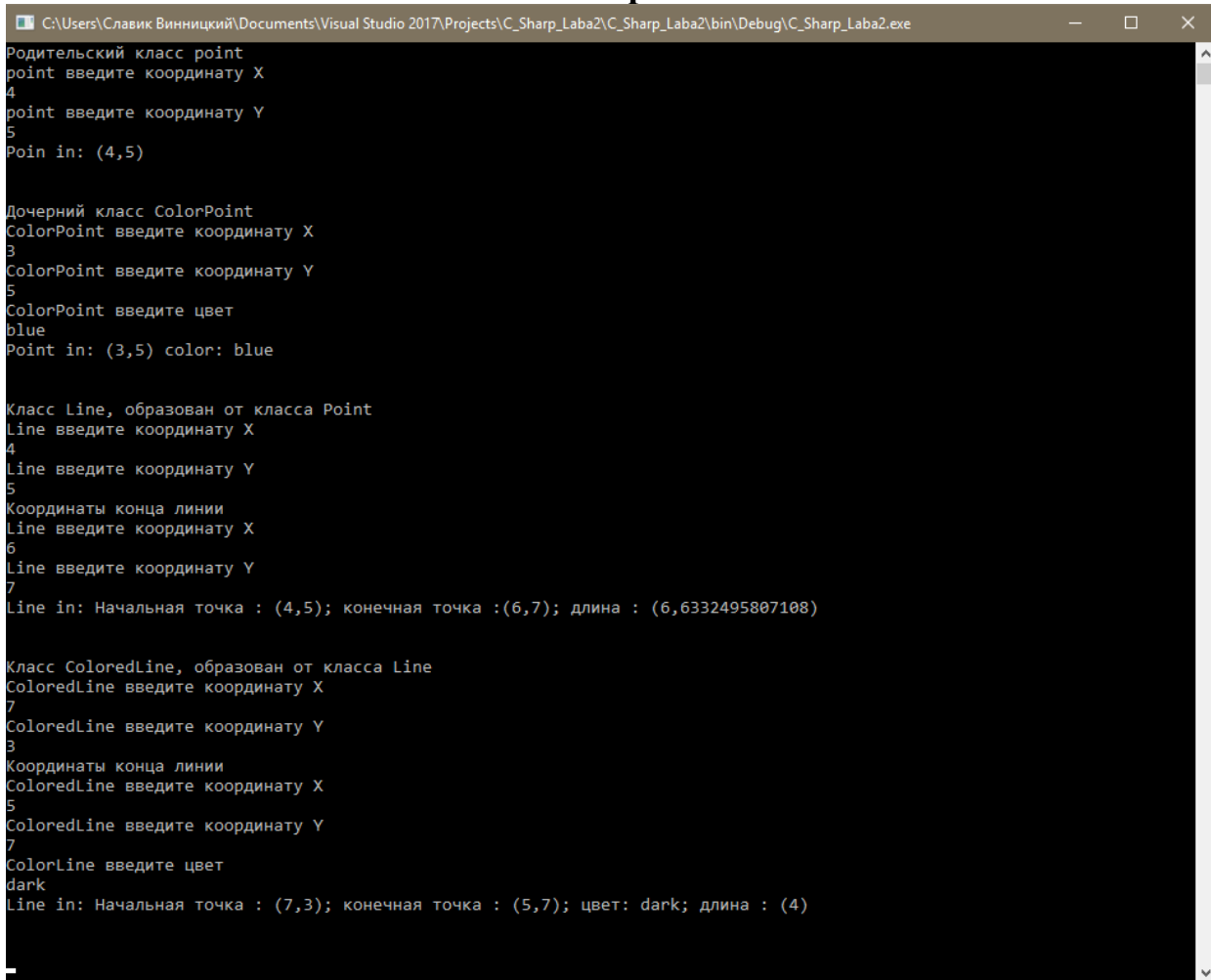
Создать абстрактный класс Point (точка). На его основе создать классы ColoredPoint и Line. На основе класса Line создать класс ColoredLine и класс PolyLine (многоугольник).

Все классы должны иметь виртуальные методы установки и получения значений всех координат, а также изменения цвета и получения текущего цвета.

Создать класс Picture, содержащий массив/параметризованную коллекцию объектов этих классов в динамической памяти.

Предусмотреть возможность вывода характеристик объектов списка.

## Скріншоти :



```
C:\Users\Славик Винницкий\Documents\Visual Studio 2017\Projects\C_Sharp_Laba2\C_Sharp_Laba2\bin\Debug\C_Sharp_Laba2.exe
Родительский класс point
point введите координату X
4
point введите координату Y
5
Poin in: (4,5)

Дочерний класс ColorPoint
ColorPoint введите координату X
3
ColorPoint введите координату Y
5
ColorPoint введите цвет
blue
Point in: (3,5) color: blue

Класс Line, образован от класса Point
Line введите координату X
4
Line введите координату Y
5
Координаты конца линии
Line введите координату X
6
Line введите координату Y
7
Line in: Начальная точка : (4,5); конечная точка :(6,7); длина : (6,6332495807108)

Класс ColoredLine, образован от класса Line
ColoredLine введите координату X
7
ColoredLine введите координату Y
3
Координаты конца линии
ColoredLine введите координату X
5
ColoredLine введите координату Y
7
ColorLine введите цвет
dark
Line in: Начальная точка : (7,3); конечная точка : (5,7); цвет: dark; длина : (4)
```

## Тексти програмного коду :

```
using System;

namespace C_Sharp_Laba2
{
    class Point
    {
        protected int Xpos;
        protected int Ypos;
        public Point(int x, int y)
        {
            Xpos = x;
            Ypos = y;
        }
        public virtual void Draw()
        {
            Console.WriteLine("Poin in: ({0},{1})", Xpos, Ypos);
        }
    }
    class ColorPoint : Point
```

```

{
    string clr;
    public ColorPoint(int x, int y, string color)
        : base(x, y)
    {
        clr = color;
    }
    public override void Draw()
    {
        Console.WriteLine("Point in: ({0},{1}) color: {2}", Xpos, Ypos, clr);
    }
}
class Line : Point
{
    protected int Xo;
    protected int Yo;
    public Line(int x, int y, int a, int u)
        : base(x, y)
    {
        Xo = a;
        Yo = u;
    }
    public override void Draw()
    {
        Console.WriteLine("Line in: Начальная точка : ({0},{1}); конечная точка : ({2},{3}); длина : ({4}) ", Xpos, Ypos, Xo, Yo, Math.Sqrt(Math.Abs(Math.Pow(Xpos, 2) - Math.Pow(Xo, 2) + Math.Pow(Ypos, 2) - Math.Pow(Yo, 2))));
    }
}
class ColoredLine : Line
{
    string clr;

    public ColoredLine(int x, int y, int a, int u, string color)
        : base(x, y, a, u)
    {
        Xo = a;
        Yo = u;
        clr = color;
    }
    public override void Draw()
    {
        Console.WriteLine("Line in: Начальная точка : ({0},{1}); конечная точка : ({2},{3}); цвет: {4}; длина : ({5}) ", Xpos, Ypos, Xo, Yo, clr, Math.Sqrt(Math.Abs(Math.Pow(Xpos, 2) - Math.Pow(Xo, 2) + Math.Pow(Ypos, 2) - Math.Pow(Yo, 2))));
    }
}
class Program
{
    static void Main(string[] args)
    {
        Console.WriteLine("Родительский класс point");
        Console.WriteLine("point введите координату X ");
        int k = Convert.ToInt16(Console.ReadLine());
        Console.WriteLine("point введите координату Y ");
        int l = Convert.ToInt16(Console.ReadLine());
        Point onepoint = new Point(k, l);
        onepoint.Draw();
        Console.WriteLine();
        Console.WriteLine();
    }
}

```

```

Console.WriteLine("Дочерний класс ColorPoint");
Console.WriteLine("ColorPoint введите координату X ");
k = Convert.ToInt16(Console.ReadLine());
Console.WriteLine("ColorPoint введите координату Y ");
l = Convert.ToInt16(Console.ReadLine());
Console.WriteLine("ColorPoint введите цвет");
string s = Console.ReadLine();
Point pt = new ColorPoint(k, l, s);
pt.Draw();
Console.WriteLine();
Console.WriteLine();
Console.WriteLine("Класс Line, образован от класса Point ");
Console.WriteLine("Line введите координату X ");
k = Convert.ToInt16(Console.ReadLine());
Console.WriteLine("Line введите координату Y ");
l = Convert.ToInt16(Console.ReadLine());
Console.WriteLine("Координаты конца линии ");
Console.WriteLine("Line введите координату X ");
int m = Convert.ToInt16(Console.ReadLine());
Console.WriteLine("Line введите координату Y ");
int z = Convert.ToInt16(Console.ReadLine());
Point lin = new Line(k, l, m, z);
lin.Draw();
Console.WriteLine();
Console.WriteLine();
Console.WriteLine("Класс ColoredLine, образован от класса Line ");
Console.WriteLine("ColoredLine введите координату X ");
try
{
    k = Convert.ToInt16(Console.ReadLine());
}
catch(MyInvalidCastException e)
{
    Console.WriteLine(e.Message);
}
Console.WriteLine("ColoredLine введите координату Y ");
try
{
    l = Convert.ToInt16(Console.ReadLine());
}
catch (MyInvalidCastException e)
{
    Console.WriteLine(e.Message);
}
Console.WriteLine("Координаты конца линии ");
Console.WriteLine("ColoredLine введите координату X ");
try
{
    m = Convert.ToInt16(Console.ReadLine());
}
catch (MyInvalidCastException e)
{
    Console.WriteLine(e.Message);
}
Console.WriteLine("ColoredLine введите координату Y ");
try
{

```

```

        z = Convert.ToInt16(Console.ReadLine());
    }
    catch (MyInvalidCastException e)
    {
        Console.WriteLine(e.Message);
    }
    Console.WriteLine("ColorLine введите цвет");
    s = Console.ReadLine();
    Point cln = new ColoredLine(k, l, m, z, s);
    cln.Draw();
    Console.WriteLine();
    Console.WriteLine();
    var picture = new Picture<Line>(4);
    Console.ReadLine();
}

class MyInvalidCastException : InvalidCastException
{
    public MyInvalidCastException()
        : base() { }

    public MyInvalidCastException(string message)
        : base(message) { }

    public MyInvalidCastException(string format, params object[] args)
        : base(string.Format(format, args)) { }

    public MyInvalidCastException(string message, Exception innerException)
        : base(message, innerException) { }

    public MyInvalidCastException(string format, Exception innerException, params
object[] args)
        : base(string.Format(format, args), innerException) { }
}

public class Picture<T>
    where T : class, IComparable<T>
{
    private static void Show_Message(string message)
    {
        Console.WriteLine(message);
    }

    public delegate void AddedNewValue(string message);

```

```

public event AddedNewValue Added;

private T[] array;

public Picture(int size)
{
    this.Added += Show_Message;
    array = new T[size];
}

public int Length
{
    get { return array.Length; }
}

public void OrderBy(bool desc = false)
{
    if (desc == true)
        Sort((x, y) => x.CompareTo(y) < 0);

    else
        Sort((x, y) => x.CompareTo(y) > 0);
}

protected void Sort(Func<T, T, bool> func)
{
    if (Length == 0)

        throw new IndexOutOfRangeException();

    for (var i = 0; i < Length - 1; i++)

        for (var j = i + 1; j < Length; j++)

            if (func(array[i], array[j]))
            {
                var temp = array[i];
                array[i] = array[j];
                array[j] = temp;
            }
}

public T this[int index]
{
    get
    {
        if (index >= Length)    throw new IndexOutOfRangeException();

        return array[index];
    }

    set
    {
        if (Added != null)
        {
            if (index >= Length)

                throw new IndexOutOfRangeException();

```



```

        Added($"Added value {value} with index {index}");
    }
    array[index] = value;
}
}

public override string ToString()
{
    string res = "";

    for (var i = 0; i < Length; i++)
        res = string.Concat(res, string.Concat(array[i] + "\n"));

    return res;
}
}
}

```