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
Халеп Тарас
Заліковка: №6427

Київ-2018

Постановка задачі до комп'ютерного практикуму № 7

При виконанні комп'ютерого 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 типів, копіюють тільки посилання. Метод DeepCop () 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'ютерого практикуму слід реалізувати наступні задачі:

a) Визначити клас, котрий містить типізовану колекцію та котрий за допомогою подій інформує ппо зміни в колекції.

Колекція складається з об'єктів силочних 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.
e)

Варіанти індивідуальних завдань:

Варіант 27%25 = 2
Вариант 2

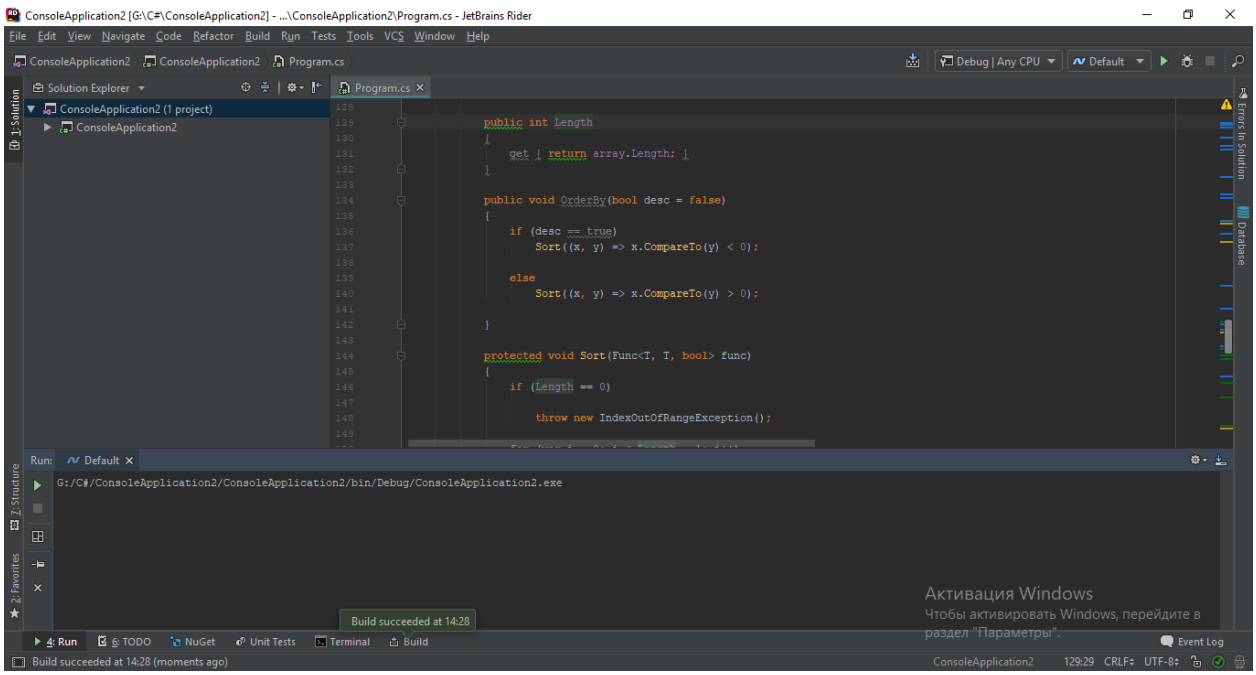
Создать абстрактный класс Point (точка). На его основе создать классы ColoredPoint и Line. На основе класса Line создать класс ColoredLine и класс PolyLine (многоугольник).

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 :



Тексти програмного коду :

```
using System;

namespace ConsoleApplication2
{
    class Point
        {
            protected int Xpos;
            protected int Ypos;
```

```
            public Point(int xpos, int ypos)
            {
                Xpos = xpos;
                Ypos = ypos;
            }

            public virtual void Draw()
            {
                Console.WriteLine("Poin in: ({0},{1})", Xpos, Ypos);
            }
        }
        class ColorPoint : Point
        {
            string clr;

            public ColorPoint(int xpos, int ypos, string clr) : base(xpos,
ypos)
            {
                this.clr = clr;
            }

            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 xpos, int ypos, int xo, int yo) : base(xpos,
ypos)
            {
                Xo = xo;
                Yo = yo;
            }

            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)
            {

            }
        }

        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)
            {
                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;

            }
    }
}
```