**Лабораторная работа 9**

Титов Павел Сергеевич, 04-1

**Базовый уровень:**

1) Реализовать класс «Точка», содержащий в качестве полей название точки (А, В, С и т.д.) и значение координат точки (X, Y).

В классе предусмотреть:

* *конструкторы по умолчанию и конструкторы с параметрами;*
* *свойства.*

2. Реализовать класс геометрическая фигура на плоскости (геометрическая фигура выбирается согласно варианту).

В классе предусмотреть:

* *поле – массив класса «Точка»;*
* *индексаторы, свойства;*
* *конструкторы;*
* *метод вычисления площади фигуры;*
* *событие (площадь фигуры равна 1). Обработчик события должен выводить старое и новое значение площади на экран (старое значение площади передавать в качестве параметра события);*
* *Предусмотреть возможность обработки исключений при вводе данных (стандартные виды исключений).*

**Повышенный уровень:**

В дополнение к базовому уровню реализовать свой класс исключение, основанный на классе Exception, которое возникает при невозможности построения данной фигуры исходя из введенных координат вершин фигуры. Реализовать реакцию на данное исключение на основе события и обработчик этого события. В качестве параметров передавать объект класса «Точка» вызвавший исключение.

Point.cs

namespace L9;  
  
public class Point  
{  
 private string \_nextName = "A";  
 private int \_nextNameNum = 0;  
  
 public double X { get; set; }  
 public double Y { get; set; }  
 public string Name { get; set; }  
  
 public Point(double x, double y)  
 {  
 X = x;  
 Y = y;  
  
 Name = \_nextName;  
 if (\_nextName.Length == 1 && \_nextName == "Z" || \_nextName.Length != 1 && \_nextName[0] == 'Z')  
 {  
 \_nextNameNum += 1;  
 \_nextName = "A" + \_nextNameNum;  
 }  
 else if (\_nextName.Length != 1 && \_nextName != "Z")  
 \_nextName = ((char)(\_nextName[0] + 1)).ToString() + \_nextNameNum;  
 else  
 \_nextName = ((char)(\_nextName[0] + 1)).ToString();  
 }  
  
 public override string ToString()  
 {  
 return $"{Name}: x={X} y={Y}";  
 }  
}

Square.cs

namespace L9;  
  
public delegate void EventHandler(double value);  
  
public class Square  
{  
 private Point[] \_points;  
 event EventHandler AreaEvent;  
  
 void AreaEqualsToOne(double oldArea)  
 {  
 Console.*WriteLine*("Старая площадь равна {0}", oldArea);  
 Console.*WriteLine*("Новая площадь равна 1");  
 }  
  
 public Point[] Points  
 {  
 get => \_points;  
 set => \_points = value;  
 }  
  
 public double Area  
 {  
 get  
 {  
 double ab = VectorLenght(Vector(Points[0], Points[1]));  
 double bc = VectorLenght(Vector(Points[1], Points[2]));  
 double ca = VectorLenght(Vector(Points[2], Points[0]));  
 double p = (ab + bc + ca) / 2;  
 return Math.*Abs*(p \* (p - ab) \* (p - bc) \* (p - ca));  
 }  
 }  
  
 public Point this[int index]  
 {  
 get  
 {  
 if (index < 0 || index >= \_points.Length)  
 throw new IndexOutOfRangeException("Такой точки нет. Вы вышли за границы массива точек.");  
 return \_points[index];  
 }  
 set  
 {  
 double area = 0;  
  
 if (\_points.Length >= 3)  
 area = Area;  
  
 Point[] points = \_points;  
 points[index] = value;  
  
 if (area != 0 && Math.*Abs*(Area - 1) < 0.00001)  
 AreaEvent?.Invoke(area);  
 }  
 }  
  
 public Square()  
 {  
 Point[] points = { new(0, 0), new(0, 1), new(1, 1), new(1, 0) };  
 AreaEvent += AreaEqualsToOne;  
 Points = points;  
 }  
  
 public Square(Point[] points)  
 **{**  
if (points[0].X == points[1].X && points[0].Y == points[1].Y) throw new InvalidInputData(points[1]);  
 if (!IsValidPolygon(points)) throw new InvalidInputData(points[2]);  
 AreaEvent += AreaEqualsToOne;  
 Points = points;  
 **}**  
  
public bool IsValidPolygon(Point[] points)  
 {  
 double ab = VectorLenght(Vector(points[0], points[1]));  
 double bc = VectorLenght(Vector(points[1], points[2]));  
 double ca = VectorLenght(Vector(points[2], points[0]));  
 if (ab \* bc \* ca < 0.01) return false;  
 if (ab + bc < ca) return false;  
 if (ca + bc < ab) return false;  
 if (ca + ab < bc) return false;  
 return true;  
 }  
  
 private double SideTurn(Point ab, Point bc)  
 {  
 return ab.X \* bc.Y - ab.Y \* bc.X;  
 }  
  
 private Point Vector(Point p1, Point p2)  
 {  
 return new Point(p2.X - p1.X, p2.Y - p1.Y);  
 }  
  
 private double VectorLenght(Point side)  
 {  
 return Math.*Sqrt*((side.X \* side.X) + (side.Y \* side.Y));  
 }  
  
 public void Info()  
 {  
 for (var i = 0; i < Points.Length; i++)  
 {  
 Console.*WriteLine*(i == Points.Length - 1  
 ? $"Сторона {this[i]} {this[0]}"  
 : $"Сторона {this[i]} {this[i + 1]}");  
 }  
 }  
}

CustomException.cs

namespace L9;  
  
public class InvalidInputData : Exception  
**{**  
private readonly Point \_poin;  
  
 public override string Message => $"точка ({\_poin.X}; {\_poin.Y}) не подходит для построения квадрата";  
  
 public InvalidInputData(Point poin)  
 {  
 \_poin = poin;  
 }  
**}**

Program.cs

namespace L9;  
  
public class Program  
{  
 static void *Main*()  
 {  
 Square square = new Square();  
 bool menu = true;  
 while (menu)  
 {  
 Console.*Clear*();  
 Console.*WriteLine*("1. Пересоздать фигуру");  
 Console.*WriteLine*("2. Изменить вершину фигуры");  
 Console.*WriteLine*("3. Вывести информацию о фигуре");  
 Console.*WriteLine*("4. Вывести площадь фигуры");  
 int choice = Convert.*ToInt32*(Console.*ReadLine*());  
 try  
 **{**  
switch (choice)  
 {  
 case 1:  
 {  
 try  
 {  
 Console.*Write*("Сколько точек хотите создать? ");  
 int numPoints = Convert.*ToInt32*(Console.*ReadLine*());  
 Point[] points = new Point[numPoints];  
 if (numPoints != 3) throw new Exception("Не 4 точки ");  
 for (int i = 0; i < numPoints; i++)  
 {  
 Console.*WriteLine*("{0}:", i);  
 Console.*Write*("Введите X: ");  
 double x = Convert.*ToDouble*(Console.*ReadLine*());  
 Console.*Write*("Введите Y: ");  
 double y = Convert.*ToDouble*(Console.*ReadLine*());  
 points[i] = new Point(x, y);  
 }  
  
 square = new Square(points);  
 }  
 catch (InvalidInputData e)  
 {  
 Console.*WriteLine*(e.Message);  
 square = new Square();  
 }  
 }  
 break;  
 case 2:  
 {  
 try  
 {  
 Console.*Write*("Введите номер точки: ");  
 int numPoint = Convert.*ToInt32*(Console.*ReadLine*());  
 Console.*Write*("Введите X: ");  
 double x = Convert.*ToDouble*(Console.*ReadLine*());  
 Console.*Write*("Введите Y: ");  
 double y = Convert.*ToDouble*(Console.*ReadLine*());  
 square[numPoint] = new Point(x, y);  
 Console.*WriteLine*("Точка изменена.");  
 }  
 catch (InvalidInputData e)  
 {  
 Console.*WriteLine*(e.Message);  
 square = new Square();  
 }  
 }  
 break;  
 case 4:  
 Console.*WriteLine*("Площадь многоугольника: {0}", square.Area);  
 break;  
 case 3:  
 square.Info();  
 break;  
 default:  
 menu = false;  
 Console.*WriteLine*("Нажмите любую клавишу, чтобы выйти...");  
 break;  
 }  
 **}**  
catch (Exception e)  
 {  
 Console.*WriteLine*(e.Message);  
 }  
  
 Console.*ReadKey*();  
 }  
 }  
}

Работа программы:



