**МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ**

**УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ**

**ГОМЕЛЬСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ**

**УНИВЕРСИТЕТ ИМЕНИ П.О.СУХОГО**

Факультет автоматизированных и информационных систем

Кафедра «Информационные технологии»

ЛАБОРАТОРНАЯ РАБОТА №1

по дисциплине: «Объектно-ориентированное проектирование и программирование»

на тему: **«**Основы синтаксиса объектно-ориентированного языка программирования**»**

Выполнил: студент гр. ИТП-21

Кислов Я.В.

Проверил: преподаватель

Карась О.В.

Гомель 2025

**Цель работы:**

Разработать класс, представляющий плоскую геометрическую фигуру (в соответствии с вариантом), реализовать методы для проверки её существования, вычисления характеристик (длины сторон, периметра, площади), проверки принадлежности точки фигуре, оформить код в виде библиотеки с документацией и создать консольное приложение для демонстрации работы класса.

**Ход работы:**

1. Необходимо разработать класс, представляющий собой плоскую фигуру, согласно варианта.
2. При именовании класса, полей и методов руководствоваться соглашением о наименовании кода.
3. Фигуру следует задавать координатами вершин.
4. Реализовать метод проверки возможности существования данной фигуры.
5. Реализовать методы вычисления длин сторон, площади и периметра.
6. Реализовать методы, проверяющие принадлежность точки, заданной своими координатами на плоскости, фигуре, её границе.
7. Класс должен быть размещён в библиотеке классов.
8. Весь код должен быть снабжён элементами документирования.
9. Сгенерировать XML-файл с документацией по проекту.
10. Создать консольное приложение с интерфейсом пользователя.
11. Подключить созданную библиотеку в консольное приложение.
12. Верифицировать разработанное приложение.
13. Сгенерировать исполняемый файл для последующего переноса на другие компьютеры.

Вариант 2

Вариант 2 – Окружность.

Для начала создаем структуру Point, где создаем два поля под x и y. Далее конструктор для инициализации полей. Код программы структуры Point находится в приложении А, A1. Далее идет абстрактный класс Figure в котором бы объявляем методы для реализации. Также создаем массив точек, который будет использоваться для определения фигуры. Код программы абстрактного класса Figure находится в приложении А, A2. Далее был создан класс Circle для определения окружности, который наследуется от класса Figure и реализует его методы. Окружность вписанная в квадрат и задается 4 точками, которые являются вершинами квадрата. Делаем пользовательский конструктор, который явно вызывает базовый конструктор класса Figure и делаем проверку на null и на то, что точек задано ровно 4. В противном случае вызывается исключение. Также создаем ещё пару методов для подсчетов длины, площади, а именно: GetCenterPoint(), GetRadius(), Distance(Point p1,Point p2). Код программы находится в приложении А, А3. Также создаем пользовательский интерфейс для работы с программой. Интерфейс сделан в виде меню с помощью case. Также был создан статический метод для инициализации точек вершин квадрата. Код программы находится в приложении А, А4.

Были разработаны тесты для верифицирования приложения. Находятся они в классе CircleTests. Для каждого метода класса Circle был написан тест. На рисунке 1 показан результат работы тестов. Код тестов расположен в приложении А, А5.

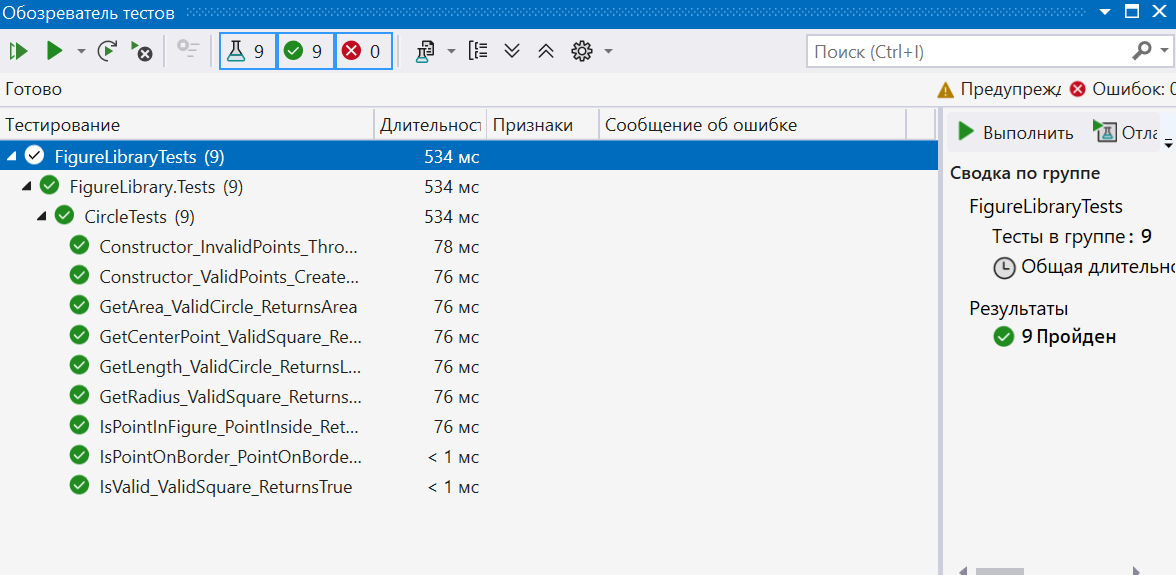


Рисунок 1 – Результат выполнения тестов

**Вывод:** в ходе лабораторной работы был успешно разработан и реализован класс, представляющий плоскую геометрическую фигуру (Окружность), в соответствии с вариантом задания. Библиотека классов была протестирована, интегрирована в консольное приложение и снабжена полной документацией.

**Приложение А**

Код программы

namespace FigureLibrary

{

/// <summary>

/// Структура, представляющая точку в двумерном пространстве

/// </summary>

public struct Point

{

/// <summary>

/// Координата X точки

/// </summary>

public double x;

/// <summary>

/// Координата Y точки

/// </summary>

public double y;

/// <summary>

/// Инициализирует новый экземпляр структуры Point с указанными координатами

/// </summary>

/// <param name="x">Координата X</param>

/// <param name="y">Координата Y</param>

public Point(double x, double y)

{

this.x = x;

this.y = y;

}

}

}

Приложение А1 – Код структуры Point

namespace FigureLibrary

{

/// <summary>

/// Абстрактный базовый класс для геометрических фигур

/// </summary>

public abstract class Figure

{

/// <summary>

/// Массив точек, определяющих фигуру

/// </summary>

public Point[] points;

/// <summary>

/// Инициализирует новый экземпляр класса Figure с указанными точками

/// </summary>

/// <param name="points">Массив точек, определяющих фигуру</param>

public Figure(Point[] points)

{

this.points = points;

}

/// <summary>

/// Вычисляет площадь фигуры

/// </summary>

/// <returns>Площадь фигуры</returns>

public abstract double GetArea();

/// <summary>

/// Вычисляет периметр или длину фигуры

/// </summary>

/// <returns>Периметр или длина фигуры</returns>

public abstract double GetLength();

/// <summary>

/// Проверяет валидность фигуры

/// </summary>

/// <returns>True если фигура валидна, иначе False</returns>

public abstract bool IsValid();

/// <summary>

/// Проверяет находится ли точка на границе фигуры

/// </summary>

/// <param name="point">Точка для проверки</param>

/// <returns>True если точка на границе, иначе False</returns>

public abstract bool IsPointOnBorder(Point point);

/// <summary>

/// Проверяет находится ли точка внутри фигуры

/// </summary>

/// <param name="point">Точка для проверки</param>

/// <returns>True если точка внутри фигуры, иначе False</returns>

public abstract bool IsPointInFigure(Point point);

}

}

Приложение А2 – Код абстрактного класса Figure

namespace FigureLibrary

{

/// <summary>

/// Представляет окружность, описанную вокруг квадрата

/// Окружность определяется четырьмя точками - вершинами квадрата

/// </summary>

public class Circle : Figure

{

/// <summary>

/// Инициализирует новый экземпляр класса Circle с указанными точками

/// </summary>

/// <param name="points">Массив из четырех точек, представляющих вершины квадрата</param>

/// <exception cref="ArgumentException">Вызывается, если points равен null или количество точек не равно 4</exception>

public Circle(Point[] points)

: base(points)

{

if (points == null || points.Length != 4)

{

throw new ArgumentException("Задано больше либо меньше 4 точек или points is null");

}

}

/// <summary>

/// Вычисляет центр окружности

/// </summary>

/// <returns>Точка центра окружности</returns>

public Point GetCenterPoint()

{

double centerX = (points[0].x + points[2].x) / 2;

double centerY = (points[0].y + points[2].y) / 2;

return new Point(centerX, centerY);

}

/// <summary>

/// Вычисляет радиус окружности

/// </summary>

/// <returns>Радиус окружности</returns>

public double GetRadius() => Distance(points[0], points[1]) / 2;

/// <summary>

/// Проверяет, образуют ли точки квадрат

/// </summary>

/// <param name="point">Массив точек для проверки</param>

/// <returns>True, если точки образуют квадрат, иначе False</returns>

public bool IsSquare(Point[] point)

{

const double error = 0.0001;

double distance1 = Distance(point[0], point[1]);

double distance2 = Distance(point[1], point[2]);

double distance3 = Distance(point[2], point[3]);

double distance4 = Distance(point[3], point[0]);

return Math.Abs(distance1 - distance2) < error &&

Math.Abs(distance2 - distance3) < error &&

Math.Abs(distance3 - distance4) < error &&

Math.Abs(distance4 - distance1) < error;

}

/// <summary>

/// Вычисляет расстояние между двумя точками

/// </summary>

/// <param name="p1">Первая точка</param>

/// <param name="p2">Вторая точка</param>

/// <returns>Расстояние между точками.</returns>

public double Distance(Point p1, Point p2) => Math.Sqrt(Math.Pow((p2.x - p1.x), 2) + Math.Pow(p2.y - p1.y, 2));

/// <summary>

/// Проверяет валидность окружности

/// Окружность считается валидной, если радиус положителен и точки образуют квадрат

/// </summary>

/// <returns>True, если окружность валидна, иначе False</returns>

public override bool IsValid()

{

if (GetRadius() > 0 && IsSquare(points))

return true;

return false;

}

/// <summary>

/// Вычисляет длину окружности

/// </summary>

/// <returns>Длина окружности</returns>

/// <exception cref="InvalidOperationException">Вызывается, если окружность не валидна</exception>

public override double GetLength()

{

if (!IsValid())

{

throw new InvalidOperationException("Окружность не валидна");

}

return 2 \* Math.PI \* GetRadius();

}

/// <summary>

/// Вычисляет площадь круга

/// </summary>

/// <returns>Площадь круга</returns>

/// <exception cref="InvalidOperationException">Вызывается, если окружность не валидна</exception>

public override double GetArea()

{

if (!IsValid())

{

throw new InvalidOperationException("Окружность не валидна");

}

return Math.PI \* Math.Pow(GetRadius(), 2);

}

/// <summary>

/// Проверяет, находится ли точка на границе окружности

/// </summary>

/// <param name="point">Точка для проверки</param>

/// <returns>True, если точка находится на границе окружности; иначе False</returns>

/// <exception cref="InvalidOperationException">Вызывается, если окружность не валидна</exception>

public override bool IsPointOnBorder(Point point)

{

if (!IsValid())

{

throw new InvalidOperationException("Окружность не валидна");

}

double distance = Distance(GetCenterPoint(), point);

return Math.Abs(distance - GetRadius()) < 0.1;

}

/// <summary>

/// Проверяет, находится ли точка внутри окружности

/// </summary>

/// <param name="point">Точка для проверки.</param>

/// <returns>True, если точка находится внутри окружности; иначе False</returns>

/// <exception cref="InvalidOperationException">Вызывается, если окружность не валидна</exception>

public override bool IsPointInFigure(Point point)

{

if (!IsValid())

{

throw new InvalidOperationException("Окружность не валидна");

}

double distance = Distance(GetCenterPoint(), point);

return distance <= GetRadius();

}

}

}

Приложение А3 – Код программы класса Circle

namespace CircleApp

{

internal class Program

{

static void Main(string[] args)

{

try

{

Console.WriteLine("=== ПРОГРАММА ДЛЯ РАБОТЫ С ОКРУЖНОСТЬЮ ===");

Point[] points = CreateCirclePoints();

Circle circle = new Circle(points);

Console.WriteLine("Окружность создана успешно!");

while (true)

{

Console.WriteLine("\n=== ГЛАВНОЕ МЕНЮ ===");

Console.WriteLine("1. Проверить валидность окружности");

Console.WriteLine("2. Вычислить длину окружности");

Console.WriteLine("3. Вычислить площадь окружности");

Console.WriteLine("4. Проверить точку на границе");

Console.WriteLine("5. Проверить точку внутри окружности");

Console.WriteLine("6. Выход");

Console.Write("Выберите действие: ");

string choice = Console.ReadLine();

switch (choice)

{

case "1":

if (circle.IsValid())

Console.WriteLine("\nОкружность валидна!");

else

Console.WriteLine("\nОкружность не валдина!");

break;

case "2":

Console.WriteLine($"\nДлина окружности: {circle.GetLength()}");

break;

case "3":

Console.WriteLine($"\nПлощадь окружности: {circle.GetArea()}");

break;

case "4":

Console.WriteLine("Введите точку: ");

Point point1 = new Point();

point1.x = Double.Parse(Console.ReadLine());

point1.y = Double.Parse(Console.ReadLine());

if (circle.IsPointOnBorder(point1))

Console.WriteLine("\nТочка на границе окружности");

else

Console.WriteLine("\nТочка не на границе окружности");

break;

case "5":

Console.WriteLine("Введите точку: ");

Point point2 = new Point();

point2.x = Double.Parse(Console.ReadLine());

point2.y = Double.Parse(Console.ReadLine());

if (circle.IsPointInFigure(point2))

Console.WriteLine("\nТочка в окружности");

else

Console.WriteLine("\nТочка за окружностью");

break;

case "6":

return;

default:

Console.WriteLine("Неверный выбор. Попробуйте снова.");

break;

}

}

}

catch (Exception ex)

{

Console.WriteLine($"Возникло исключение: {ex.Message}");

Console.ReadKey();

}

}

public static Point[] CreateCirclePoints()

{

Point[] points = new Point[4];

for (int i = 0; i < points.Length; i++)

{

Console.WriteLine($"Точка {i + 1}: ");

points[i].x = Double.Parse(Console.ReadLine());

points[i].y = Double.Parse(Console.ReadLine());

}

return points;

}

}

}

Приложение А4 – Код программы Main и статический метод CreateCirclePoints()

namespace FigureLibrary.Tests

{

[TestClass]

public class CircleTests

{

[TestMethod]

public void Constructor\_ValidPoints\_CreatesCircle()

{

Point[] points = new Point[]

{

new Point(0, 0),

new Point(2, 0),

new Point(2, 2),

new Point(0, 2)

};

Circle circle = new Circle(points);

Assert.IsNotNull(circle);

}

[TestMethod]

[ExpectedException(typeof(ArgumentException))]

public void Constructor\_InvalidPoints\_ThrowsException()

{

Point[] points = new Point[]

{

new Point(0, 0),

new Point(2, 0),

new Point(2, 2)

};

Circle circle = new Circle(points);

}

[TestMethod]

public void GetRadius\_ValidSquare\_ReturnsRadius()

{

Point[] points = new Point[]

{

new Point(0, 0),

new Point(4, 0),

new Point(4, 4),

new Point(0, 4)

};

Circle circle = new Circle(points);

double radius = circle.GetRadius();

Assert.AreEqual(2.0, radius, 0.001);

}

[TestMethod]

public void IsValid\_ValidSquare\_ReturnsTrue()

{

Point[] points = new Point[]

{

new Point(0, 0),

new Point(2, 0),

new Point(2, 2),

new Point(0, 2)

};

Circle circle = new Circle(points);

bool isValid = circle.IsValid();

Assert.IsTrue(isValid);

}

[TestMethod]

public void GetArea\_ValidCircle\_ReturnsArea()

{

Point[] points = new Point[]

{

new Point(0, 0),

new Point(2, 0),

new Point(2, 2),

new Point(0, 2)

};

Circle circle = new Circle(points);

double area = circle.GetArea();

double expectedArea = Math.PI \* 1 \* 1;

Assert.AreEqual(expectedArea, area, 0.001);

}

[TestMethod]

public void GetLength\_ValidCircle\_ReturnsLength()

{

Point[] points = new Point[]

{

new Point(0, 0),

new Point(2, 0),

new Point(2, 2),

new Point(0, 2)

};

Circle circle = new Circle(points);

double length = circle.GetLength();

double expectedLength = 2 \* Math.PI \* 1;

Assert.AreEqual(expectedLength, length, 0.001);

}

[TestMethod]

public void IsPointInFigure\_PointInside\_ReturnsTrue()

{

Point[] points = new Point[]

{

new Point(0, 0),

new Point(4, 0),

new Point(4, 4),

new Point(0, 4)

};

Circle circle = new Circle(points);

Point testPoint = new Point(2, 2);

bool isInside = circle.IsPointInFigure(testPoint);

Assert.IsTrue(isInside);

}

[TestMethod]

public void IsPointOnBorder\_PointOnBorder\_ReturnsTrue()

{

Point[] points = new Point[]

{

new Point(0, 0),

new Point(4, 0),

new Point(4, 4),

new Point(0, 4)

};

Circle circle = new Circle(points);

Point testPoint = new Point(2, 0);

bool isOnBorder = circle.IsPointOnBorder(testPoint);

Assert.IsTrue(isOnBorder);

}

[TestMethod]

public void GetCenterPoint\_ValidSquare\_ReturnsCenter()

{

Point[] points = new Point[]

{

new Point(1, 1),

new Point(5, 1),

new Point(5, 5),

new Point(1, 5)

};

Circle circle = new Circle(points);

Point center = circle.GetCenterPoint();

Assert.AreEqual(3, center.x, 0.001);

Assert.AreEqual(3, center.y, 0.001);

}

}

}

Приложение А5 – Код программы тестов CircleTests