Задание:

11 вариант Класс «Круг»
Член-данные класса
Координаты центра круга и его радиус.
Методы класса
1) конструкторы: по умолчанию, с аргументами (или с аргументами по умолчанию);
2) ввод/вывод круга;
3) нахождение площади;
4) нахождение длины окружности;
5) умножение круга на число (центр не меняется, радиус

умножается на число);
6) проверка двух кругов на равенство (два круга равны, если равны их радиусы);
7) проверка, попадает ли заданная точка в круг
8) проверка пересечения кругов;
9) проверка, лежит ли один круг внутри другого;
10) нахождение радиуса круга, вписанного в пересечение.

Код:

```
using System;

namespace Rusik {
class TestRunner {
public static void RunTests()
{
TestCircleInput();
TestCircleOutput();
TestCircleSquare();
```

```
TestCircleCircumference();
TestCircleMul();
TestCircleEquation();
TestCircleContainsPoint();
TestCircleIntersection();
TestCircleContains();
TestCircleRadius();
}

private static void AssertEqual(double expected, double actual, double tolerance = 0.0001)
{
if (Math.Abs(expected - actual) > tolerance)
{
throw new Exception($"Assertion failed. Expected: {expected}, Actual: {actual}");
}
}

private static void AssertTrue(bool condition)
{
if (!condition)
{
throw new Exception("Assertion failed. Condition is not true.");
}
}

private static void AssertFalse(bool condition)
{
if (condition)
{
throw new Exception("Assertion failed. Condition is not false.");
}
}

private static void TestCircleInput()
{
var circle = new Circle(0, 0, 5);
AssertEqual(5, circle.r);

try
{
new Circle(0, 0, -5);
throw new Exception("Expected exception for negative radius not thrown.");
}
```

```
catch (Exception) { }
}

private static void TestCircleOutput()
{
var circle = new Circle(0, 0, 5);
AssertEqual(5, circle.r);
}

private static void TestCircleSquare()
{
var circle = new Circle(0, 0, 5);
AssertEqual(Math.PI * 25, circle.square());
}

private static void TestCircleCircumference()
{
var circle = new Circle(0, 0, 5);
AssertEqual(2 * Math.PI * 5, Circle.circumference(circle));
}

private static void TestCircleMul()
{
var circle = new Circle(0, 0, 5);
var newCircle = Circle.mul(circle, 2);
AssertEqual(10, newCircle.r);
}

private static void TestCircleEquation()
{
var circle = new Circle(0, 0, 5);

var circle2 = new Circle(1, 2, 3);
AssertFalse(Circle.equation(circle, circle2));
}

private static void TestCircleContainsPoint()
{
var circle = new Circle(0, 0, 5);
AssertTrue(Circle.containsPoint(3, 4, circle));
AssertTrue(Circle.containsPoint(0, 0, circle));
AssertFalse(Circle.containsPoint(6, 0, circle));
}
```

```
private static void TestCircleIntersection()
{
var circle1 = new Circle(0, 0, 5);
var circle2 = new Circle(8, 0, 5);
var circle3 = new Circle(11, 0, 5);

AssertTrue(Circle.intersection(circle1, circle2));
AssertFalse(Circle.intersection(circle1, circle3));
}

private static void TestCircleContains()
{
var circle1 = new Circle(0, 0, 10);
var circle2 = new Circle(0, 0, 5);
var circle3 = new Circle(8, 0, 5);

AssertTrue(Circle.contains(circle1, circle2));
AssertFalse(Circle.contains(circle1, circle3));
}

private static void TestCircleRadius()
{
var circle = new Circle(0, 0, 5);
AssertEqual(5, circle.r);

var newCircle = Circle.mul(circle, 2);
AssertEqual(10, newCircle.r);
}
}

interface Figure{
double square();
void input();
void output();
}
public class Circle : Figure{
public static void Main(string[] args){
TestRunner.RunTests();
}
double _x, _y, _r;
public Circle(double center_x=1, double center_y=1, double radius=1){
this._x = center_x;
```

```
this._y = center_y;
if(radius <= 0){throw new Exception("Negative or null radius");}
this._r = radius;
}
public double x{
get{return _x;}
set{_x = value;}
}
public double y{
get{return _y;}
set{_y = value;}
}
public double r{
get{return _r;}
set{
if (value <= 0){throw new Exception("Negative or null radius");}
_r = value;}
}
void Figure.input(){
Console.WriteLine("input x coordinate: ");
x = Convert.ToDouble(Console.ReadLine());
Console.WriteLine("input y coordinate: ");
y = Convert.ToDouble(Console.ReadLine());
Console.WriteLine("input radius: ");
r = Math.Abs(Convert.ToDouble(Console.ReadLine()));
}
void Figure.output(){
Console.WriteLine("x coordinate: {0}", x);
Console.WriteLine("y coordinate: {0}", y);
Console.WriteLine("r coordinate: {0}", r);
}
public double square(){
return Math.PI * r * r;
}
public static double circumference(Circle circle){
return 2 * Math.PI * circle.r;
}
public static Circle mul(Circle circle, int n){
circle.r *= n;
return circle;
}
public static bool equation(Circle circle, Circle circle2){
return circle.r == circle2.r;
}
```

```
public static bool containsPoint(double xPoint, double yPoint, Circle circle){
double distance = Math.Sqrt(Math.Pow(xPoint - circle.x, 2) + Math.Pow(yPoint - circle.y, 2));
return distance <= circle.r;
}
public static bool intersection(Circle circle1, Circle circle2){
double tmp = (circle2.x+circle1.x)*(circle2.x+circle1.x) + (circle2.y+circle1.y)*(circle2.y+circle1.y);
return ((circle1.r-circle2.r)*(circle1.r-circle2.r) <= tmp && tmp <=
(circle2.r+circle1.r)*(circle2.r+circle1.r));
}

public static bool contains(Circle circle1, Circle circle2){
double d = Math.Sqrt((circle1.x - circle2.x)*(circle1.x - circle2.x) + (circle1.y - circle2.y)*(circle1.y -
circle2.y));
return circle1.r > d + circle2.r;
}

public static float radius(Circle circle1, Circle circle2){
if(intersection(circle1, circle2)){
double d = Math.Sqrt((circle1.x - circle2.x)*(circle1.x - circle2.x) + (circle1.y - circle2.y)*(circle1.y -
circle2.y));
return (float)Convert.ToDouble((circle1.r + circle2.r - d))/2;
}
return 0;
}
}
}
```

Класс `TestRunner` предназначен для проверки функциональности класса `Circle` и связанных с ним методов. Ниже приведено описание того, как проходят тесты и какие случаи рассматриваются:

1. `TestCircleInput()`

- **Цель**: Убедиться, что конструктор `Circle` правильно инициализирует круг и обрабатывает некорректные входные данные.

- **Случаи**:

- Создание круга с положительным радиусом (5). Радиус должен быть установлен правильно.

- Попытка создать круг с отрицательным радиусом (-5). Должно быть выброшено исключение.

2. `TestCircleOutput()`

- **Цель**: Проверить, что радиус круга правильно сохраняется и возвращается.

- **Случаи**:

- Создание круга с радиусом 5 и проверка, что радиус установлен как 5.

3. `TestCircleSquare()`

- **Цель**: Убедиться, что метод `square` правильно вычисляет площадь круга.

- **Случаи**:

- Вычисление площади круга с радиусом 5. Ожидаемая площадь равна $\pi \times 5^2 = 25\pi$.

4. `TestCircleCircumference()`

- **Цель**: Проверить, что метод `circumference` правильно вычисляет длину окружности.

- **Случаи**:

- Вычисление длины окружности круга с радиусом 5. Ожидаемая длина окружности равна $2\pi \times 5 = 10\pi$.

5. `TestCircleMul()`

- **Цель**: Проверить метод `mul`, который масштабирует радиус круга на заданный коэффициент.

- **Случаи**:

- Умножение круга с радиусом 5 на 2. Новый радиус должен быть равен 10.

6. `TestCircleEquation()`

- **Цель**: Проверить, правильно ли метод `equation` определяет, имеют ли два круга одинаковое уравнение.

- **Случаи**:

  - Сравнение двух различных кругов. Метод должен вернуть `false`, так как уравнения не совпадают.

7. `TestCircleContainsPoint()`

- **Цель**: Проверить, правильно ли метод `containsPoint` определяет, находится ли точка внутри круга или на его границе.

- **Случаи**:

  - Проверка, находится ли точка (3, 4) внутри круга с радиусом 5 и центром в (0, 0).

  - Проверка, содержит ли круг свою центральную точку (0, 0).

  - Проверка, находится ли точка (6, 0) за пределами круга.

8. `TestCircleIntersection()`

- **Цель**: Проверить метод `intersection`, который определяет, пересекаются ли два круга.

- **Случаи**:

  - Два круга с радиусом 5 и центрами (0, 0) и (8, 0) соответственно. Они должны пересекаться.

  - Два круга с радиусом 5 и центрами (0, 0) и (11, 0) соответственно. Они не должны пересекаться.

9. `TestCircleContains()`

- **Цель**: Проверить метод `contains`, который определяет, содержит ли один круг другой.

- **Случаи**:

  - Большой круг с радиусом 10 и центром в (0, 0) содержит меньший круг с радиусом 5 и тем же центром.

  - Тот же большой круг не содержит другой круг с радиусом 5 и центром в (8, 0).

10. `TestCircleRadius()`

- **Цель**: Убедиться, что радиус круга правильно управляется и может быть масштабирован.

- **Случаи**:

  - Проверка, что радиус круга, созданного с радиусом 5, равен 5.

  - Проверка, что умножение этого круга на 2 приводит к новому кругу с радиусом 10.

Как проходят тесты

1. **Инициализация тестов**: Метод `RunTests` последовательно вызывает каждый тестовый метод.

2. **Утверждения**: Каждый тестовый метод использует собственные методы утверждений (`AssertEqual`, `AssertTrue` и `AssertFalse`) для проверки условий.

3. **Обработка ошибок**: Если какое-либо утверждение не выполняется, выбрасывается исключение с описательным сообщением.

4. **Вывод тестов**: Метод `Main` в классе `Program` запускает все тесты и выводит "All tests passed.", если все тесты прошли успешно, или выводит сообщение об ошибке, если какой-либо тест не прошел.

Этот тестовый модуль обеспечивает полное покрытие методов класса `Circle` и гарантирует, что проверяются различные граничные случаи и типичные сценарии использования.