PROG8170

Software Quality Assurance Techniques

Assignment #1

Akshay Madhu

7753114

Submitted On:6/1/2017

**Program Source Code (Program.cs):**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace CircleModify

{

public class Program

{

static void Main(string[] args)

{

int choice = 0;

double radius = 0;

//Reading radius from user

Console.WriteLine("Enter the radius");

radius = Convert.ToDouble(Console.ReadLine());

//Creating an object of class Circle with paramter radius

Circle circleObj1 = new Circle(radius);

do

{

Console.Clear();

Console.WriteLine("Select From the menu\n");

Console.WriteLine("1.Add radius");

Console.WriteLine("2.Subtract radius");

Console.WriteLine("3.Find Circumference");

Console.WriteLine("4.Find Area");

Console.WriteLine("5.Exit");

try

{

Console.Write("Please Enter your choice(1/2/3/4/5) : ");

choice = Convert.ToInt16(Console.ReadLine());

}

catch (Exception e)

{

Console.WriteLine("Your choice is invalid...please enter a valid chocie(1/2/3/4/5)\n");

Console.WriteLine("Press any key to continue");

Console.ReadLine();

}

finally

{

Console.Clear();

}

switch (choice)

{

case 1:

double numberToADD=0;

try

{

Console.WriteLine("Enter the value to Add\n");

numberToADD = Convert.ToInt32(Console.ReadLine());

circleObj1.AddToRadius(numberToADD);

}

catch (Exception e)

{

Console.WriteLine("Your input is invalid..A number is expected\n");

Console.WriteLine("Press any key to continue");

Console.ReadLine();

}

break;

case 2:

try

{

int set = 0;

double numberToSubtract = 0;

do

{

Console.WriteLine("Enter the value to Subtract\n");

numberToSubtract = Convert.ToInt32(Console.ReadLine());

if (numberToSubtract > radius)

{

Console.WriteLine("Enter a value less than the given radius\n");

}

else

{

set = 1;

}

} while (set != 1);

circleObj1.SubtractFromRadius(numberToSubtract);

}

catch (Exception e)

{

Console.WriteLine("Your input is invalid..A number is expected\n");

Console.WriteLine("Press any key to continue");

Console.ReadLine();

}

break;

case 3:

double circumference = circleObj1.GetCircumference();

Console.WriteLine("\nCircumference of the cirle is:" + circumference);

break;

case 4:

double area = circleObj1.GetArea();

Console.WriteLine("\narea of the cirle is:" + area);

break;

case 5:

Environment.Exit(0);

break;

default:

Console.WriteLine("Please enter a valid choice");

Console.WriteLine("Enter a key to continue..");

Console.ReadLine();

break;

}

} while (choice != 5);

}

}

}

**Circle Class Source Code(Circle.cs)**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace CircleModify

{

public class Circle

{

public const double pie= 3.14;

public double radiusOfCircle { get; set; }

//Default Constructor

public Circle()

{

}

//non-default Constructor

public Circle(double radius)

{

this.radiusOfCircle = radius;

Console.WriteLine("Radius=" +this.radiusOfCircle);

}

//Function to add radius

public void AddToRadius(double num)

{

this.radiusOfCircle = this.radiusOfCircle + num;

Console.WriteLine(" Radius is {0}", radiusOfCircle);

Console.WriteLine("Press a key to continue");

Console.ReadLine();

}

//Function to subract from the given radius

public void SubtractFromRadius(double num)

{

this.radiusOfCircle = this.radiusOfCircle - (num);

Console.WriteLine("Radius is {0}", radiusOfCircle);

Console.WriteLine("Press a key to continue..");

Console.ReadLine();

}

//Function to calculate circumfrence from given radius

public double GetCircumference()

{

double circumference=0;

if (this.radiusOfCircle > 0)

{

circumference = 2 \* pie \* radiusOfCircle;

return circumference;

}

else

{

return circumference;

}

}

//Function to calculate area from given radius

public double GetArea()

{

double area=0;

if (this.radiusOfCircle > 0)

{

area = pie \* radiusOfCircle \* radiusOfCircle;

return area;

}

else

{

return area;

}

}

}

}

**Unit Test Class Source Code(CircleTest.cs)**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using NUnit.Framework;

using CircleModify;

namespace CircleModify\_Test

{

[TestFixture]

public class CircleTest

{

[Test]

public void Constructortest()

{

Circle constructtestobj = new Circle(2);

Assert.AreEqual(2, constructtestobj.radiusOfCircle);

}

[Test]

public void AddToRadius\_withPositiveradius()

{

Circle Positiveradius = new Circle();

Positiveradius.radiusOfCircle = 10;

Positiveradius.AddToRadius(6);

Assert.AreEqual(16, Positiveradius.radiusOfCircle);

}

[Test]

public void AddToRadius\_withnegetiveradiusTest()

{

Circle negativeradius = new Circle();

negativeradius.radiusOfCircle = 2;

negativeradius.AddToRadius(-4);

Assert.AreEqual(-2, negativeradius.radiusOfCircle);

}

[Test]

public void SubtractFromRadius\_withPositiveradius()

{

Circle Positiverad = new Circle();

Positiverad.radiusOfCircle = 10;

Positiverad.SubtractFromRadius(6);

Assert.AreEqual(4, Positiverad.radiusOfCircle);

}

[Test]

public void SubtractFromRadius\_withnegetiveradius()

{

Circle negativerad = new Circle();

negativerad.radiusOfCircle = 2;

negativerad.SubtractFromRadius(6);

Assert.AreEqual(-4, negativerad.radiusOfCircle);

}

[Test]

public void GetCircumference\_postiveradius()

{

Circle circletestobj1 = new Circle();

circletestobj1.radiusOfCircle = 3;

circletestobj1.GetCircumference();

Assert.AreEqual(18.84, circletestobj1.GetCircumference());

}

[Test]

public void GetCircumference\_negetiveradius()

{

Circle circletestobj2 = new Circle();

circletestobj2.radiusOfCircle = -2;

circletestobj2.GetCircumference();

Assert.AreEqual(0, circletestobj2.GetCircumference());

}

[Test]

public void GetAreaTest\_positiveradius()

{

Circle circletestobj3 = new Circle();

circletestobj3.radiusOfCircle = 2;

circletestobj3.GetArea();

Assert.AreEqual(12.56, circletestobj3.GetArea());

}

[Test]

public void GetAreaTest\_negetiveradius()

{

Circle circletestobj4 = new Circle();

circletestobj4.radiusOfCircle = -2;

circletestobj4.GetArea();

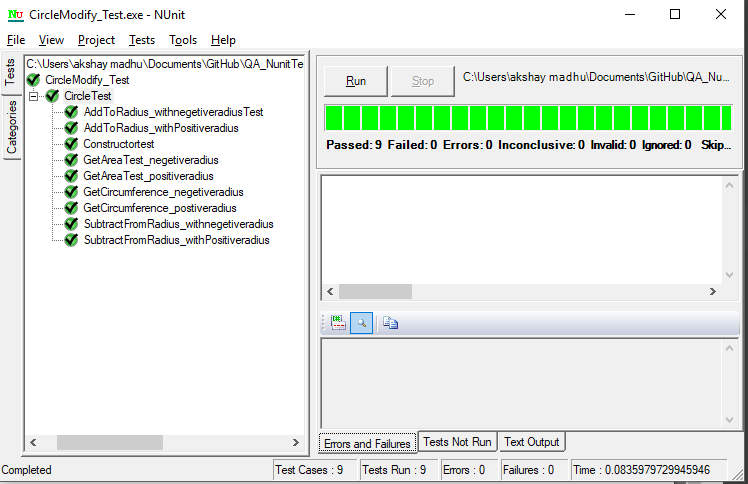
Assert.AreEqual(0, circletestobj4.GetArea());

}

}

}

**Results of the Unit Test:**



**Test1:**

**AddToRadius\_withnegetiveradiusTest**:This is done to test the AddToRadius function,whether the function adds a negetive value to the entered radius.

**Test2:**

**AddToRadius\_withPositiveRadius**:This test is done to test the AddToRAdius function, whether the function adds all positive value to the entered radius..

**Test3:**

**ConstructorTest**: This is done to test the whether the non-default constructor takes the radius value.

**Test4:**

**GetAreaTest\_negetiveradius**: This test is done whether the GetArea function accepts negative values of radius. The test returns a 0 if a negative radius is given. The function won’t accept negative radius as negative area is incorrect.

**Test5:**

**GetAreaTest\_Postiveradius**: This test is done whether the GetArea function accepts all positive values of radius.

**Test6:**

**GetCircumference\_negetiveradius**: This test is done whether the GetCircumferece function accepts negative values of radius. The test returns 0 if a negative radius is given. The function won’t accept negative radius as negative circumference is incorrect.

**Test7:**

**GetCircumference\_positiveradius**: This test is done whether the GetArea function accepts all positive values of radius.

**Test8:**

**SubtractFromRadius\_withPositiveradius**:This test is done to test SubtractFromRadius function whether the function subtracts a radius value from the entered radius value

**Test9:**

**SubtractFrom\_Radius\_withnegetiveradius:**This test is done to test SubtractFromRadius function whether the function subtracts a higher value from the entered radius.