# Test – Softwareentwicklung 4. JG (16.2.2023)

Name: Johannes Werner

|  |  |  |
| --- | --- | --- |
| **Beispiel** | **Mögliche Punkte** | **Punkte** |
| **Theorie** | 14 |  |
| **IShape** | 7 |  |
| **PrimeNumbersAsync** | 4 |  |
| Gesamt | **25** |  |

## IShape (5 Punkte)

Gegeben ist folgendes Interface:

public interface IShape

{

public double CalculateArea();

public double CalculateCircumference();

}

Implementieren Sie die Klassen Rectangle und Circle, welche jeweils das Interface implementieren. Überlegen Sie für jede Klasse einen sinnvollen Konstruktor.

Um die Klassen zu testen, führen Sie folgende Aktionen aus:

* Erzeugen Sie jeweils zwei Objekte jeder Klasse
* Erzeugen Sie eine Collection (generische Liste oder Array) und fügen Sie die vier Objekte dieser Collection hinzu. Überlegen, Sie welcher Datentyp für die Collection zielführend ist.
* Geben Sie Fläche und Umfang jedes Objekts in der Collection aus (in einer Schleife).

# Code

Circle.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace IShape\_2\_1

{

internal class Circle : IShape

{

private double radius;

public Circle(double radius)

{

this.radius = radius;

}

public double CalculateArea()

{

return Math.Round(Math.Sqrt(radius) \* Math.PI, 2);

}

public double CalculateCircumference()

{

return Math.Round(2\* radius \* Math.PI, 2);

}

}

}

Rectangle.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace IShape\_2\_1

{

internal class Rectangle : IShape

{

private double sideA;

private double sideB;

public Rectangle(double sideA, double sideB)

{

this.sideA = sideA;

this.sideB = sideB;

}

public double CalculateArea()

{

return Math.Round(sideA \* sideB, 2);

}

public double CalculateCircumference()

{

return Math.Round(2 \* sideA + 2\* sideB, 2);

}

}

}

Programm.cs

using IShape\_2\_1;

List <IShape> shapes = new List <IShape> ();

shapes.Add(new Circle(3.5));

shapes.Add(new Rectangle(3, 4));

shapes.Add(new Circle(10));

shapes.Add(new Rectangle(5, 5));

shapes.ForEach(item => {

Console.WriteLine("Area: " + item.CalculateArea());

Console.WriteLine("Cirumference: " + item.CalculateCircumference());

});

## PrimeNumbersAsync (4 Punkte)

Gegeben ist die Klasse PrimeNumber, welche eine Methode GeneratePrimeNumbersAsync anbietet.

public class PrimeNumber

{

public Action<int> PrimeNumberFound;

// event anbieten, wann Primenumber gefunden mit Func und Action

public async Task GeneratePrimeNumbersAsync(int start, int end)

{

await Task.Run(() =>

{

for (int i = start; i < end; i++)

{

if (IsPrime(i))

{

if(PrimeNumberFound != null)

{

PrimeNumberFound(i);

}

}

}

});

}

private bool IsPrime(int number)

{

if (number <= 1)

{

return false;

}

for (int i = 2; i <= Math.Sqrt(number); i++)

{

if (number % i == 0)

{

return false;

}

}

return true;

}

}

Führen sie im Hauptprogramm folgende Aktionen durch:

* Erzeugen eines Objekts der Klasse PrimeNumber
* Melden Sie sich für das delegate PrimeNumberFound an. Wenn eine Primzahl gefunden wurde, soll diese auf der Konsole ausgegeben werden.
* Aufruf der Methode GeneratePrimeNumbersAsync.
* Erzeugen eines weiteren Objekts der Klasse PrimeNumber.
* Konsumieren Sie das delegte PrimeNumberFound in Form einer Lambda-Expression. Wenn eine Primzahl gefunden wurde, soll diese auf der Konsole ausgegeben werden.

# Code

Programm.cs

using PrimeNumbers\_2\_2;

int startNumber;

int endNumber;

PrimeNumber primeNumber = new PrimeNumber();

primeNumber.PrimeNumberFound += prim => Console.WriteLine(prim.ToString() +" is a prime number!");

Console.WriteLine("For the search of a prime number you have to define a start and a end number.");

while (true)

{

Console.Write("Enter a start number: ");

if (int.TryParse(Console.ReadLine(), out startNumber))

{

break;

}

}

while (true)

{

Console.Write("Enter a end number: ");

if (int.TryParse(Console.ReadLine(), out endNumber))

{

break;

}

}

Console.WriteLine("Search for prime numbers between {0} and {1} started!", startNumber, endNumber);

Task t = primeNumber.GeneratePrimeNumbersAsync(startNumber, endNumber);

t.Wait();

Console.WriteLine("Search for prime numbers between {0} and {1} finished!", startNumber, endNumber);

**Geben Sie dieses Dokument und Ihre Programme (Zip-Archiv des gesamten Projektordners) in Eduvidual ab.**