2020/21 - Wprowadzenie do programowania (C#) - LAB (K. Molenda)

Pytanie **3**

Niepoprawnie

Ocena: 0,00 z 5,00

Oflaguj pytanie

Figury - hierarchia klas

Twoim zadaniem jest rozbudowa hierarchii klas.

Pracujesz w przestrzeni nazw `FiguryLib`.

Dane są interfejsy `IMierzalna1D`, `IMierzalna2D`, klasa abstrakcyjna `Figura` oraz klasy: `Punkt` i `Odcinek`. Kod - dla celów informacyjnych - podano poniżej:

```
public interface IMierzalna1D
{
    double Dlugosc { get; }
}

public interface IMierzalna2D : IMierzalna1D
{
    double Pole { get; }
    double Obwod { get; }
}

abstract public class Figura
{
    public ConsoleColor DefaultColor { get; protected set; } = ConsoleColor.Black;
    // wypisuje na konsolę figurę
    public virtual void Rysuj()
    {
        Console.ResetColor();
        Console.ForegroundColor = this.DefaultColor;
        Console.WriteLine(this);
        Console.ResetColor();
    }
}

// immutable
public class Punkt : Figura, IEquatable<Punkt>
{
    public readonly int X, Y;
    public Punkt(int x = 0, int y = 0) { X = x; Y = y; }
    public override string ToString() => $"P({X}, {Y})";
    public bool Equals(Punkt other) =>
        other != null && X == other.X && Y == other.Y;
}

public class Odcinek : Figura, IMierzalna1D, IEquatable<Odcinek>
{
    public Punkt P1 { get; private set; }
    public Punkt P2 { get; private set; }

    public Odcinek() : this(new Punkt(), new Punkt())
    { }

    public Odcinek(Punkt p1, Punkt p2)
    {
        if (p1 is null || p2 is null)
            throw new ArgumentException("Punkt nie może być null");
        P1 = p1; P2 = p2;
        DefaultColor = ConsoleColor.Blue;
    }

    public double Dlugosc =>
        Math.Round(Math.Sqrt((P2.X - P1.X) * (P2.X - P1.X) + (P2.Y - P1.Y) * (P2.Y - P1.Y)), 2);

    public override string ToString() => $"L({P1}, {P2})";

    public override void Rysuj()
    {
        Console.ResetColor();
```

```
        Console.ForegroundColor = this.DefaultColor;
        Console.WriteLine(this + $" dlugosc={Dlugosc:F2}");
        Console.ResetColor();
    }

    public bool Equals(Odcinek other) =>
        other != null && P1 == other.P1 && P2 == other.P2;
}
```

Zaimplementuj klasę `Kwadrat`, spełniającą poniższe warunki:

- boki kwadratu są równoległe do osi układu współrzędnych
- kwadrat zdefiniowany jest przez punkt określający jego lewy dolny wierzchołek (publiczna właściwość `LewyDolny`, read-write) oraz długość boku (publiczna właściwość `Bok`, read-write)
  - punkt nie może być `null` - próba przypisania `null` powoduje zgłoszenie wyjątku `ArgumentException` z komunikatem `Punkt nie może być null`
  - bok jest nieujemną liczbą całkowitą. Próba przypisania wartości ujemnej skutkuje przypisaniem wartości `0`
- domyślnym kwadratem jest kwadrat o lewym dolnym wierzchołku w `(0,0)` i boku o długości `1`
- domyślnym kolorem kwadratu jest `Green`
- klasa `Kwadrat` dziedziczy z `Figura` i implementuje interfejs `IMierzalna2D`
- tekstowa reprezentacja obiektu w postaci `K({LewyDolny}, {Bok})`
- rysowanie kwadratu polega na wypisaniu tekstu w kolorze domyślnym o formacie `K({LewyDolny}, {Bok}) obwod={Obwod} pole={Pole}`, wymiary w zaokrągleniu do 2 miejsc po przecinku

Bazując na klasie `Kwadrat` zaimplementuj klasę `Prostokat`, spełniającą poniższe warunki:

- `Prostokat` jest uogólnieniem pojęcia `Kwadrat` w rozumieniu "prostokąt = rozciągnięty lub ściśnięty kwadrat". Zatem prostokąt, oprócz jednego boku (zdefiniowanego w kwadracie), ma również drugi bok (publiczna właściwość `BokDrugi`, read-write)
- klasa `Prostokat` dziedziczy z klasy `Kwadrat` (i implementuje interfejs `IMierzalna2D`)
- domyślnym prostokątem jest prostokąt o lewym dolnym wierzchołku w `(0,0)` i długości pierwszego boku `1` oraz drugiego boku `2`
- domyślnym kolorem prostokąta jest `Yellow`
- tekstowa reprezentacja obiektu w postaci `R({LewyDolny}, {Bok} x {BokDrugi})`
- rysowanie prostokąta polega na wypisaniu tekstu w kolorze domyślnym o formacie `R({LewyDolny}, {Bok} x {BokDrugi}) obwod={Obwod} pole={Pole}`, wymiary w zaokrągleniu do 2 miejsc po przecinku

Dany jest niekompletny kod klasy `Ekran`. Obiekt tego typu przechowuje prywatną listę zarejestrowanych figur (figury można dodawać i usuwać za pomocą dedykowanych metod).

```
public class Ekran
{
    private List<Figura> figury = new List<Figura>();
    public void Dodaj(Figura f) => figury.Add(f);
    public void Usun(Figura f) => figury.Remove(f);
    public void Rysuj() => figury.ForEach(f => f.Rysuj());

    public double SumarycznyObwod() => throw new NotImplementedException();
    public double SumarycznePole() => throw new NotImplementedException();
}
```

Uzupełnij kod klasy, implementując metody:

- `SumarycznyObwod` - zwraca sumę obwodów wszystkich zarejestrowanych figur, w zaokrągleniu do 2 miejsc po przecinku.
- `SumarycznePole` - zwraca sumę pól powierzchni wszystkich zarejestrowanych figur, w zaokrągleniu do 2 miejsc po przecinku.

Ocenianie

Implementując kod sugeruj się udostępnianymi testami.

Ocena zależy od liczby zaliczonych testów i liczby sprawdzeń kodu. Początkowe próby sprawdzenia nie są obarczone odjęciem punktów.

Na przykład:

Test	Wynik

`// klasa Kwadrat, modyfikacje`
`Kwadrat k;`
`try{`
`  k = new Kwadrat(null, -1);`
`  Console.WriteLine(k);`
`}`
`catch(ArgumentException e){`
`  Console.WriteLine(e.Message);`
`}`
 	
`k = new Kwadrat(new Punkt(1,1),1);`
`Console.WriteLine(k);`
`k.LewyDolny = new Punkt(0,0);`
`k.Bok = 5;`
	`Punkt nie może być null`
	`K(P(1, 1), 1)`
	`K(P(0, 0), 5) obwod=20.00, pole=25.00`
	`K(P(0, 0), 0) obwod=0.00, pole=0.00`

```
k.Rysuj();
k.Bok = -1;
k.Rysuj();
```

```
// klasa Prostokat, modyfikacje
Prostokat p;
try{
  p = new Prostokat(null, 1, -1);
  Console.WriteLine(p);
}
catch(ArgumentException e){
  Console.WriteLine(e.Message);
}

p = new Prostokat(new Punkt(1,1), 2, bokDrugi: 3);
Console.WriteLine(p);
p.LewyDolny = new Punkt(0,0);
p.BokDrugi = 5;
p.Rysuj();
p.BokDrugi = -1;
p.Rysuj();
```

```
Punkt nie może być null
R(P(1, 1), 2 x 3)
R(P(0, 0), 2 x 5) obwod=14.00, pole=10.00
R(P(0, 0), 2 x 0) obwod=4.00, pole=0.00
```

Odpowiedź: (system kar: 0, 0, 0, 1, 2, 5, 10, 20, ... %)

Zresetuj odpowiedź

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace FiguryLib
{

    public class Prostokat :Kwadrat
    {
        private int bokdwa;

        public int BokDrugi
        {
            get
            {
                return bokdwa;
            }
            set
            {
                if (value < 0)
                    bokdwa = 0;
                    else {
                        bokdwa = value;
                    }
            }
        }
        public Prostokat(Punkt DOLNY, int Bok, int BokDrugi) : base(DOLNY, Bok)
        {
            if (BokDrugi < 0)
                bokdwa = 0;
            else
            {
                bokdwa = BokDrugi;
            }
            DefaultColor = ConsoleColor.Yellow;
        }
        public Prostokat() : this(new Punkt(), 1, 2) { }
        public override string ToString() => $"R({LewyDolny}, {Bok} x {BokDrugi})";


        public new void Rysuj()
        {
            Console.ResetColor();
            Console.ForegroundColor = this.DefaultColor;
            Console.WriteLine($"R({LewyDolny}, {Bok} x {BokDrugi}) obwod={Obwod:F2}, pole={Pole:F2}"
            Console.ResetColor();
        }


        public new double Pole => Bok * BokDrugi;

        public new double Obwod => Bok * 2 + BokDrugi *2;
```

```
56          public new double Dlugosc => Bok * 2 + BokDrugi * 2;
57
58
59      }
60
61
```

	Test	Oczekiwane	Otrzymar

✔	`// klasa Kwadrat, dziedziczenie z Figura,`
	`// implementacja IMierzalna2D`
	`var k = new Kwadrat();`
	`Console.WriteLine(k is FiguryLib.Figura);`
	`Console.WriteLine(k is FiguryLib.IMierzalna2D);`
		`True`
		`True`
			`True`
			`True`

✔	`// klasa Kwadrat, konstruktor domyślny`
	`Kwadrat k = new Kwadrat();`
	`Console.WriteLine(k);`
	`k.Rysuj();`
		`K(P(0, 0), 1)`
		`K(P(0, 0), 1) obwod=4.00, pole=1.00`
			`K(P(0, 0)`
			`K(P(0, 0)`

✔	`// klasa Kwadrat, konstruktor dwuargumentowy`
	`Kwadrat k = new Kwadrat(new Punkt(1,2), 2);`
	`Console.WriteLine(k);`
	`k.Rysuj();`
		`K(P(1, 2), 2)`
		`K(P(1, 2), 2) obwod=8.00, pole=4.00`
			`K(P(1, 2)`
			`K(P(1, 2)`

✔	`// klasa Kwadrat, modyfikacje`
	`Kwadrat k;`
	`try{`
	`  k = new Kwadrat(null, -1);`
	`  Console.WriteLine(k);`
	`}`
	`catch(ArgumentException e){`
	`  Console.WriteLine(e.Message);`
	`}`
	
	`k = new Kwadrat(new Punkt(1,1),1);`
	`Console.WriteLine(k);`
	`k.LewyDolny = new Punkt(0,0);`
	`k.Bok = 5;`
	`k.Rysuj();`
	`k.Bok = -1;`
	`k.Rysuj();`
		`Punkt nie może być null`
		`K(P(1, 1), 1)`
		`K(P(0, 0), 5) obwod=20.00, pole=25.00`
		`K(P(0, 0), 0) obwod=0.00, pole=0.00`
			`Punkt nie`
			`K(P(1, 1)`
			`K(P(0, 0)`
			`K(P(0, 0)`

✔	`// klasa Prostokat, dziedziczenie z Figura,`
	`// implementacja IMierzalna2D`
	`var p = new Prostokat();`
	`Console.WriteLine(p is FiguryLib.Figura);`
	`Console.WriteLine(p is FiguryLib.IMierzalna2D);`
		`True`
		`True`
			`True`
			`True`

✔	`// klasa Prostokat, konstruktor domyślny`
	`Prostokat p = new Prostokat();`
	`Console.WriteLine(p);`
	`p.Rysuj();`
		`R(P(0, 0), 1 x 2)`
		`R(P(0, 0), 1 x 2) obwod=6.00, pole=2.00`
			`R(P(0, 0)`
			`R(P(0, 0)`

✘	`// klasa Prostokat, konstruktor dwuargumentowy`
	`Prostokat p = new Prostokat(new Punkt(1,2), bok: 2, bokDrugi: 3);`
	`Console.WriteLine(p);`
	`p.Rysuj();`
		`R(P(1, 2), 2 x 3)`
		`R(P(1, 2), 2 x 3) obwod=10.00, pole=6.00`
			`Compilati`
			
			`***Błąd**`
			`./Program`
			`./Solutio`
			`./Program`
			`./Solutio`
			`** Compil`

Testowanie zostało przerwane z powodu błędu.
Twój kod musi przejść wszystkie testy, aby uzyskać jakiekolwiek punkty. Spróbuj ponownie.

Pokaż różnice

Niepoprawnie

Punkty dla tej odpowiedzi: 0,00/5,00.

Nawigacja w teście

Pokaż wszystkie pytania na stronie

Zakończ przegląd