IK

Inżynieria Komputerowa

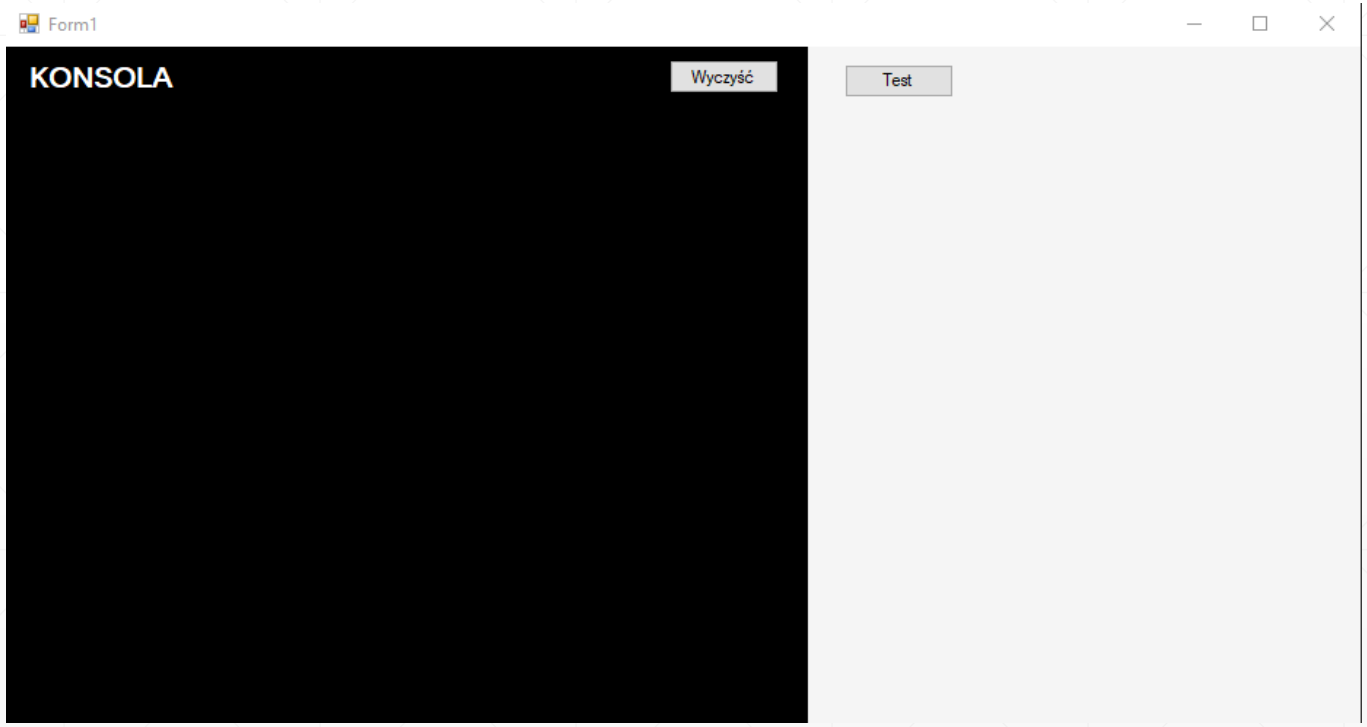
mgr inż. Patryk Kaczmarek
patryk.kaczmarek@ansleszno.pl

Programowanie obiektowe

W solucji **PK_InzynieraKomputerowa** znajduje się projekt **PK_IK_Lab03**.

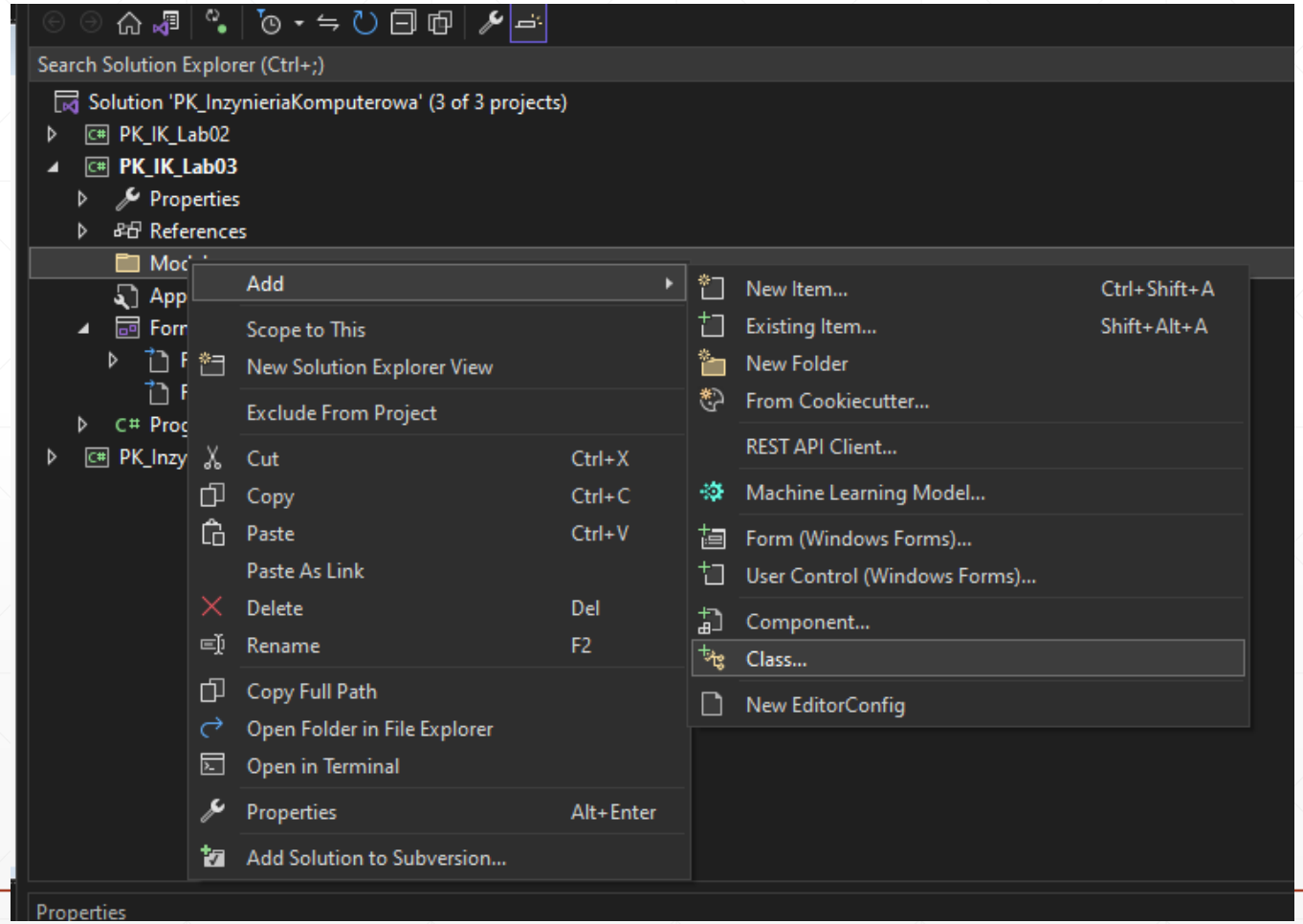
Jest to pusty projekt w którym będziemy tworzyć program wykorzystujący tworzenie nowych klas, obiektów, dziedziczenie.

Przetestuj działanie programu.



Programowanie obiektowe – ZADANIE 1

Stwórz w folderze Model klasę o nazwie **Figura**



Programowanie obiektowe – ZADANIE 1

Stwórz w folderze Model klasę o nazwie **Figura.** Dodaj następujące pola do klasy:

```
namespace PK_IK_Lab03.Model
{
    0 references
    public class Figura
    {
        private string _Nazwa = "Figura";
        0 references
        public string Nazwa
        {
            get { return _Nazwa; }
            set { _Nazwa = value; }
        }
    }
}
```

Programowanie obiektowe – ZADANIE 1

Stwórz obiekt figura i wyświetl nazwę w konsoli.

(Można wykorzystać Button Test, kliknij dwa razy w przycisk – automatycznie otworzy się funkcja, wprowadź do niej poniższy kod)

```
1 reference
private void button1_Click(object sender, EventArgs e)
{
    string napis = "Przykładowy tekst wyświetlany w konsoli" + Environment.NewLine;
    //napis += "wyświetlany w dwóch liniach.";

    Figura figura = new Figura();
    napis = $"Nazwa figury [{nameof(Figura)}]: {figura.Nazwa}";
    ZapiszWynikKonsola(napis);
```

Uruchom program i sprawdź wynik w konsoli

Programowanie obiektowe – ZADANIE 2

Stwórz w folderze Model klasę o nazwie **Prostokat**, która dziedziczy klasę figura**.** Dodaj następujące pola do klasy:

```
namespace PK_IK_Lab03.Model
{
    1 reference
    public class Prostokat : Figura
    {
        0 references
        public Prostokat()
        {
            Nazwa = "Prostokat";
        }
        int bokA;
        int bokB;

        0 references
        public void UstawBok(int A, int B)
        {
            bokA = A;
            bokB = B;
        }

        0 references
        public int Pole() { return bokA * bokB; }
        0 references
        public int Obwod() { return (2 * bokA) + (2 * bokB); }

    }
}
```

Programowanie obiektowe – ZADANIE 2

Stwórz obiekt prostokąt i wyświetl nazwę w konsoli.

```
Prostokat prostokat = new Prostokat();
napis = $"Nazwa figury [{nameof(Prostokat)}]: {prostokat.Nazwa}";
ZapiszWynikKonsola(napis);
```

Uruchom program i sprawdź wynik w konsoli

Programowanie obiektowe – ZADANIE 3

Stwórz w folderze Model klasę o nazwie **Kwadrat**, która dziedziczy klasę Prostokat**.** Dodaj następujące pola do klasy:

```
namespace PK_IK_Lab03.Model
{
    1 reference
    public class Kwadrat : Prostokat
    {
        0 references
        public Kwadrat()
        {
            Nazwa = "Kwadrat";
        }
        0 references
        public void UstawBok(int a)
        {
            UstawBok(a, a);
        }
    }
}
```

Programowanie obiektowe – ZADANIE 3

Stwórz obiekt kwadrat i wyświetl nazwę w konsoli.

```
Kwadrat kwadrat = new Kwadrat();
napis = $"Nazwa figury [{nameof(Kwadrat)}]: {kwadrat.Nazwa}";
ZapiszWynikKonsola(napis);
```

Uruchom program i sprawdź wynik w konsoli

Programowanie obiektowe – ZADANIE 4

Zmodyfikuj poprzednie zadanie – obiekty figur, uzupełniając je o funkcje ustawiające długości boków (mogą być na sztywno), wyświetl pole i obwód figur.

Programowanie obiektowe – ZADANIE 5

Dodaj do odpowiednich klas funkcję, która wyświetli nazwę figury oraz długość boków.

Zwróć uwagę na kwadrat – po co wyświetlać 2 takie same boki. Możemy zmienić funkcję dla kwadratu ?

```
0 references
public string InfoFigura()
{
    return $"{Nazwa} długość boku A: {BokA} B: {BokB}";
}
```

Programowanie obiektowe – ZADANIE 6

Zmodyfikuj klasę Figura:

```
4 references
public class Figura
{
    private string _Nazwa = "Figura";
    8 references
    public string Nazwa
    {
        get { return _Nazwa; }
        set { _Nazwa = value; }
    }

    0 references
    public virtual string InfoFigura() {
        return $"{Nazwa}";
    }
}
```

Programowanie obiektowe – ZADANIE 7

Zmodyfikuj klasę Prostokat:

```
3 references
public class Prostokat : Figura
{
    1 reference
    public Prostokat() {
        Nazwa = "Prostokat";
    }
    int bokA;
    int bokB;

    1 reference
    public void UstawBok(int A, int B) {
        bokA = A;
        bokB = B;
    }

    0 references
    public int Pole() { return bokA * bokB; }
    0 references
    public int Obwod() { return (2 * bokA) + (2 * bokB); }

    1 reference
    public override string InfoFigura() {
        return $"{Nazwa} długość boku A: {bokA} B: {bokB}";
    }

}
```

Programowanie obiektowe – ZADANIE 8

Zmodyfikuj klasę Kwadrat:

Jak zrobić aby wyświetliła
Się tylko informacja o 1 boku?

```
namespace PK_IK_Lab03.Model
{
    3 references
    public class Kwadrat : Prostokat
    {
        1 reference
        public Kwadrat()
        {
            Nazwa = "Kwadrat";
        }
        1 reference
        public void UstawBok(int a)
        {
            UstawBok(a, a);
        }

        5 references
        public override string InfoFigura()
        {
            return base.InfoFigura();
        }
    }
}
```

Programowanie obiektowe – ZADANIE 9

Zmodyfikuj klasę Prostokąt o 2 Funkcję pobierania długości Boków

```
5 references
public class Prostokat : Figura
{
    1 reference
    public Prostokat() {
        Nazwa = "Prostokat";
    }
    int bokA;
    int bokB;

    1 reference
    public void UstawBok(int A, int B) {
        bokA = A;
        bokB = B;
    }

    0 references
    public int Pole() { return bokA * bokB; }
    0 references
    public int Obwod() { return (2 * bokA) + (2 * bokB); }

    0 references
    public int GetBokA => bokA;
    0 references
    public int GetBokB => bokB;

    1 reference
    public override string InfoFigura() {
        return $"{Nazwa} długość boku A: {bokA} B: {bokB}";
    }

}
```

Następnie popraw funkcję InfoFigura w Kwadracie.

Programowanie obiektowe – ZADANIE 10

Zamieńmy teraz kolejności dziedziczenia:

Stwórz klasę **FiguraNowa**,

```
0 references
public class FiguraNowa
{
    private string _Nazwa = "Figura";
    1 reference
    public string Nazwa
    {
        get { return _Nazwa; }
        set { _Nazwa = value; }
    }

    0 references
    public virtual string InfoFigura() => Nazwa;
    0 references
    public virtual int Pole() => throw new NotImplementedException("Brak funkcji");
    0 references
    public virtual int Obwod() => throw new NotImplementedException("Brak funkcji");
}
```

Programowanie obiektowe – ZADANIE 11

Zamieńmy teraz kolejności dziedziczenia:

Stwórz klasę **KwadratNowy**, który dziedziczy klasę **FiguraNowa**

```
2 references
public class KwadratNowy : FiguraNowa
{
    0 references
    public KwadratNowy() { }
    0 references
    public KwadratNowy(int bok) {
        UstawBokA(bok);
    }

    private int bokA;
    1 reference
    public void UstawBokA(int A) { bokA = A; }
    0 references
    public int GetBokA() { return bokA; }

    1 reference
    public override string InfoFigura() { return $"Kwadrat długość boku A: {bokA}"; }
    1 reference
    public override int Obwod() { return 4 * bokA; }
    1 reference
    public override int Pole() { return bokA * bokA; }
}
```

Programowanie obiektowe – ZADANIE 12

Zamieńmy teraz kolejności dziedziczenia:

Stwórz klasę **ProstokatNowy**, który dziedziczy klasę **KwadratNowy**

```
2 references
public class ProstokatNowy : KwadratNowy
{
    0 references
    public ProstokatNowy() { }
    0 references
    public ProstokatNowy(int bokA, int bokB) {
        UstawBoki(bokA, bokB);
    }

    private int bokB;
    1 reference
    public void UstawBoki(int A, int B) {
        bokB = B;
        UstawBokA(A);
    }
    2 references
    public override string InfoFigura() { return $"Prostokąt długość boku A: {GetBokA()} boku B: {bokB}"; }
    2 references
    public override int Obwod() { return (2 * bokB) + (2 * GetBokA()); }
    2 references
    public override int Pole() { return bokB * GetBokA(); }
}
```

Programowanie obiektowe – ZADANIE 13

Popraw klasę figura:

```
9 references
public class FiguraNowa
{
    private string _Nazwa = "Figura";
    1 reference
    public string Nazwa
    {
        get { return _Nazwa; }
        set { _Nazwa = value; }
    }

    4 references
    public virtual string InfoFigura() => Nazwa;
    4 references
    public virtual double Pole() => throw new NotImplementedException("Brak funkcji");
    4 references
    public virtual double Obwod() => throw new NotImplementedException("Brak funkcji");
}
```

Programowanie obiektowe – ZADANIE 13

Popraw również klasy KwadratNowy i ProstokatNowy

```
5 references
public class KwadratNowy : FiguraNowa
{
    0 references
    public KwadratNowy() { }
    2 references
    public KwadratNowy(int bok) {
        UstawBokA(bok);
    }

    private int bokA;
    2 references
    public void UstawBokA(int A) { bokA = A; }
    3 references
    public int GetBokA() { return bokA; }

    3 references
    public override string InfoFigura() { return $"Kwadrat długość boku
    3 references
    public override double Obwod() { return 4 * bokA; }
    3 references
    public override double Pole() { return bokA * bokA; }
}
```

```
public class ProstokatNowy : KwadratNowy
{
    0 references
    public ProstokatNowy() { }
    2 references
    public ProstokatNowy(int bokA, int bokB) {
        UstawBoki(bokA, bokB);
    }

    private int bokB;
    1 reference
    public void UstawBoki(int A, int B) {
        bokB = B;
        UstawBokA(A);
    }
    3 references
    public override string InfoFigura() { return $"Prostokąt długość boku A: {GetBokA()} boku B: {bokB}"; }
    3 references
    public override double Obwod() { return (2 * bokB) + (2 * GetBokA()); }
    3 references
    public override double Pole() { return bokB * GetBokA(); }
}
```

Programowanie obiektowe – ZADANIE 13

Dodaj klasę KoloNowe

```
2 references
public class KoloNowe : FiguraNowa
{
    0 references
    public KoloNowe() { }
    0 references
    public KoloNowe(int promien) {
        UstawPromien(promien);
    }

    private int promien;
    1 reference
    public void UstawPromien(int r) { promien = r; }

    2 references
    public override string InfoFigura() { return $"Pole promień R: {promien}"; }
    2 references
    public override double Obwod() { return 2 * System.Math.PI * promien; }
    2 references
    public override double Pole() { return System.Math.PI * promien * promien; }
}
```

Programowanie obiektowe – ZADANIE 13

Sprawdź działanie następującego kodu i napisz odpowiednie wnioski

```
string napis = "Przykładowy tekst wyświetlany w konsoli" + Environment.NewLine;

List<FiguraNowa> ListaFigury = new List<FiguraNowa>();

FiguraNowa kwadrat = new KwadratNowy(5);
ListaFigury.Add(kwadrat);
FiguraNowa kwadrat2 = new KwadratNowy(3);
ListaFigury.Add(kwadrat2);
FiguraNowa prostokat = new ProstokatNowy(5,5);
ListaFigury.Add(prostokat);
FiguraNowa prostokat2 = new ProstokatNowy(3,8);
ListaFigury.Add(prostokat2);
FiguraNowa kolo = new KoloNowe(3);
ListaFigury.Add(kolo);
FiguraNowa kolo2 = new KoloNowe(5);
ListaFigury.Add(kolo2);


foreach (FiguraNowa figura in ListaFigury) {

    napis += $"{figura.InfoFigura()}";
    napis += Environment.NewLine;
    napis += $"Pole: {figura.Pole()}";
    napis += Environment.NewLine;
    napis += $"Obwód: {figura.Obwod()}";
    napis += Environment.NewLine;
    napis += Environment.NewLine;
}

ZapiszWynikKonsola(napis);
```