Laboratorium 5

Programowanie w C# (semestr zimowy 2019/2020)

Temat: Programowanie obiektowe (łagodne wprowadzenie)

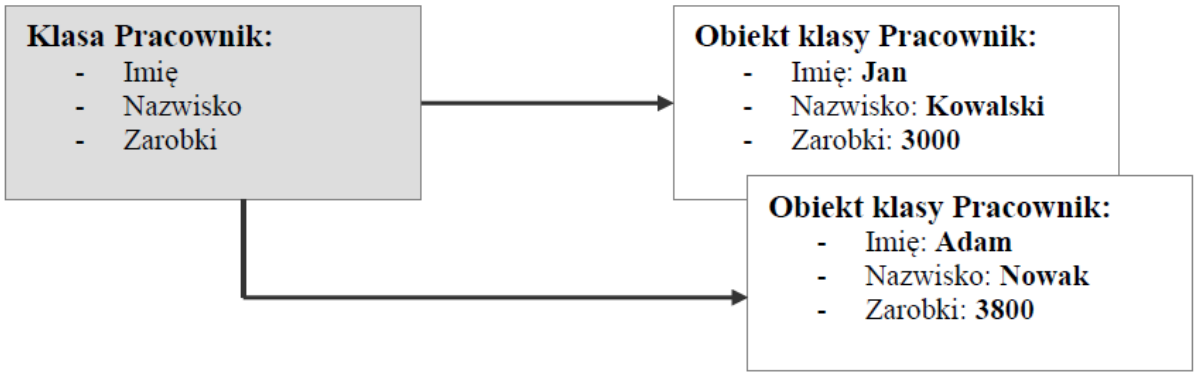
Prowadzący: Piotr Pięta

1. Poruszone zagadnienia

Klasa to typ danych. Definiując własną klasę (nowy typ danych) definiujemy jej składniki, którymi są pola i metody. Metody (funkcje klasy) realizują zwykle różne operacje na swoich polach. Definiowanie klasy możemy utożsamiać z tworzeniem pól formularza, którym później będziemy przypisywać wartości.

Obiekt jest zmienną typu klasy (*instancją* klasy). Tworząc obiekt danej klasy, zwykle polom klasy przypisujemy konkretne wartości, czyli wypełniamy, przygotowany wcześniej formularz, konkretnymi danymi

1.1 Klasa a obiekt



Paradygmat *obiektowy* (**enkapsulacja** – dziedziczenie – polimorfizm)

Podstawowa jednostka enkapsulacji (hermetyzacji) to klasa. Zalety: łączenie danych z kodem, kontrola dostępu do składników (klasy)

1.2 Składnia klasy

```
[Atrybuty][Modyfikatory] class Nazwa [lista elementów bazowych]
{
        //Ciało klasy
}
```

1.2.1 Modyfikatory dostępu do klasy

Wyróżnia się pięć modyfikatorów dostępu do klasy:
public, private, protected, internal oraz protected internal.

Przy opisie modyfikatorów używa się terminu zestaw (ang. assembly) – wykonywalny kod programów w środowisku .NET jest umieszczany w tzw. zestawach (inne polskie nazwy to kompilat, podzespół, pakiet).

Modyfikator **public** oznacza, że klasa jest **dostępna z poziomu każdego zestawu**. Modyfikatory **private** oraz **protected** dotyczą tylko klas zagnieżdżonych – private oznacza, że dostęp ogranicza się do klasy zawierającej (czyli tej, w której dana klasa jest zagnieżdżona), a protected ponadto daje dostęp klasom potomnym klasy zawierającej.

W przypadku modyfikatora internal dostęp dotyczy klas z tego samego zestawu – jest to domyślny modyfikator klasy. Modyfikator **protected internal** oznacza, że dostęp ogranicza się do danego zestawu oraz klas potomnych względem klasy zawierającej.

1.2.2 Modyfikatory rodzaju klasy

static - modyfikator dla klasy statycznej, zawierającej wyłącznie składowe statyczne
abstract – modyfikator dla klasy abstrakcyjnej. Klasa abstrakcyjna wykorzystywana jest jedynie w postaci klasy bazowej (inne klasy po niej dziedziczą) i nie można na jej
podstawie tworzyć obiektów
sealed – modyfikator dla klasy, która nie może pełnić roli klasy bazowej

Dotychczas, najczęściej korzystaliśmy z modyfikatora static

1.3 Proste przykłady

```
public class Pracownik
{
      public string nazwisko;
      private double zarobki;
      public static int liczbaPracownikow; //nie dotyczy
      //konkretnego obiektu - statyczna
      public const double etat = 1.0; // stała (const)
}
```

```
/************************************************/

public class Pracownik
```

```
{
	public string nazwisko;
	private double zarobki;

	//
	public Pracownik(string n, double z) // konstruktor
	{
		nazwisko = n;
		zarobki = z;
	}

	public void PokazPracownika() // metoda
	{
		Console.WriteLine("{0,-15} {1,10}", nazwisko,
	zarobki);
	}

}


class Program
{
	static void Main(string[] args)
	{
		Pracownik p1 = new Pracownik("Kowalski", 1000);
		p1.PokazPracownika();
		Console.ReadKey();
	}
}
```

Operator new() tworzy obiekt i wywołuje konstruktor (patrz: deklaracja zmiennych – poprzednie laboratoria, o typach)

```
class Program
{
	static void Main(string[] args)
	{
		Pracownik p1 = new Pracownik("Kowalski", 1000);
		p1.nazwisko = "Nowak"; // zmiana wartości pola obiektu
		p1.PokazPracownika();
		Console.ReadKey();
	}
}
```

Czy zmiana pola zarobki jest możliwa?

```
	p1.zarobki = 8000;
```

Proszę zapoznać się z materiałami w odnośnikach:

[1] https://docs.microsoft.com/pl-pl/dotnet/csharp/programming-guide/classes-and-structs/classes

[2] https://docs.microsoft.com/pl-pl/dotnet/csharp/tour-of-csharp/classes-and-objects

[3] https://www.tutorialsteacher.com/csharp/csharp-class

[4] https://www.w3schools.com/cs/cs_classes.asp

[5] https://www.plukasiewicz.net/CSharp_dla_poczatkujacych/Klasy

2. Zadania do samodzielnego rozwiązania

2.1 Napisz program, który tworzy klasę *Prostokat*, zawierającą dwie prywatne dane składowe: *dlugosc*, *szerokosc*, dwie prywatne metody: *powierzchnia()*, *obwod()* oraz metodę publiczną – *Prezentuj()* (która wyświetla powierzchnię i obwód prostokąta) i konstruktor (inicjalizujący). W metodzie *Main()* zdefiniuj obiekt i uruchom dla niego metodę *Prezentuj()*.

2.2 Napisz program, który oblicza pierwiastki równania kwadratowego z wykorzystaniem instrukcji wyboru switch-case.
Klasa powinna zawierać trzy metody: wczytującą dane (obsłużenie sytuacji, gdy a=0, liczby a, b, c – rzeczywiste, wprowadzone z klawiatury), przetwarzająca te dane (odpowiedzialna za wykonanie niezbędnych obliczeń), a także wyświetlająca wynik na ekranie monitora.

2.3 Przygotuj program, który w macierzy 10x10 umieszcza *losowo* na przekątnej macierzy liczby od 0-9 (pozostałe komórki macierzy wypełnia zerami).
Program ponadto powinien zawierać metody pozwalające na wyświetlenie macierzy, wyświetlenie (i obliczenie) sumy liczb znajdujących się na przekątnej macierzy.

Random rand = new Random(); //generowanie liczby pseudolos.
matrix[a,b] = Math.Round(9*(rand.NextDouble())); //wpisywanie liczb pseudolosowych od 0-9 w macierzy (wykonywane w pętli)

2.4 Napisz program (przy użyciu klas oraz tablicy) pozwalający na posortowanie *n* liczb. Program powinien zawierać metody pozwalające na wyświetlenie danych, sortowanie (dowolnym sposobem, np. sortowanie bąbelkowe), oraz wczytanie danych)