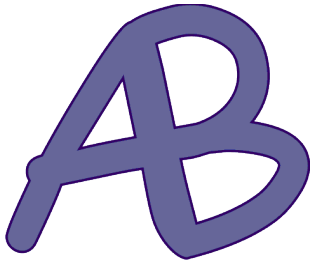

# Wykład
## PASCAL

## Wstęp do programowania obiektowego

## Programowanie obiektowe - definicja

**Programowanie obiektowe** – paradygmat programowania, w którym programy definiuje się za pomocą obiektów – elementów łączących *stan* (czyli dane, nazywane najczęściej polami) i *zachowanie* (czyli procedury, tu: metody).

Obiektowy program komputerowy wyrażony jest jako zbiór takich obiektów, komunikujących się pomiędzy sobą w celu wykonywania zadań.

Źródło: Wikipedia

# Definicja typu obiektowego

```
Type
  Nazwa = Object
            Private
              {deklaracje pól obiektu}
            Public
              {deklaracje pól obiektu}
          End;
```

- Typ obiektowy składa się z dwóch sekcji: prywatnej i publicznej.
- Obie sekcje mogą zawierać deklaracje zmiennych opisujących obiekt lub metod, które będą wykonywane na obiekcie.
- Domyślnie wszystkie deklaracje są publiczne.

# Definicja typu obiektowego

Przykład obiektu:

```
type
  trojkat =  object
    a, b, c: real;
    function obwod:real;
  end;

function trojkat.obwod:real;
begin
  obwod:=a+b+c;
end;
 var
  t: trojkat;
begin
  readln(t.a, t.b, t.c);
  writeln('Obwod=', t.obwod);
. . . .
```

# Definicja typu obiektowego

- W deklaracji typu obiektowego w pierwszej kolejności wymieniane są pola obiektu, a w drugiej metody.
- Każda metoda deklarowana jest w typie obiektowym w postaci zapowiedzi zawierającej jedynie nagłówek metody.
- Kompletna deklaracja metody podawana jest odrębnie poza deklaracją typu obiektowego. Nazwa metody jest uzupełniana prefiksem – nazwą typu obiektowego. Lista parametrów w nagłówku nie może być zmieniona w stosunku do poprzedzającej zapowiedzi. Może być jedynie powtórzona bez zmian lub opuszczona w całości.
- Typy obiektowe mogą być deklarowane tylko globalnie, w programie lub w module lecz nie wewnątrz procedury.

# Definicja typu obiektowego

- Zmienne obiektowe muszą być deklarowane przez wykorzystanie identyfikatora typu (zdefiniowanego zawsze globalnie), a nie opisu typu.
- Metoda należy do obiektu i w związku z tym posiada dostęp do pól i innych metod obiektu bezpośrednio. W treści procedur można korzystać z nazw pól bez konieczności dopisywania prefiksów lub umieszczania ich w parametrach metod - obydwa rozwiązania są nawet zabronione. Np.:

```
procedure. Trojkat obwod;
begin
  Trojkat := a + b + c;
end;
```

# Definicja typu obiektowego

- Poza definicją typu obiektowego i definicjami związanych z nim metod przy odnoszeniu się do pól lub metod obiektu należy używać nazw dwuczęściowych:

  **<nazwa zmiennej obiektowej> . < nazwa pola lub metody>**

- Podobnie jak w przypadku rekordów możliwe jest uproszczenie dostępu do składowych obiektu przy wykorzystaniu instrukcji with.
- Zmienne lokalne i parametry metody nie mogą mieć tych samych nazw co pola obiektu.
- W metodach można wywoływać inne metody obiektu nawet te zdefiniowane szczegółowo później (bez prefiksu, wystarczy nazwa metody)

# Hermetyzacja

Czyli ukrywanie implementacji (enkapsulacja). Zapewnia, że obiekt nie może zmieniać stanu wewnętrznego innych obiektów w nieoczekiwany sposób.

Tylko własne metody obiektu są uprawnione do zmiany jego stanu.

Każdy typ obiektu prezentuje innym obiektom swój interfejs, który określa dopuszczalne metody współpracy.

Źródło: Wikipedia

# Hermetyzacja

Przykład metod dostępowych:

```
procedure trojkat.UstawDane(x, y, z: Real);
 begin
   a := x; b := y; c:= z;
 end;

procedure trojkat.ZwrocDane(var x, y, z: Real);
 begin
   x := a; y := b; z:= a;
 end;

function BokA: real;
 begin
   BokA:=a;
 end;
```

# Dziedziczenie

- Dziedziczenie umożliwia definiowanie i tworzenie specjalizowanych obiektów na podstawie bardziej ogólnych.
- Jest ważną cechą programowania obiektowego pozwalającą na łatwą rozbudowę, a także modyfikację opracowanych obiektów.
- Istotne jest to, że rozbudowa obiektu nie wymaga ingerencji w już opracowany kod programu. Co więcej, wystarczy by moduły zawierające definicje modyfikowanych typów obiektowych były dostępne w postaci skompilowanej.

# Dziedziczenie

```
type trojkat2 = object(Trojkat)
                  function wysokoscA: real;
                  function Rwpisany: real;
                end;

function trojkat2.wysokoscA: real;
  begin
    wysokoscA:=2*pole/BokA;
  end;

function trojkat2.Rwpisany: real;
  begin
    Rwpisany:=4*pole/obwod;
  end;
```

# Literatura:

**W prezentacji wykorzystano przykłady i fragmenty:**

- R. Jarża, *Turbo Pascal. Szkoła programowania*, Wydawnictwo Robomatic 2000. *(dostępne w bibliotece uczelni)*
- http://pascal.kurs-programowania.pl
- http://4programmers.net
- T. M. Sadowski, *Turbo Pascal. Programowanie*, Helion 1996.