Table of Contents

[Wstęp Do Programowania Obiektowego: 2](#_Toc62931917)

[Klasa A Obiekt: 2](#_Toc62931918)

[Istota Programowania Obiektowego: 2](#_Toc62931919)

[Tworzenie Klasy: 3](#_Toc62931920)

[Metody, Atrybuty I Konstruktory Klas: 8](#_Toc62931921)

[Metoda Klasy (Czym Jest I Do Czego Służy): 8](#_Toc62931922)

[Deklarowanie Metody: 9](#_Toc62931923)

[Atrybuty Klasy: 12](#_Toc62931924)

[Definiowanie Atrybutów: 13](#_Toc62931925)

[Konstruktory: 13](#_Toc62931926)

[Deklaracja Konstruktora: 15](#_Toc62931927)

# Wstęp Do Programowania Obiektowego:

## Klasa A Obiekt:

Klasa ­– Opis nowej struktury danych

Obiekt – Struktura danych stworzona zgodnie z opisem klasy

## Istota Programowania Obiektowego:

Programowanie obiektowe umożliwia opisanie cech oraz funkcjonalności elementów, na przykład drukarki, monitora, klawiatury, samochodu itd. za pomocą abstrakcyjnego pojęcia zwanego klasą, które ułatwia implementację danego problemu.

Dzięki podejściu obiektowemu możliwy jest logiczny podział projektu na komponenty (obiekty), w celu ułatwienia implementacji rozwiązania a w szczególności ułatwienia rozbudowy projektu.

## Tworzenie Klasy:

W celu utworzenia klasy należy umieścić poniższy kod, w tym przypadku będzie to klasa BasicUser:



*Rysunek 1 utworzenie klasy*

W celu utworzenia obiektu klasy BasicUser należy umieścić poniższy kod:

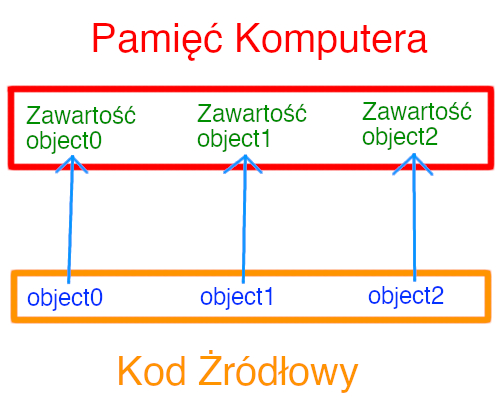


*Rysunek 2 utworzenie Obiektu Klasy*

Obiekt klasy (typu) BasicUser jest tworzony i umieszczany w zasięgu metody Main.

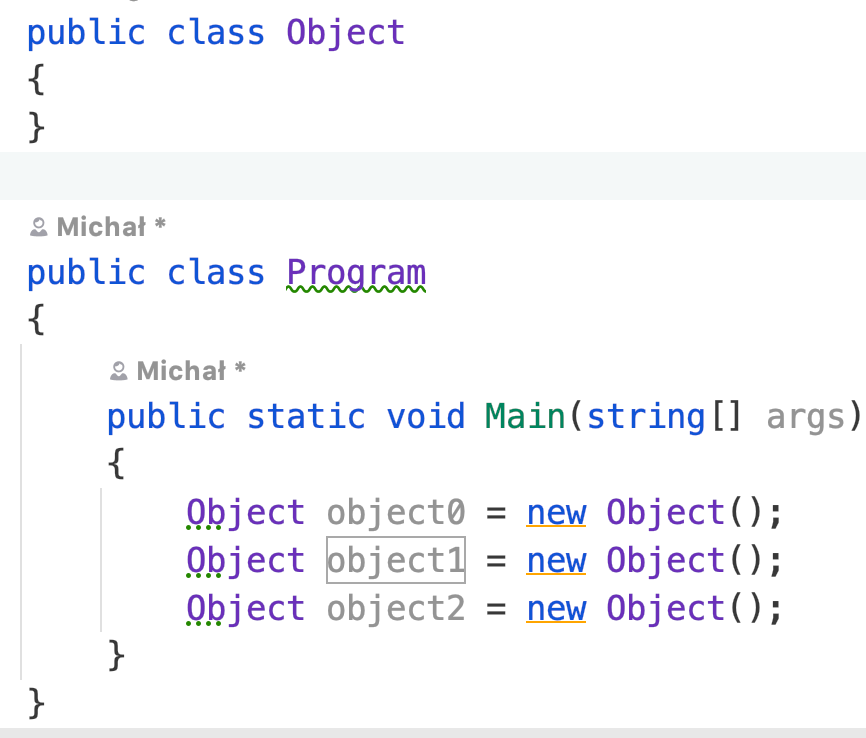
Słowo kluczowe **namespace** będzie omawianie w następnych rozdziałach, tyczy się to także modyfikatora dostępu **public**, słowa kluczowego **static** oraz wyjaśnienia pojęcia **metod** w programowaniu obiektowym.

Mechanizm Generowania Obiektów Klas:



*Rysunek 3 kod źródłowy a pamięć komputera*

Możliwe jest utworzenie kilku obiektów tej samej klasy. Istotny jest fakt, że każdy obiekt jest niezależny względem pozostałych obiektów, które powstały i tych, które mogą później zostać zdefiniowane.

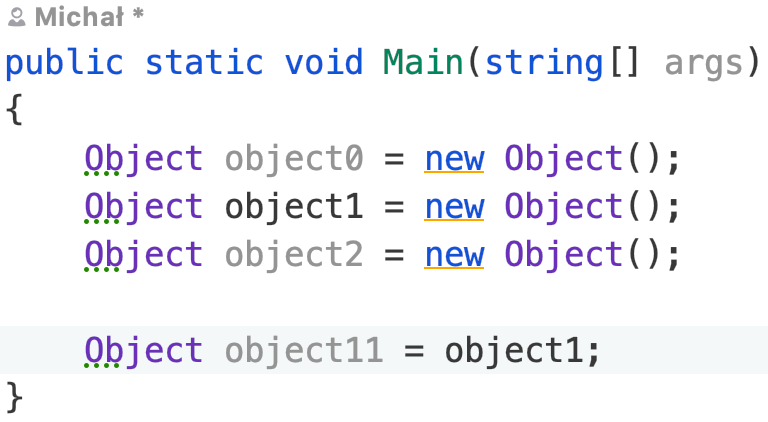


*Rysunek 4 kod źródłowy Do rysunku3*

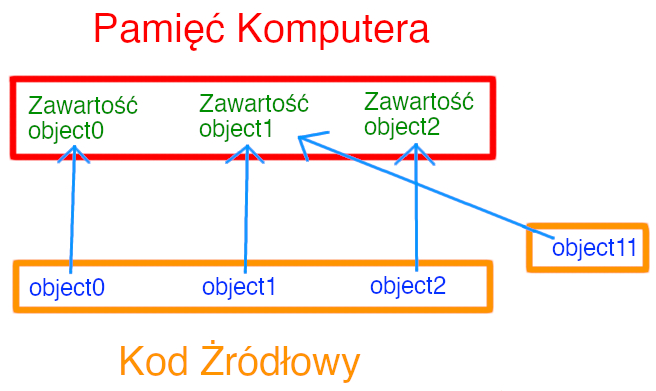
W procesie definiowania nowych obiektów ważnym aspektem jest umieszczenie słowa kluczowego **new**, które sprawia, że tworzona jest nowa, niezależna instancja obiektu określonej klasy.

Odwołując się do rysunku 3 można zauważyć, że każdy element zdefiniowany w kodzie jako object0, object1, object2 jest w swej istocie jedynie aliasem (referencją) obiektu, który jest umieszczony w określonym miejscu pamięci komputera.

Biorąc ten fakt pod uwagę poniższy kod powoduje, że istnieją dwa odniesienia do zawartości obiektu object1, co daje możliwość operowania na zawartości obiektu1 za pomocą dwóch aliasów, w tym przypadku object11.



*Rysunek 5 kod źródłowy z dodatkową referencją*



*Rysunek 6 Uwzględnienie nowej referencji*

# Metody, Atrybuty I Konstruktory Klas:

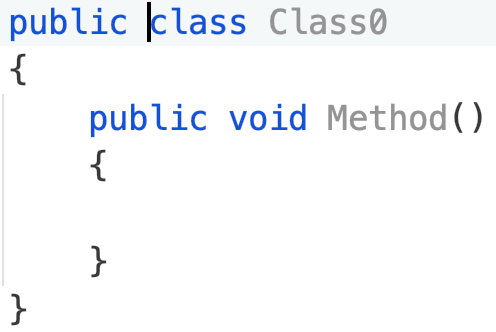
## Metoda Klasy (Czym Jest I Do Czego Służy):

Metoda jest odpowiednikiem funkcji w podejściu strukturalnym. Metoda wchodzi w skład klasy, umożliwiając operowanie na danych będących:

* Elementami składowymi klasy
* Danymi, przekazywanymi do metody za pomocą argumentów formalnych
* Danymi zaalokowanymi statycznie (dostępnych w obrębie całej aplikacji, bądź określonej przestrzeni nazw)
* Danymi zaalokowanymi w zasięgu bloku metody

## Deklarowanie Metody:

W celu deklaracji należy stworzyć klasę i zdefiniować metodę, tak jak na przykład ilustruje to poniższy kod:

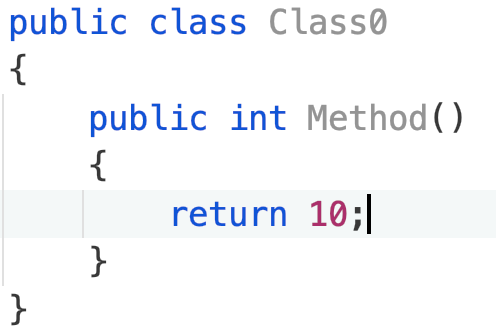


*Rysunek 7 Deklaracja Metody typu void*

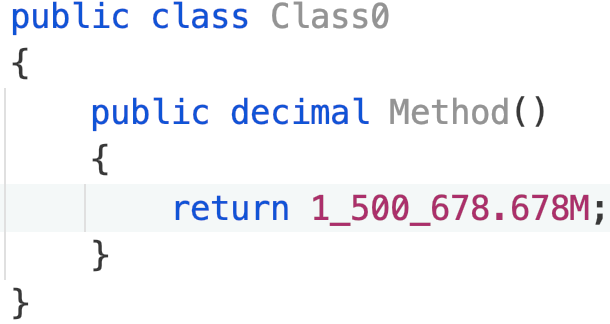
Słowo kluczowe **public** zostanie omówione w następnym rozdziale.

Istotne jest w tym przypadku słowo kluczowe **void**, które informuje, że metoda będzie przetwarzać dane, nie zwracając żadnej wartości po zakończeniu jej wykonywania w miejscu jej wywołania.

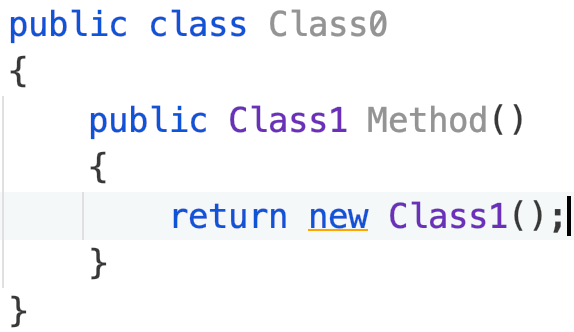
Deklaracja każdej metody może być opatrzona słowem kluczowym **void** bądź **typem prostym** bądź **typem obiektu**. W dwóch ostatnich przypadkach oznacza to, że po zakończeniu wykonywania metody, zwróci ona dane typu zdefiniowanego w deklaracji metody.



*RYSUNEK 8 DEKLARACJA METODY TYPU prostego (int)*



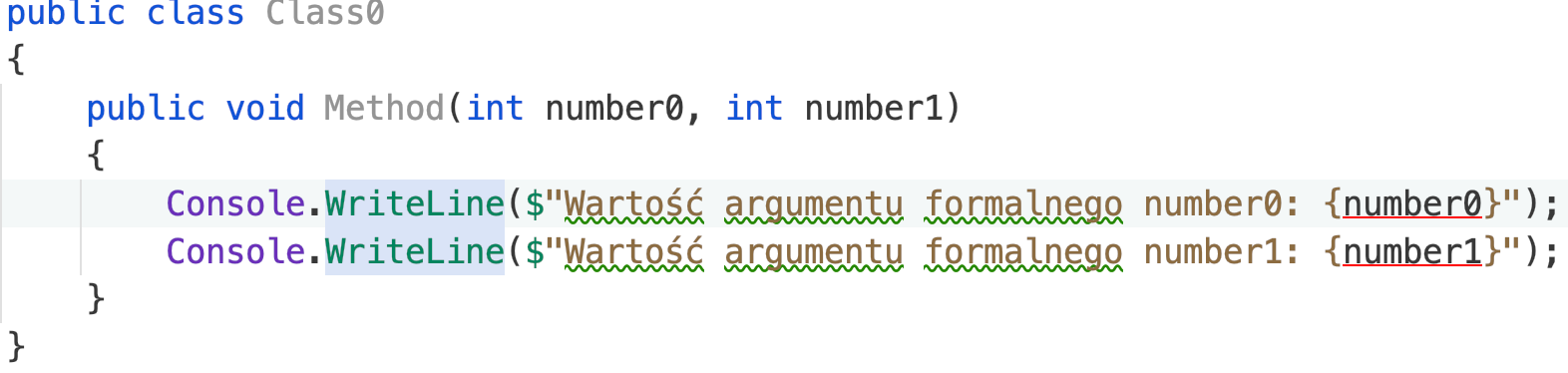
*RYSUNEK 9 deklaracja metody typu prostego (decimal)*



*Rysunek 10 DEKLARACJA METODY TYPU obiekt*

Każda metoda zwracająca wartość (**typ prosty**, bądź **typ obiektu**) musi posiadać instrukcję **return**, informującą jaka wartość, bądź obiekt będzie zwrócona/zwrócony.

Jak wcześniej opisano, metoda może przyjmować wartości zwane argumentami formalnymi. W celu zdefiniowania takiej metody można posłużyć się poniższym kodem:



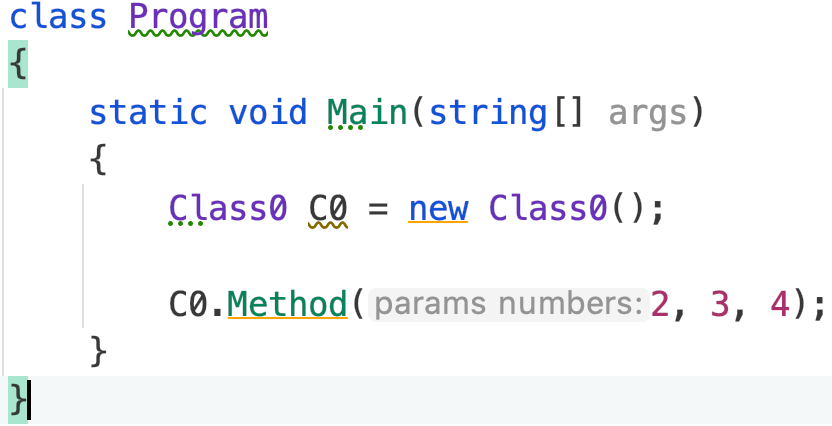
*RYSUNEK 11 DEKLARACJA METODY z parametrami formalnymi*

W celu wyświetlenia przekazanych wartości do metody w terminalu użyto metody **WriteLine** klasy **Console** i **interpolacji łańcuchów znakowych** ($”{} {} {}”).

Możliwe jest zdefiniowanie metody przyjmującej dowolną ilość danych danego typu. W celu zdefiniowania takiej metody należy skorzystać ze słowa kluczowego **params.**

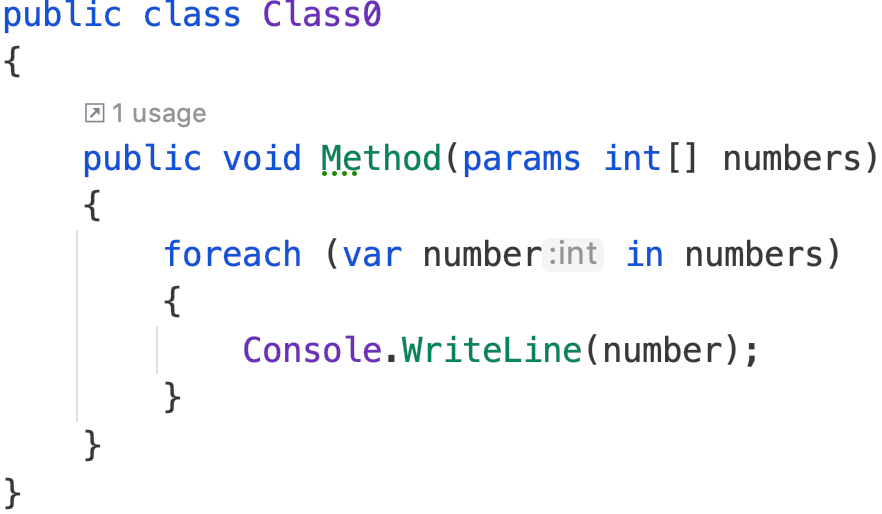
Przykładowa definicja metody korzystającej z wyżej opisywanego mechanizmu w celu wyświetlenia wszystkich przesłanych danych typu int została zilustrowana poniżej:

## Atrybuty Klasy:



*RYSUNEK 13 Przykładowe wykorzystanie metody ze zmienną ilością parametrów*

*RYSUNEK 12 DEKLARACJA METODY ze zmienną  
 ilością parametrów*

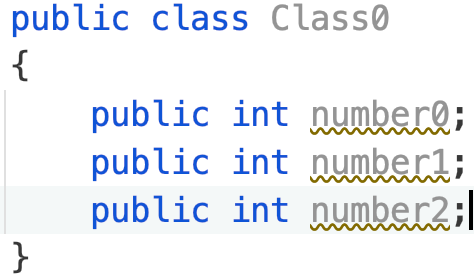


Atrybuty są elementami składowymi klas służącymi do przechowywania danych związanych z daną klasą. Dane te mogą być:

* Typem prostym (int, long, double, decimal itd.)
* Typem złożonym (obiekt, struktura danych itp.)

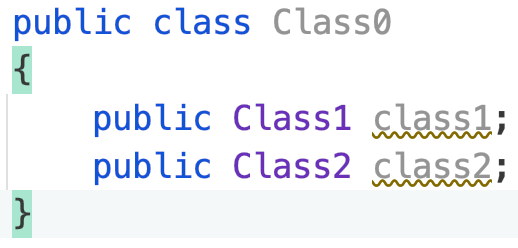
## Definiowanie Atrybutów:

W celu zdefiniowania atrybutów klasy można posłużyć się poniższym przykładem:



*RYSUNEK 14 DEKLARACJA atrybutów klasy*

## Konstruktory:



*RYSUNEK 15 DEKLARACJA atrybutów klasy jako obiektów*

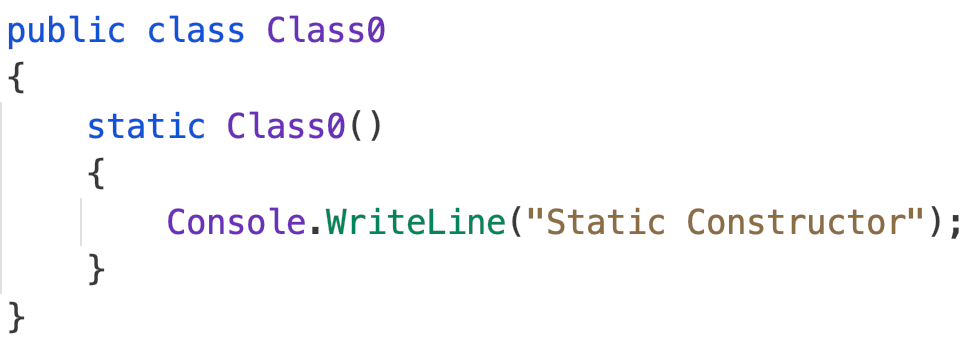
Konstruktor jest to metoda, wywoływana w chwili utworzenia nowej instancji klasy. Konstruktor może być:

* Bezparametryczny (sygnatura konstruktora jest pusta)
* Wieloargumentowy (sygnatura konstruktora składa się   
  z przynajmniej jednego argumentu formalnego)
* Statyczny (Wywoływany w momencie pierwszej interakcji z klasą, na przykład utworzenie nowej instancji klasy, bądź odczytanie wartości statycznego atrybutu klasy)

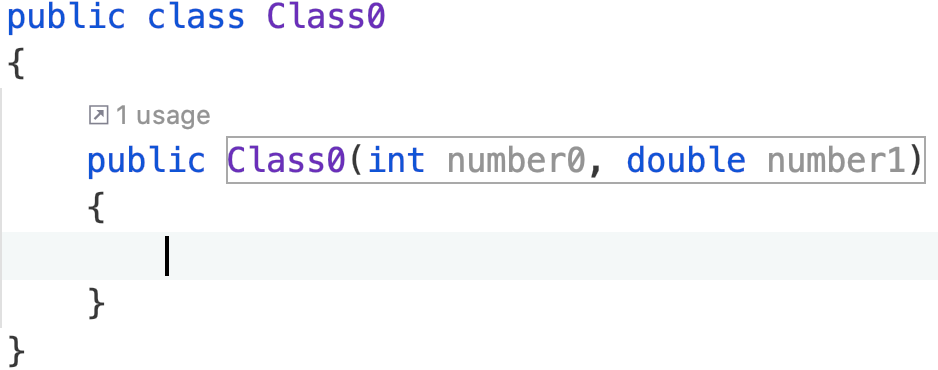
## Deklaracja Konstruktora:



*RYSUNEK 16 DEKLARACJA Konstruktora Bezparametrycznego*



*RYSUNEK 18 DEKLARACJA Konstruktora statycznego*



*RYSUNEK 17 DEKLARACJA Konstruktora wieloargumentowego*

Blok konstruktora może posiadać instrukcje, które mają być wykonane w trakcie obsługi kodu konstruktora, jak na przykład: wyświetlenie określonego łańcucha znakowego.