Programowanie obiektowe

www.jpnowicki.com

Programowanie zorientowane obiektowo (*ang. Object Oriented Programming)* powstało w odpowiedzi na problemy pojawiające się w programowaniu strukturalnym. W tym pierwszym skupialiśmy się na funkcjach (funkcje wywołują funkcje, które też wywołują funkcje). Program zakodowany obiektowo jest zorganizowany wokół danych, a nie funkcji. Często OOP określa się jako dane kontrolujące dostęp do kodu.

Podstawowymi pojęciami w programowaniu obiektowym są **klasy** i **obiekty**. Klasa składa się ze zbioru zmiennych oraz zbioru metod. Pełni ona rolę wzorca dla obiektów. Obiekt jest instancją danej klasy – posiada cechy zdefiniowane przez jego klasę. Obiekt może przechowywać dane oraz wykonywać zadania. Zbiór metod, które mogą być wykonane przez dany obiekt określa jego **interfejs**.

Programowanie obiektowe można porównać do zbioru obiektów, które poprzez wysyłanie komunikatów za pomocą metod mówią sobie nawzajem, co robić.

W praktyce wygląda to tak, że najpierw musimy zdefiniować daną klasę. Na jej podstawie możemy tworzyć obiekty czyli konkretne egzemplarze danej klasy.

- Najpierw tworzymy obiekt danej klasy za pomocą instrukcji :

  $zmienna_obiektowa=new nazwa_klasy();

- Aby odwołać się do dowolnej składowej klasy należy użyć instrukcji :

  $nazwa_obiektu → nazwa pola

  lub

  $nazwa_obiektu → nazwa_metody(argumenty_metody);

Trzeba też wyjaśnić co się dzieje podczas tworzenia nowego obiektu lub jego niszczenia. Tutaj przychodzą z pomocą dwie metody, które nazywamy **konstruktorami** i **destruktorami**. Konstruktor jest specjalną metodą wywoływaną automatycznie podczas tworzenia obiektu danej klasy. Natomiast destruktor jest przeciwieństwem konstruktora – wykonuje niezbędne czynności podczas usuwania obiekt.

W języku PHP składowe danej klasy (pola i metody) muszą mieć określony sposób dostępu zdefiniowany za pomocą poniższych modyfikatorów :

- public – dostęp publiczny
- private – dostęp prywatny
- protected- dostęp chroniony

W pierwszym przypadku dostęp do danej składowej jest nieograniczony. W drugim przypadku dostęp do składowej jest ograniczony tylko do klasy, w której składowa została zdefiniowana. W ostatnim przypadku składowa jest dostępna z poziomu danej klasy oraz poziomu klas, które po niej dziedziczą.

Zwykle by móc używać składowych zdefiniowanych w danej klasy, musimy najpierw utworzyć obiekt dane klasy, a następnie go z tą klasą powiązać.

Istnieje jednak szybszy sposób dostępu do metody danej klasy. Służy do tego słowo kluczowe **static**. W tym wypadku deklaracja będzie miała poniższą postać :

- specyfikator_dostępu static $nazwa_pola

  lub
- specyfikator_dostępu static function nazwa_metody(argumenty)

Powyższa deklaracja umożliwia dostęp do składowej statycznej za pomocą instrukcji :

- nazwa_klasy :: nazwa_pola

  lub
- nazwa_klasy :: nazwa_metody(argumenty)

Ostatnią cechą programowania obiektowego, które chcę przedstawić jest **dziedziczene**. Jeżeli klasa A odziedziczy cechy klasy B to mówimy, że klasa A jest klasą nadrzędną (*ang. Superclass)*, natomiast klasa B jest klasą podrzędną (*ang. Subclass)*. Klasa B jest wyspecjalizowaną wersją klasy A, ponieważ dziedziczy ona wszystkie pola i metody zdefiniowane w klasie nadrzędnej.