|  |  |  |  |
| --- | --- | --- | --- |
| Sprawozdanie nr  **2** | Temat  **Charakterystyka programowania obiektowego** | | |
| I IS(s)  **L4** | Imiona i nazwiska  **Krzysztof Radoń** | | |
| Data wykonania  2016-05-25 | Data oddania  2016-06-24 | Data poprawy | Ocena |

*Wykład nr 2: Charakterystyka programowania obiekt.*

1. ***Definicja programowanie obiektowego***

***Programowanie obiektowe*** – [paradygmat programowania](https://pl.wikipedia.org/wiki/Paradygmat_programowania), w którym programy definiuje się za pomocą [obiektów](https://pl.wikipedia.org/wiki/Obiekt_%28programowanie_obiektowe%29) – elementów łączących *stan* (czyli [dane](https://pl.wikipedia.org/wiki/Dane), nazywane najczęściej [polami](https://pl.wikipedia.org/wiki/Pole_%28informatyka%29)) i *zachowanie* (czyli procedury, tu: [metody](https://pl.wikipedia.org/wiki/Metoda_%28programowanie_obiektowe%29)). Obiektowy program komputerowy wyrażony jest jako zbiór takich obiektów, komunikujących się pomiędzy sobą w celu wykonywania zadań.

Podejście to różni się od tradycyjnego [programowania proceduralnego](https://pl.wikipedia.org/wiki/Programowanie_proceduralne), gdzie dane i [procedury](https://pl.wikipedia.org/wiki/Podprogram) nie są ze sobą bezpośrednio związane. Programowanie obiektowe ma ułatwić pisanie, konserwację i wielokrotne użycie programów lub ich fragmentów.

Największym atutem programowania, projektowania oraz analizy obiektowej jest zgodność takiego podejścia z rzeczywistością – mózg ludzki jest w naturalny sposób najlepiej przystosowany do takiego podejścia przy przetwarzaniu informacji.

1. ***Podstawowe założenia paradygmatu obiektowego***

Istnieje pewna różnica zdań co do tego, jakie cechy [języków programowania](https://pl.wikipedia.org/wiki/J%C4%99zyk_programowania) czynią je obiektowymi. Powszechnie uważa się, że najważniejsze są następujące cechy:

[***Abstrakcja***](https://pl.wikipedia.org/wiki/Abstrakcja_%28programowanie%29)

Każdy obiekt w systemie służy jako model abstrakcyjnego „wykonawcy”, który może wykonywać pracę, opisywać i zmieniać swój stan oraz komunikować się z innymi obiektami w systemie bez ujawniania, w jaki sposób zaimplementowano dane cechy. Procesy, funkcje lub metody mogą być również abstrahowane, a kiedy tak się dzieje, konieczne są rozmaite techniki rozszerzania abstrakcji.

[***Hermetyzacja***](https://pl.wikipedia.org/wiki/Hermetyzacja_%28informatyka%29)

Czyli ukrywanie [implementacji](https://pl.wikipedia.org/wiki/Implementacja_%28informatyka%29), 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](https://pl.wikipedia.org/wiki/Interfejs_%28programowanie_obiektowe%29), który określa dopuszczalne metody współpracy. Pewne języki osłabiają to założenie, dopuszczając pewien poziom bezpośredniego (kontrolowanego) dostępu do „wnętrzności” obiektu. Ograniczają w ten sposób poziom abstrakcji. Przykładowo w niektórych [kompilatorach](https://pl.wikipedia.org/wiki/Kompilator) języka [C++](https://pl.wikipedia.org/wiki/C%2B%2B) istnieje możliwość tymczasowego wyłączenia mechanizmu enkapsulacji; otwiera to dostęp do wszystkich pól i metod prywatnych, ułatwiając programistom pracę nad pośrednimi etapami tworzenia kodu i [znajdowaniem błędów](https://pl.wikipedia.org/wiki/Debugowanie).

[***Polimorfizm***](https://pl.wikipedia.org/wiki/Polimorfizm_%28informatyka%29)

[Referencje](https://pl.wikipedia.org/wiki/Referencja_%28informatyka%29) i kolekcje obiektów mogą dotyczyć obiektów różnego [typu](https://pl.wikipedia.org/wiki/Typ_danych), a wywołanie metody dla referencji spowoduje zachowanie odpowiednie dla pełnego typu obiektu wywoływanego. Jeśli dzieje się to w czasie działania programu, to nazywa się to *późnym wiązaniem* lub *wiązaniem dynamicznym*. Niektóre języki udostępniają bardziej statyczne (w trakcie kompilacji) rozwiązania polimorfizmu – na przykład [szablony](https://pl.wikipedia.org/wiki/Szablon_%28programowanie%29) i [przeciążanie operatorów](https://pl.wikipedia.org/wiki/Przeci%C4%85%C5%BCanie_operator%C3%B3w) w [C++](https://pl.wikipedia.org/wiki/C%2B%2B).

[***Dziedziczenie***](https://pl.wikipedia.org/wiki/Dziedziczenie_%28programowanie%29)

Porządkuje i wspomaga polimorfizm i enkapsulację dzięki umożliwieniu definiowania i tworzenia specjalizowanych obiektów na podstawie bardziej ogólnych. Dla obiektów specjalizowanych nie trzeba redefiniować całej funkcjonalności, lecz tylko tę, której nie ma obiekt ogólniejszy. W typowym przypadku powstają grupy obiektów zwane *klasami*, oraz grupy klas zwane *drzewami*. Odzwierciedlają one wspólne cechy obiektów.

## *Podział*

Można wyróżnić dwa zasadnicze podtypy programowania obiektowego:

### *Programowanie oparte na klasach*

Definiowane są klasy, czyli typy zmiennych, a następnie tworzone są obiekty, czyli zmienne (w uproszczeniu) tych typów.

### *Programowanie oparte na prototypach*

W tym podejściu nie istnieje pojęcie klasy. Nowe obiekty tworzy się w oparciu o istniejący już obiekt – prototyp, po którym dziedziczone są pola i metody i można go rozszerzać o nowe. Spotykany raczej w [językach interpretowanych](https://pl.wikipedia.org/wiki/J%C4%99zyk_interpretowany) np. [JavaScript](https://pl.wikipedia.org/wiki/JavaScript).