PRZYPOMNIENIE: KLASY, OBIEKTY I KONSTRUKTORY

Co to jest klasa?

Klasa to szablon do tworzenia obiektów.

Zawiera:

- pola - opisują cechy obiektu (dane),
- metody - opisują zachowanie obiektu,
- konstruktory - służą do inicjalizacji obiektu.

Co to jest obiekt?

Obiekt to instancja klasy, czyli rzeczywisty element utworzony w pamięci komputera na podstawie klasy.

Konstruktory

Konstruktor to specjalna metoda, która jest wywoływana automatycznie w momencie tworzenia obiektu.

Cechy konstruktora:

- ma taką samą nazwę jak klasa,
- nie posiada typu zwracanego (nawet `void`),
- może przyjmować parametry,
- w jednej klasie może występować wiele konstruktorów (przeciążanie).

Domyślny konstruktor

Jeżeli w klasie nie zdefiniujemy żadnego konstruktora, Java automatycznie tworzy konstruktor domyślny:

```
public Samochod() {}
```

Działa tylko wtedy, gdy nie napiszesz własnego.

Konstruktor

- Konstruktor to specjalna metoda, która ustawia początkowe wartości pól.

- Można tworzyć wiele konstruktorów z przeciążaniem, np. z różnymi parametrami.

Słowo kluczowe this

`this` oznacza bieżący obiekt, na którym aktualnie pracuje program.

Stosujemy je najczęściej, gdy:

- nazwa pola klasy i nazwa parametru konstruktora są takie same,
- chcemy jednoznacznie odwołać się do pola obiektu.

```
class Samochod {
    String marka;

    Samochod(String marka) {
        this.marka = marka;
    }
}
```

POLIMORFIZM

Co to jest polimorfizm?

Polimorfizm to mechanizm, który pozwala traktować obiekty różnych klas w taki sam sposób, o ile mają wspólną klasę bazową.

W praktyce oznacza to:

Jeden typ → wiele zachowań.

O co w tym chodzi

Wyobraź sobie, że:

- masz różne obiekty,
- każdy z nich potrafi wykonać tę samą operację,
- ale każdy robi to na swój sposób.

Program:

- nie musi wiedzieć, z jakim dokładnie obiektem pracuje,
- wystarczy, że zna wspólny typ i dostępne metody.

Jak to działa w Javie

W Javie:

- zmienna może mieć typ klasy bazowej,
- ale przechowywać obiekt klasy pochodnej.

Podczas wywołania metody:

- Java sprawdza rzeczywisty typ obiektu,
- a nie typ zmiennej,
- i wybiera odpowiednią wersję metody.

Dlaczego polimorfizm jest potrzebny?

Polimorfizm:

- upraszcza kod,
- zwiększa elastyczność programu,
- ułatwia rozbudowę aplikacji w przyszłości.

Dzięki niemu:

- nie trzeba sprawdzać typu obiektu (`if`, `instanceof`),
- nie trzeba pisać wielu warunków,
- łatwiej dodawać nowe klasy bez zmiany istniejącego kodu.

Co daje polimorfizm programiście?

- mniej powtarzającego się kodu,
- lepszą czytelność programu,
- łatwiejsze utrzymanie i rozwój projektu,
- zgodność z zasadami programowania obiektowego.