Programowanie 1

Sławomir Pluciński (splucinski@pjwstk.edu.pl)

Programowanie zorientowane obiektowo

Strategie wytwarzania oprogramowania:

imperatywne -> funkcyjne -> obiektowe

Zalety programowania zorientowanego obiektowo:

- odwzorowanie świata rzeczywistego
- czytelny kod
- możliwość wielokrotnie użycia raz napisanego kodu
- spójne podejście do zarządzania kodem
- stabilność i możliwość rozwoju
- pogrupowanie kodu

Wady programowania zorientowanego obiektowo:

- utrudnione testowanie i szukanie błędów
- mniejsza elastyczność
- nie odpowiednie dla małych projektów
- nie ma zastosowania dla wszystkich typów aplikacji

Podstawowe pojęcia

Klasa – częściowa lub całkowita definicja obiektów. Definicja obejmuje dopuszczalny stan obiektu oraz jego zachowanie. Obiekt, który został utworzony na podstawie klasy nazywa się jego instancją. Klasa będzie typem danych w naszych programach.

Obiekt – konkretny i zdefiniowana dający się wyodrębnić element rzeczywistości reprezentujący klasy. Każdy obiekt będzie miał trzy cechy:

- tożsamość – cecha umożliwiające jego identyfikację
- stan – aktualny stan danych
- zachowanie – zestaw dostępnych metod do manipulowania stanem

Paradygmaty programowania obiektowego

paradygmat – to zbiór pojęć i teorii tworzących podstawy danej nauki.

paradygmat programowania – wzorzec programowania komputerów przedkładany w danym okresie rozwoju informatyki ponad inne lub ceniony w pewnych okolicznościach lub zastosowaniach. Paradygmat programowania definiuje sposób patrzenia programisty na przepływ sterowania i wykonywanie programu komputerowego.

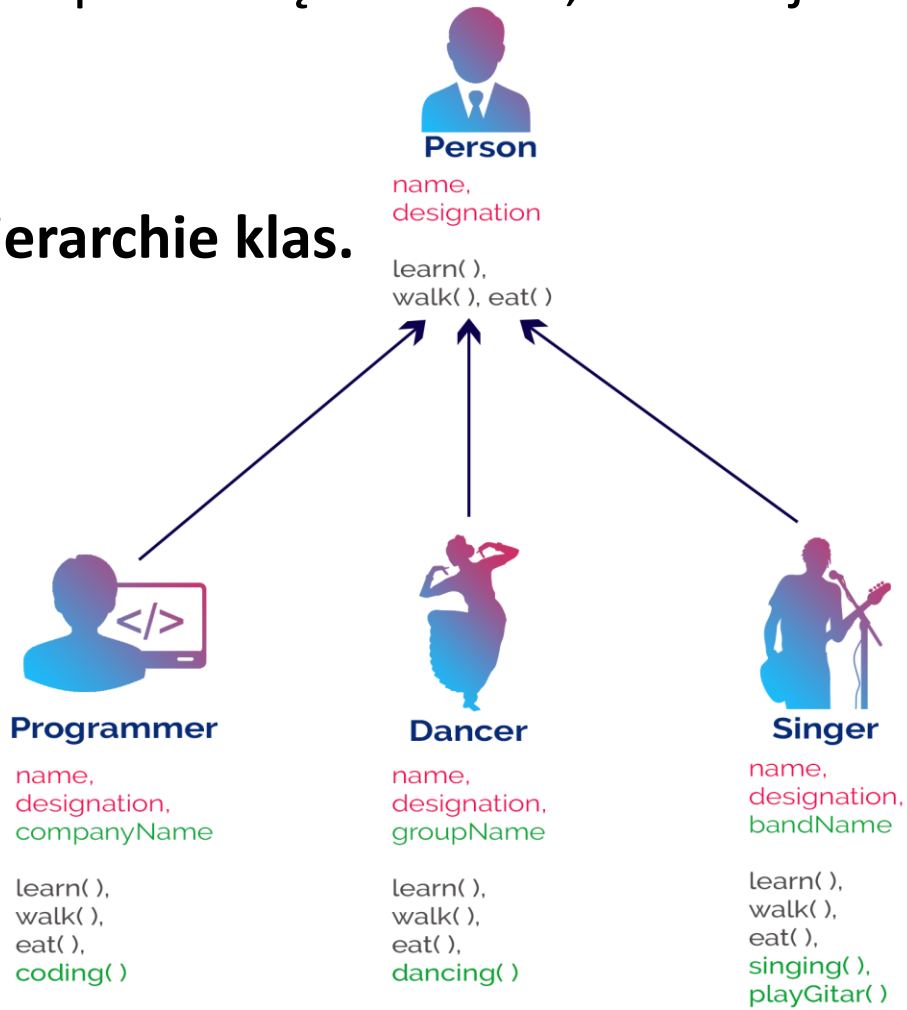
paradygmat programowanie obiektowego – paradygmat programowania, w którym programy definiuje się za pomocą obiektów – elementów łączących stan (czyli dane, nazywane najczęściej atrybutami) i zachowanie (czyli procedury, tu: metody). Obiektowy program komputerowy wyrażony jest jako zbiór takich obiektów, komunikujących się pomiędzy sobą w celu wykonywania zadań.

Dziedziczenie

definicja: mechanizm współdzielenia funkcjonalności między klasami. Klasa może dziedziczyć po innej klasie, co oznacza, że oprócz swoich własnych atrybutów oraz zachowań, uzyskuje także te pochodzące z klasy, z której dziedziczy.

Klasa dziedzicząca jest nazywana klasą pochodną lub potomną zaś klasa, z której następuje dziedziczenie — klasą bazową.

Klasy bazowe i klasy pochodne będą budować tzw. **hierarchie klas.**



Rodzaje dziedziczenia

- Pojedyncze – klasa pochodna może dziedziczyć po maksymalnie jednej klasie bazowej.

- Wielokrotne – klasa pochodna może dziedziczyć po dowolnej ilości klasach bazowych.

Problemy tego podejścia:

- niejednoznaczność semantyczna (problem diamentu)
- problem z łańcuchowym wywołaniem konstruktorów
- trudności implementacyjne

Abstrakcja

definicja: przedstawienie opisywanego świata w postaci uogólnionego, uproszczonego modelu zawierającego kluczowe atrybuty i metody grupy obiektów.

Do realizacji paradygmatu abstrakcji wykorzystywane są klasy abstrakcyjne lub interfejsy.

Interfejs jest specyfikacją metody abstrakcyjnej, nie posiada implementacji a jedynie definicję. Implementację musi zostać dostarczona w klasie implementującej dany interfejs.

Klasy abstrakcyjne

Wiele języków programowania pozwala na wskazanie, że dana klasa jest klasą abstrakcyjną co oznacza, że nie dostarczona jest implementacja danych metod w klasie, a jedynie ich definicja. Nie ma możliwości utworzenia obiektu takiej klasy, możemy po nic jedynie dziedziczyć. W C++ klasą abstrakcyjną nazywamy taką klasę, którą posiada co najmniej jedną metodę czysto wirtualną.

Klasy abstrakcyjne wykorzystywane są do definicji „szkieletu” dla innych klas dziedziczących po takiej klasie.

Hermetyzacja

definicja: polega na ukrywaniu pewnych danych składowych lub metod obiektów danej klasy tak, aby były one dostępne tylko metodom wewnętrznym danej klasy lub funkcjom zaprzyjaźnionym. Gdy dostęp do wszystkich pól danej klasy jest możliwy wyłącznie poprzez metody, lub inaczej mówiąc: gdy wszystkie pola w klasie znajdują się w sekcji prywatnej lub chronionej, to taką hermetyzację nazywa się hermetyzacją pełną.

Dostęp do metod czy atrybutów jest realizowany przez modyfikatory dostępu:

- public – dostęp nie jest ograniczony
- protected – dostęp jedynie wewnątrz klasy i klas podrzędnych (pochodnych)
- private – dostęp jedynie wewnątrz klasy

```java
public class Book {
    private Long id;
    private String title;
    private String publisher;

    public Book() {
    }

    public Book(String title, String publisher) {
        this.title = title;
        this.publisher = publisher
    }

    public Long getId() {
        return id;
    }

    public String getPublisher() {
        return publisher;
    }

    public void setPublisher(String publisher) {
        this.publisher = publisher;
    }
}
```

Polimorfizm

definicja: mechanizm pozwalający przedstawić implementację tej samej zmiennej czy funkcji na kilka różnych sposobów. W zależności od klasy w ramach których jest wywoływana dana metoda, może przyjmować inną postać.

Wyróżniamy dwa typy polimorfizmu:

- polimorfizm statyczny
  - przeciążanie funkcji
  - przeciążanie operatorów

- polimorfizm dynamiczny
  - funkcje wirtualne
  - funkcje abstrakcyjne

```
class Zwierz {
  String name = "bez imienia";
  Zwierz() { }
  Zwierz(String s) { name = s; }
  String getTyp()      { return "Jakis zwierz"; }
  String getName()     { return  name;   }
  String getVoice()    { return "?"; }

  // Metoda speak symuluje wydanie głosu poprzez wypisanie odpowiedniego komunikatu
  void speak()  {
    System.out.println(getTyp()+" "+getName()+" mówi "+getVoice());
  }
}

class Pies extends Zwierz {
  Pies()         { }
  Pies(String s) { super(s); }
  String getTyp()      { return "Pies";   }
  String getVoice()    { return "HAU, HAU!"; }
}

class Kot extends Zwierz {
  Kot()         { }
  Kot(String s) { super(s); }
  String getTyp()      { return "Kot";        }
  String getVoice()    { return "Miauuuu..."; }
}
```