# Technologie obiektowe

Wstęp do programowania zorientowanego obiektowo

# Zakres przedmiotu

Podstawowe pojęcia obiektowości

Diagramy UML

SOLID

Wzorce projektowe

Testy

Refactoring

# Struktura przedmiotu

Wykład: 30h

Laboratoria: 30h

Konsultacje: 20h

Nakład pracy studenta (bez obecności nauczyciela): 70h (4.5-5h/tydzień)

ECTS: 6.0

# Warunki zaliczenia przedmiotu

Uzyskanie oceny pozytywnej z laboratorium

Warunki zaliczenia laboratoriów określane suwerennie przez osoby prowadzące poszczególne grupy

# Literatura

Gamma E. et al — Design Patterns: Elements of Reusable Object-Oriented Software, , 1994, Addison-Wesley

# Kontakt z nauczycielem akademickim

Piotr Szuster, mgr inż.

retsuz@gmail.com | pszuster@pk.edu.pl

www.retsuz.pl

Konsultacje po rezerwacji terminu

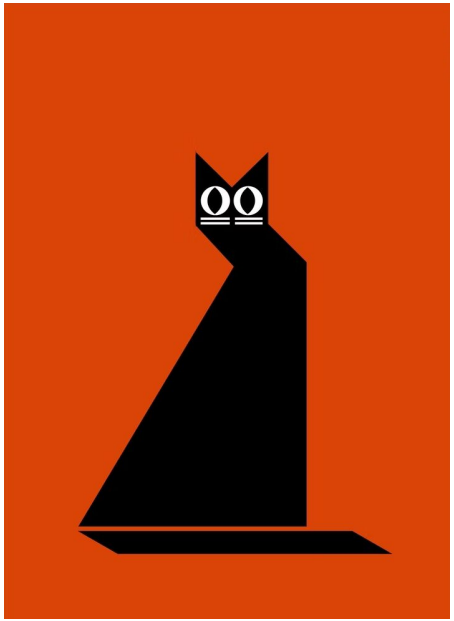
# Wstęp do programowania zorientowanego obiektowo

# Pojęcie abstrakcji

Wolność pojęcia od jego reprezentacji

# Pojęcie abstrakcji

Kot



# Pojęcie abstrakcji

Konstrukcja umożliwiająca ukrycie szczegółów działania zapewniająca jednocześnie korzystanie z funkcjonalności.

# Abstrakcja a programowanie obiektowe

Granica abstrakcji

Struktury abstrakcyjne:

Wektor (matematyka)
Zwierze (zoologia)
Samochód (motoryzacja)

Implementacja:

zmienne
listy
tablice
struktury drzewiaste
zarządzanie pamięcią
operacje arytmetyczne...

PROGRAMOWANIE OBIEKTOWE

# Pojęcie klasy

Definicja sposobu reprezentacji bytu

# Pojęcie klasy

Definicja nowej ‘struktury danych’

Interfejs: zbiór możliwych do wykonania operacji

Implementacja: określenie sposobu wykonania tych operacji

Klasy umożliwiają określone współdzielenie danych dzięki enkapsulacji

# Enkapsulacja

Proces grupowania powiązanych z sobą informacji i stosownych funkcji oraz określania sposobu dostępu do nich

# Pojęcie obiektu

Obiekt - instancja (egzemplarz) klasy

Tożsamość - identyfikacja obiektu w sposób jednoznaczny

Stan - wartość atrybutów

Zachowanie - działanie metod

# Budowa klasy

# Części składowe klasy

Atrybuty (zmienne, pola)

Metody

Konstruktor

# Atrybuty

Zmienne przechowywane wewnątrz klasy/obiektu

Domyślnie dostęp do nich jest ograniczony

Określają stan obiektu w danym momencie

Zmienne statyczne odnoszą się do klas, a nie obiektów.

# Metody

Funkcje wywoływane na obiekcie/klasie

Stanowią interfejs

Działanie może zależeć od stanu

Metody statyczne odnoszą się do klas, a nie obiektów

# Konstruktor

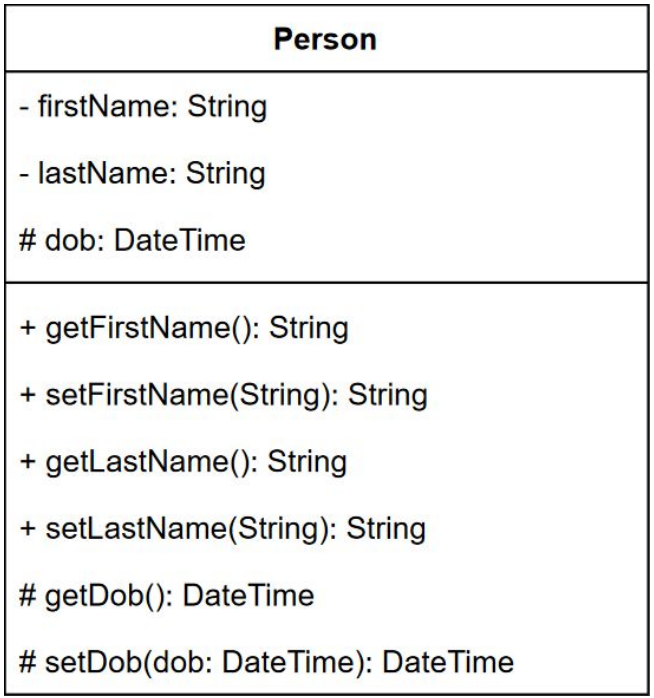
Specjalna metoda tworząca obiekt

# Klasa: przykład

Nazwa klasy
Atrybuty:
<modyfikator dostępu> nazwa: Typ

Metody:

<modyfikator dostępu> nazwa(parametr: Typ): Typ_zwracany



# Obiekty

# Tożsamość

Jednoznaczna identyfikacja obiektu

Operator porównania (porównuje tożsamość)

Metoda equals() (porównuje stan) (zwrotna, symetryczna, przechodnia)

hashCode() (identyfikator stanu obiektu)

# Stan

Wartość zmiennych w danym momencie

Wiele obiektów może znajdować się w tym samym stanie

Zmiany stanu zazwyczaj przeprowadzane przez metody

Hermetyzacja stanu (ograniczenie dostępu)

# Zachowanie

Realizowane przez metody

Metody mogą podlegać hermetyzacji

# Cechy obiektu

Tożsamość

Stan

Zachowanie

# Pozostałe podstawowe pojęcia programowania obiektowego

# Dziedziczenie

Relacja przekazywania cech między klasami.

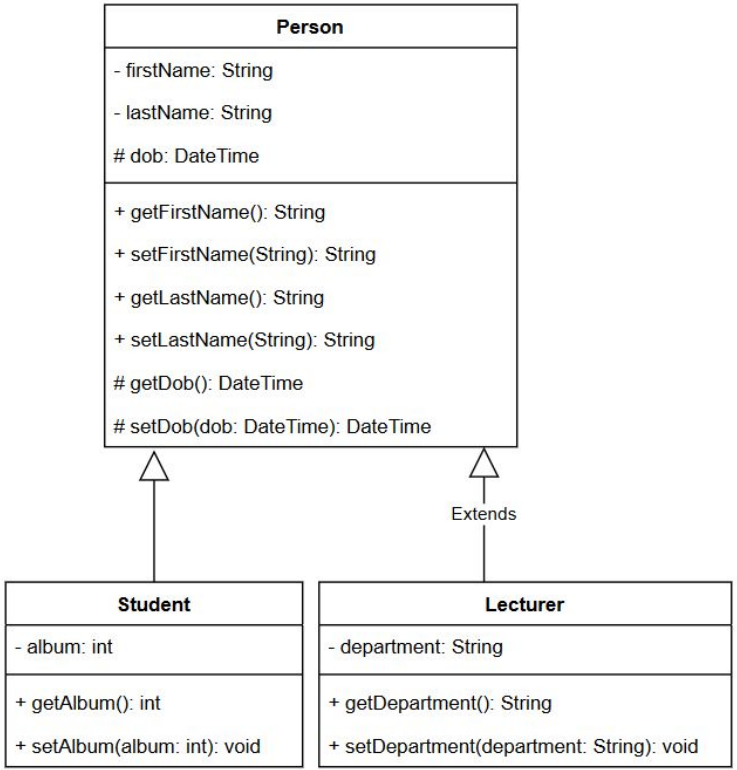
# Dziedziczenie

Wszystkie cechy podlegają przekazaniu

Klasa przekazująca cechy: klasa bazowa

Klasa przejmująca cechy: klasa dziedzicząca

Wykonywane na etapie kompilacji



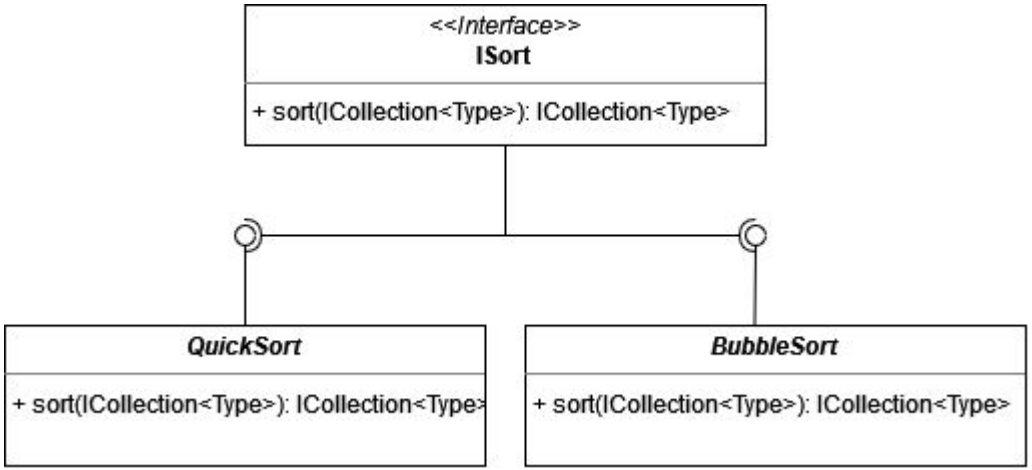
# Polimorfizm

Współdzielenie jednakowego interfejsu przez różne implementacje.

# Polimorfizm

Implementacja interfejsu

Przykład: algorytmy sortujące



# Klasa abstrakcyjna

Klasa nieposiadająca instancji.

# Metoda wirtualna

Metoda, umożliwiająca przeciążenie (zmianę ciała w klasie dziedziczącej).

# Metoda statyczna

Metoda odnosząca się do klasy.

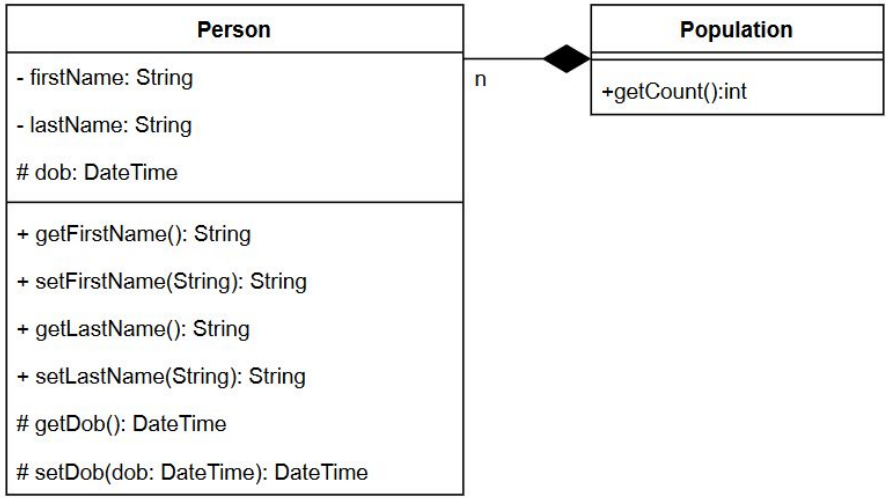
# Metoda abstrakcyjna

Metoda nieposiadająca ciała.

# Kompozycja

Część nie może istnieć bez całości

Zarządzanie częścią odbywa się przez całość

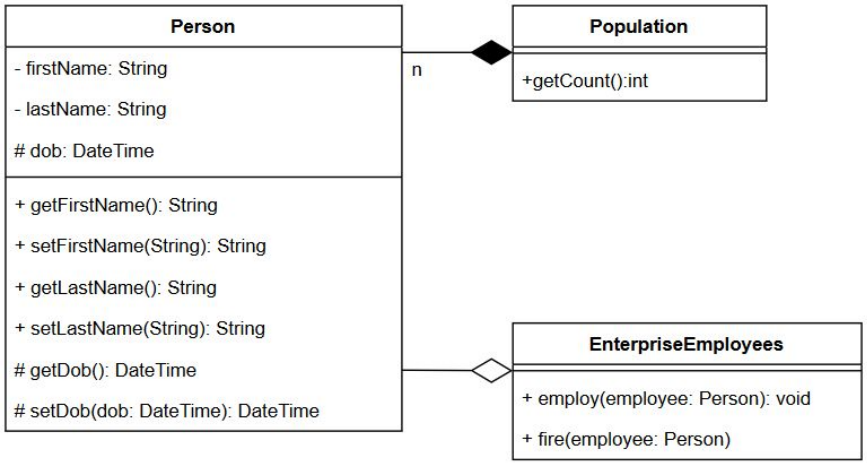


# Agregacja

Części nie są zarządzane przez całość

Przykład: osoba jako częsc populacji może być zatrudniona

Brak pracy nie implikuje utraty życia.

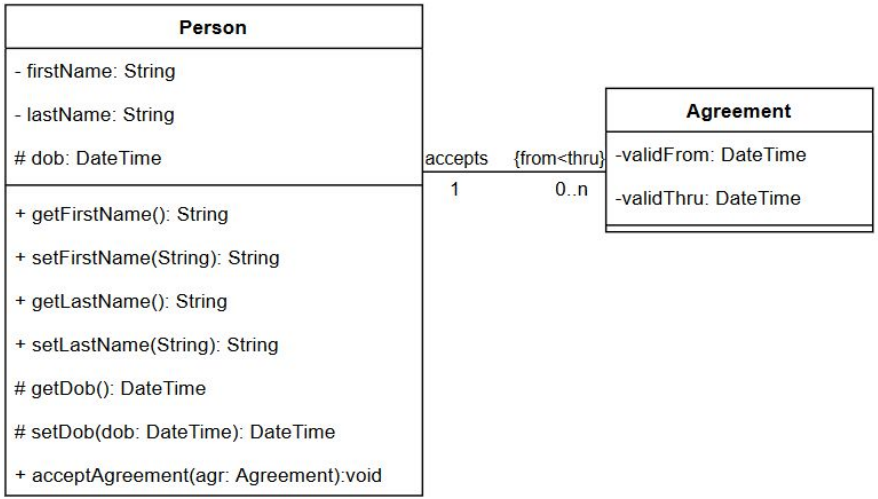


# Asocjacja

Czasowy związek między klasami

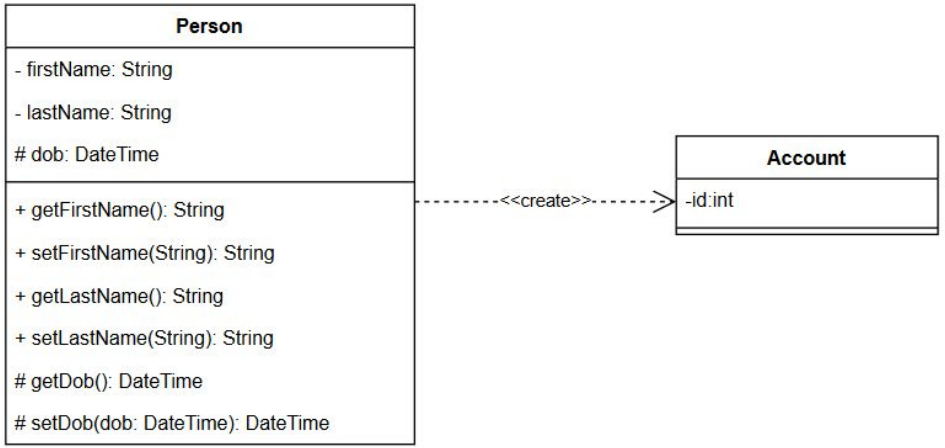
Brak wzajemnego zarządzania

Znaczenie określane przez adnotację



# Zależność

Oddziaływanie między klasami



# Zasady SOLID

# Single responsibility principle

# Open for extension, closed for modification

# Liskov substitution principle

# Interface segregation principle

# Dependency inversion principle