

Object Oriented Programming in Java



Paweł Matyjasik

Senior Developer in Epam Systems, Conference Speaker,
Amateur Music Producer

Paradygmaty programowania

Programowanie proceduralne

Programowanie obiektowe

Programowanie funkcyjne

Pole klasy

Pojedynczy atrybut (cecha) klasy.

Zadanie: utworzenie klasy z atrybutami

- Tworzymy klasę człowiek z publicznymi polami wzrost, waga, imię
- Tworzymy instancję klasy
- Ustawiamy wartości dla kolejnych pól
- Zmieniamy widoczność pól na private

Metoda

Zawiera jedno zachowanie klasy

Zadanie: dodanie metod

- Dodajemy metodę nadającą imię
- Dodajemy metodę przedstawiającą daną osobę

Konstruktor

Specjalna metoda klasy wywoływana podczas tworzenia instancji obiektu

Zadanie: dodanie konstruktora do osoby

- Dodajemy konstruktor bez parametrów
- Dodajemy konstruktor z dwoma parametrami – wzrost i waga
- Dodajemy nowe pola typu final – miejsce urodzenia i pesel
- Utworzenie nowego konstruktora z wymaganymi parametrami

Tworzenie obiektu

```
<JavaType> <variable> = new <JavaObject>();
```

Tworzy nowy obiekt i alokuje potrzebną pamięć.

Przeciążanie

Metoda o tej samej nazwie ale różnych parametrach

Zadanie: przeciążenie metody

- Trzy warianty metody sayThis, przyjmująca daną typu:
 - String – wypisze na konsoli String`a
 - boolean – wypisze na konsoli YES lub NO
 - Integer – wypisze na konsoli kolejne liczby

Dziedziczenie

Tworzenie klasy na podstawie innej klasy nadrzędnej

Zadanie: utworzenie hierarchii dziedziczenia

- Utworzenie nowej klasy – pies
- Dodanie pól wzrost, waga, imię
- Utworzenie klasy rodzica – Ssak

Zadanie: utworzenie hierarchii dziedziczenia

- Dodanie metody służącej do chodzenia w rodzicu
- Nadpisanie metody w klasach dziedziczących

Klasa abstrakcyjna

Klasa dla której nie można utworzyć instancji

Generyczność

Umożliwia operacje na obiektach różnego typu z zachowaniem silnego typowania

Generyczność

```
List<String> names = new ArrayList<String>();
```

```
String firstName = names.get(0);
```

```
List<Person> people = new ArrayList<Person>();
```

```
Person firstPerson = people.get(0);
```

Zadanie: napisanie klasy Optional<T>

- Problem: często występujące NPE na produkcji
- Zaimplementowanie klasy Optional:
 - Optional.empty()
 - Optional.from(T)
 - Optional.isPresent()
 - Optional.get()

Typy, Autoboxing, Unboxing

Typy proste mają swoją reprezentację obiektową

A solid purple vertical bar is positioned on the left side of the slide.

Koncepcje programowania obiektowego

Dziedziczenie

Enkapsulacja

Polimorfizm

Koncepcje programowania obiektowego

Dziedziczenie

Tworzenie klasy na
podstawie innej klasy
nadrzędnej

Enkapsulacja

Polimorfizm

Koncepcje programowania obiektowego

Dziedziczenie

Enkapsulacja

Ograniczanie dostępu do pola/metody poprzez jeden z atrybutów:

- private
- protected
- package private
- public

Polimorfizm

Koncepcje programowania obiektowego

Dziedziczenie

Enkapsulacja

Polimorfizm

Warstwa abstrakcyjna
może być użyta na różne
sposoby

Koncepcje programowania obiektowego

Dziedziczenie

Tworzenie klasy na podstawie innej klasy nadrzędnej

Enkapsulacja

Ograniczanie dostępu do pola/metody poprzez jeden z atrybutów:

- private
- protected
- package private
- public

Polimorfizm

Warstwa abstrakcyjna może być użyta na różne sposoby

Interfejs

Definiuje kontrakt metody, nie definiuje implementacji

Klasa anonimowa

Nie posiadająca nazwy; występuje tylko jedna instancja

Zadanie: utworzenie klasy anonimowej

- Dla klasy człowiek zdefiniowanie metody zjedz
- Jako argument przyjmuje interfejs posiłku
- W metodzie logujemy dane z przekazanej instancji
- Wywołujemy metodę przy użyciu klasy anonimowej

SOLID

Dobre praktyki programowania obiektowego

SOLID

Single responsibility principle

SOLID

Open/Closed Principle

SOLID

Liskov Substitution Principle

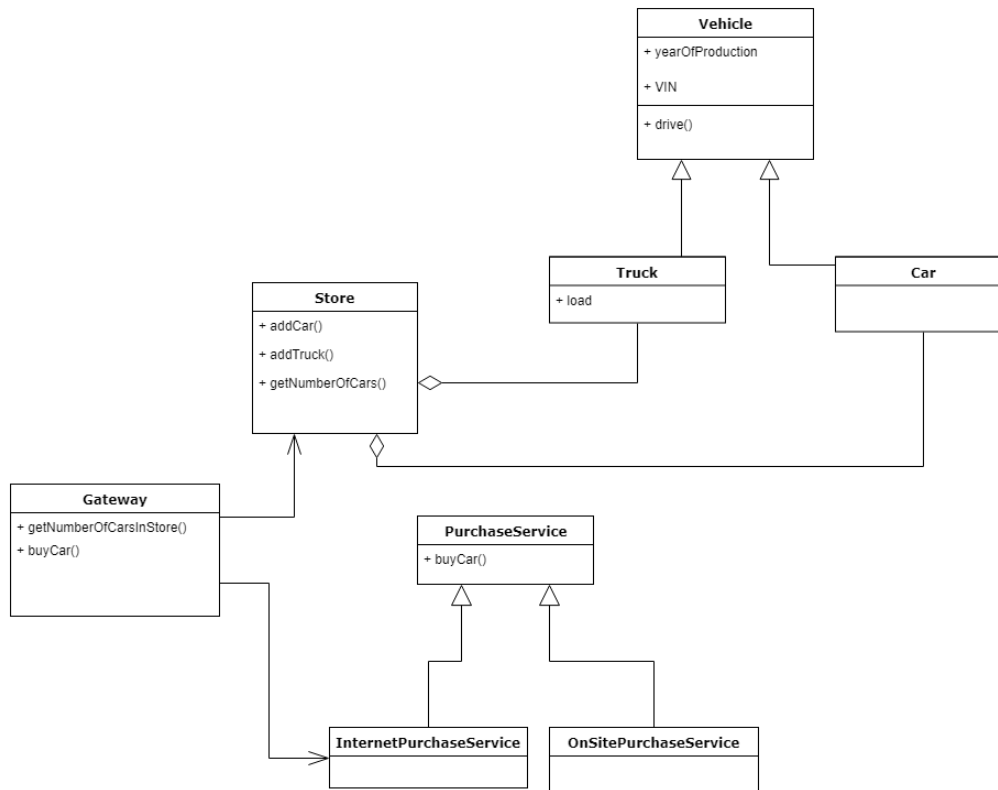
SOLID

Interface Segregation Principle

SOLID

Dependency Inversion Principle

Zadanie: system zarządzania pojazdami



Zadanie: system zarządzania pojazdami

- Utworzyć klasy reprezentujące pojazdy: samochód osobowy, ciężarówka, skuter
- Zidentyfikować pola typu final i pozostałe pola
- Utworzenie wymaganych konstruktorów
- Przeniesienie wspólnych pól do klasy rodzica

Dzięki

Paweł Matyjasik