POLITECHNIKA RZESZOWSKA
im. IGNACEGO ŁUKASIEWICZA

Katedra
Informatyki i Automatyki
Politechnika Rzeszowska

Inżynieria Oprogramowania

Sprawozdanie nr 5

pt.: „Diagram klas (Class diagram)”

Data wykonania: 09.05.2023

Grupa: L8

Daniel Kleczyński

Spis treści

1. Cel pracy

Celem niniejszego ćwiczenia diagram klas na podstawie kodu: Celem tego zadania jest zrozumienie podstawowych elementów diagramu klas oraz umiejętność przekształcania istniejącego kodu na diagram klas. Zadanie to wymaga identyfikacji klas, ich składników, relacji między nimi oraz odpowiedniego użycia notacji. diagram klas w podejściu ORM: Celem tego zadania jest zapoznanie się z mapowaniem obiektowo-relacyjnym (ORM) oraz umiejętność tworzenia diagramów klas na podstawie schematu ERD. Zadanie to wymaga zrozumienia relacji między tabelami oraz sposobu reprezentacji tych relacji w diagramie klas. Diagram klas dla systemu obsługującego wypożyczalnie samochodów: Celem tego zadania jest praktyczne zastosowanie diagramu klas w kontekście konkretnego systemu. Zadanie to wymaga identyfikacji kluczowych klas oraz ich właściwości, określenia relacji między klasami oraz uwzględnienia funkcjonalności systemu w diagramie. podsumuj te zadania kilkoma zdaniami

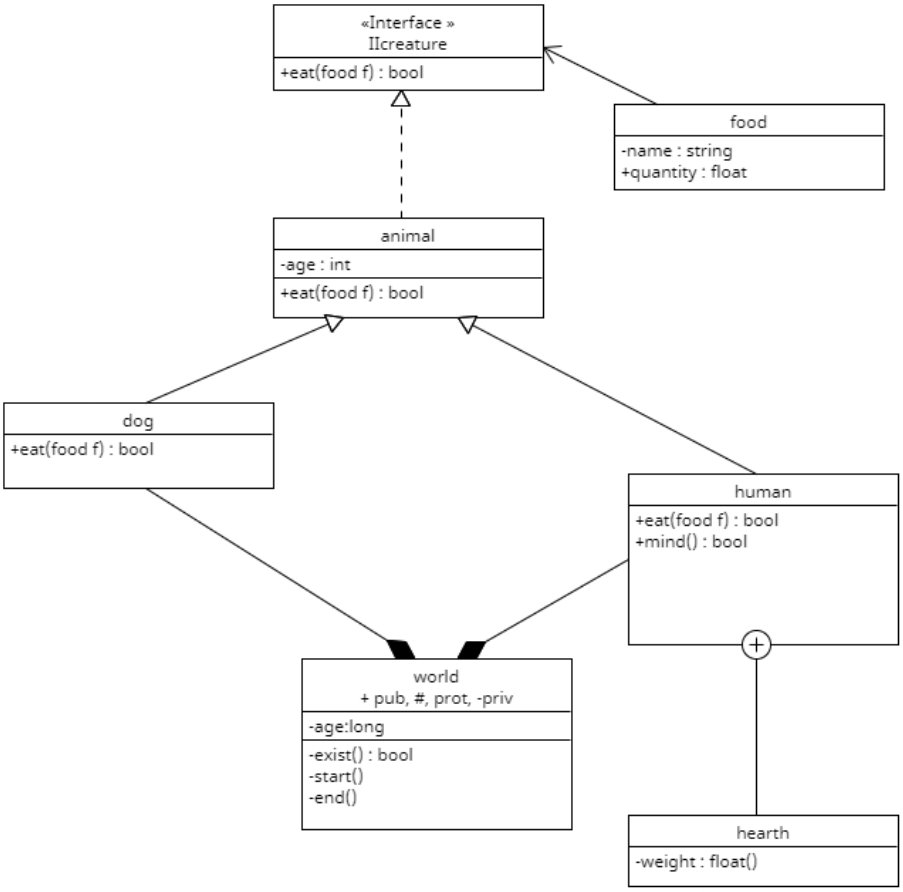
2. Przebieg Pracy

2.1. Stworzenie diagramu klas dla przykładowego kodu w java

KOD:

```
public class world() {
	private long age;
	private bool exist() {...}
	private start() {...}
	private end() {...}
	private List<dog> dogs;
	private List<human> people;
}

public class human extends animal {
	public bool eat(food f) {...};
	public bool mind() {...};
	private class heart {
		private float weight;
	}
}

public class dog extends animal {
	public bool eat(food f) {...}
}

public class animal implements Icreature{
	private int age;
	public bool eat(food f) {...}
}

public interface Icreature  {
	public bool eat(food f);
}

public class food {
	private string name;
	public float quantity;
}
```

Dagram sekwencji:



Ten diagram klas przedstawia hierarchię dziedziczenia i interfejsów pomiędzy różnymi klasami reprezentującymi świat, zwierzęta, człowieka, psy i jedzenie. Ogólnie rzecz biorąc, system opisuje relacje między różnymi obiektami w kontekście świata.

Klasa "World" przechowuje informacje dotyczące wieku świata oraz listy psów i ludzi w tym świecie. Klasa "Human" reprezentuje człowieka, który ma możliwość jedzenia i myślenia. Klasa "Dog" reprezentuje psa, który również może jeść. Klasa "Animal" jest ogólną klasą dla zwierząt i implementuje interfejs "ICreature", który definiuje akcję jedzenia.

Klasa "Food" reprezentuje jedzenie i przechowuje informacje o nazwie i ilości dostępnego jedzenia.

Ogólnie rzecz biorąc, ten system opisuje interakcje między różnymi obiektami, takimi jak ludzie, psy i jedzenie, oraz określa ich zachowanie, takie jak jedzenie, myślenie itp.

2.2. Stworzenie diagramu klas na podstawie diagramu ERD

ERD:

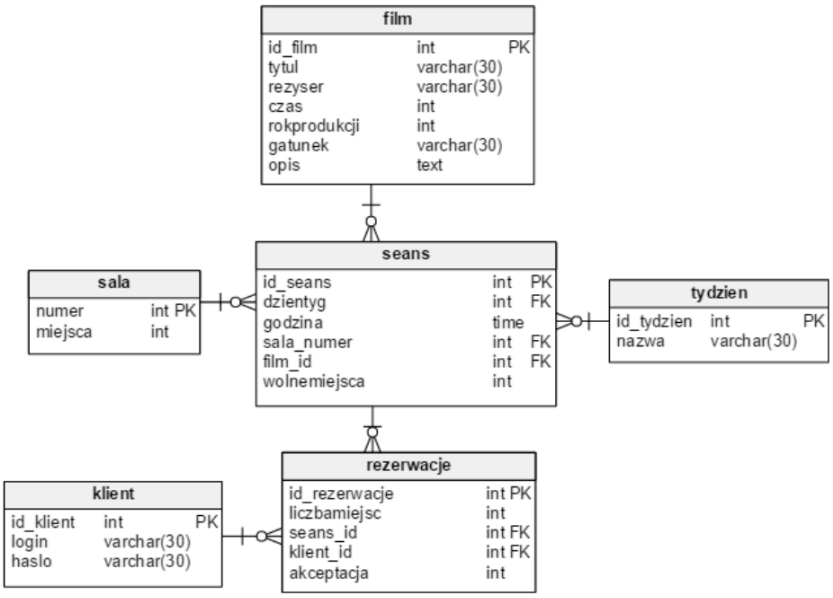
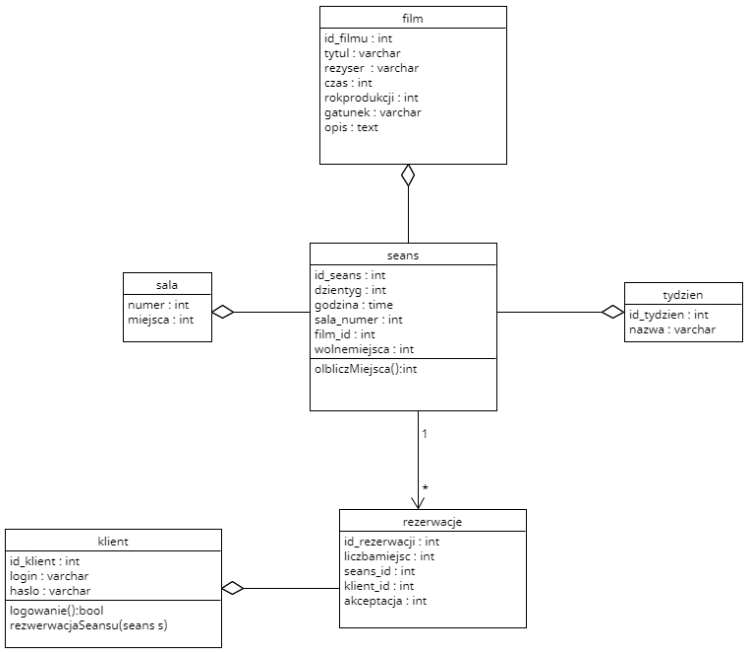


Diagram Klas:



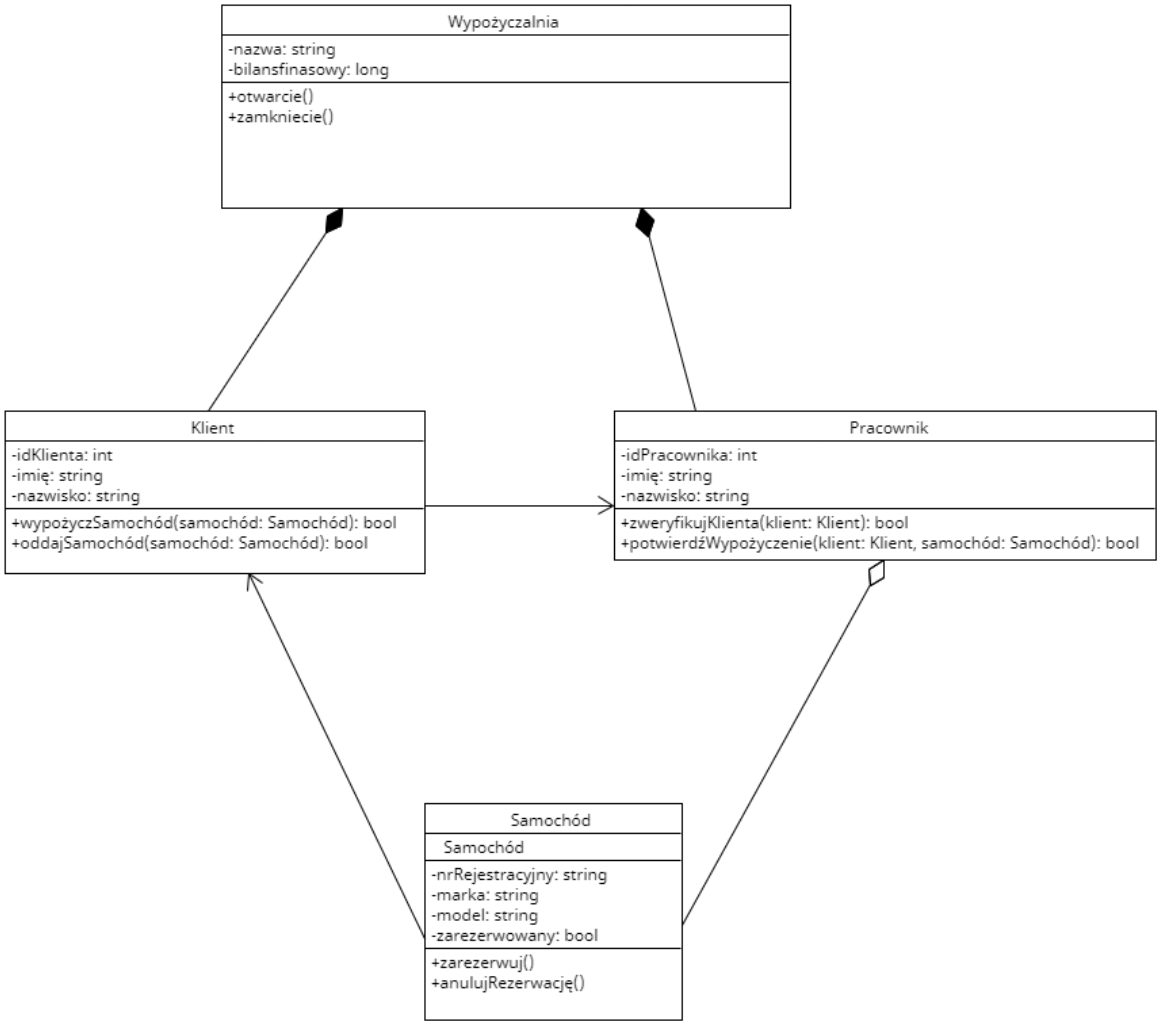
Konwersja diagramu ERD (Entity-Relationship Diagram) na diagram klas jest procesem transformacji modelu danych relacyjnych na model obiektowy. Diagram klas jest często stosowany w programowaniu obiektowym i umożliwia reprezentację struktury klas, relacji między nimi oraz zachowań obiektów.

Korzyści z konwersji ERD do diagramu klas obejmują. Odzwierciedlenie obiektowej natury systemu: Diagram klas dostarcza bardziej intuicyjnego sposobu reprezentacji obiektów, klas, atrybutów i relacji między nimi. Ułatwia to zrozumienie struktury i zachowań systemu, szczególnie dla programistów pracujących w paradygmacie programowania obiektowego.

Łatwiejsze tworzenie i zarządzanie kodem: Diagram klas stanowi podstawę do generowania kodu w wielu językach programowania. Przejście od diagramu klas do kodu jest zazwyczaj prostsze i mniej czasochłonne niż z ERD. Diagram klas dostarcza również wytycznych dotyczących hierarchii dziedziczenia, interfejsów, metod i atrybutów, co ułatwia implementację kodu.

Wsparcie dla dziedziczenia i polimorfizmu: Diagram klas umożliwia reprezentację dziedziczenia między klasami, co jest istotne w programowaniu obiektowym. Możemy zdefiniować hierarchię dziedziczenia, a także wykorzystać polimorfizm do tworzenia bardziej elastycznych i modułowych struktur.

2.3. Stworzenie diagramu klas dla wypożyczalni samochodów



Klasa "Wypożyczalnia" reprezentuje główną funkcjonalność systemu. Posiada metody takie jak "wypożyczSamochód", które umożliwiają wypożyczenie samochodu przez klienta, "weryfikujKlienta", służące do weryfikacji danych klienta, oraz "potwierdźWypożyczenie", potwierdzające wypożyczenie samochodu przez klienta.

Klasa "Klient" reprezentuje klienta wypożyczalni. Przechowuje informacje takie jak identyfikator klienta, imię i nazwisko. Udostępnia metody "wypożyczSamochód" i "oddajSamochód", które pozwalają klientowi wypożyczyć samochód oraz oddać go po zakończeniu wypożyczenia.

Klasa "Pracownik" reprezentuje pracownika wypożyczalni. Przechowuje informacje takie jak identyfikator pracownika, imię i nazwisko. Udostępnia metody "zweryfikujKlienta" do weryfikacji danych klienta oraz "potwierdźWypożyczenie" do potwierdzenia wypożyczenia samochodu przez klienta.

Klasa "Samochód" reprezentuje samochód wypożyczalni. Przechowuje informacje takie jak numer rejestracyjny, marka i model. Udostępnia metody "zarezerwuj" i "anulujRezerwację" umożliwiające rezerwację samochodu oraz anulowanie rezerwacji.

Ten diagram klas obrazuje strukturę systemu obsługującego wypożyczalnię samochodów oraz interakcje między klientami, pracownikami i samochodami w ramach funkcjonalności wypożyczenia, weryfikacji i potwierdzenia wypożyczenia.

3. Wnioski

Diagram klas jest narzędziem o dużym znaczeniu w inżynierii oprogramowania, które pomaga w projektowaniu i modelowaniu struktury systemu. Przedstawia relacje między klasami, ich właściwości oraz metody, co umożliwia lepsze zrozumienie zależności i interakcji między różnymi komponentami systemu.

Wizualizacja struktury: Diagram klas umożliwia wizualizację struktury systemu, co pozwala programistom, projektantom i innym zainteresowanym osobom lepiej zrozumieć architekturę i organizację systemu. Jest to szczególnie ważne w przypadku większych projektów, gdzie istnieje wiele klas i zależności między nimi.

Zrozumienie relacji: Diagram klas pokazuje relacje między klasami, takie jak dziedziczenie, implementacja interfejsów, kompozycja czy agregacja. Pozwala to zrozumieć hierarchię klas, jak również zależności i interakcje między nimi. Dzięki temu programiści mogą lepiej zidentyfikować, jak poszczególne komponenty systemu współpracują ze sobą.

Projektowanie rozszerzalnego systemu: Diagram klas umożliwia projektowanie rozszerzalnych systemów poprzez hierarchię dziedziczenia i abstrakcję. Dzięki dziedziczeniu można tworzyć specjalizowane klasy, które rozszerzają funkcjonalność ogólnej klasy bazowej. Diagram klas pozwala zobaczyć, jak te klasy współpracują i jak można wprowadzać nowe funkcjonalności poprzez dziedziczenie.

Komunikacja między zespołami: Diagram klas jest skutecznym narzędziem komunikacji między członkami zespołu projektowego. Może służyć jako wspólny język, który pozwala zrozumieć strukturę systemu i jego komponenty. Umożliwia także identyfikację potencjalnych problemów projektowych i zaplanowanie dalszych działań.

Podstawa dla implementacji: Diagram klas może stanowić podstawę dla implementacji systemu, ponieważ przedstawia strukturę klas, ich właściwości i metody. Może być używany jako punkt odniesienia dla programistów, którzy tworzą kod źródłowy na podstawie diagramu klas, co przyspiesza proces implementacji i minimalizuje ryzyko błędów.

Podsumowując, diagram klas jest niezwykle użytecznym narzędziem w procesie projektowania oprogramowania, umożliwiającym lepsze zrozumienie struktury systemu, relacji między klasami oraz wsp