****

**Programowanie obiektowe**

*Aplikacja do sprzedaży pojazdów*

Prowadzący: Autor:

mgr inż. Ewa Żesławska Jakub Mosior

w65500

Kierunek: 3 IIZ/2021, grupa GP02

Rzeszów, r.a. 2023/2024

Spis treści

1.**Cele projektu.  
2.Opis techniczny projektu  
3.Harmonogram realizacji projektu  
4. Prezentacja warstwy użytkowej projektu wraz z jej wizualizacją.  
5.Testy jednostkowe  
6. Zbiór informacji, system kontroli wersji.  
7. Podsumowanie.  
8. Literatura.**

1. **Cele projektu**

Celem projektu jest stworzenie programu który będzie symulował serwis do sprzedaży pojazdów.

**Wymagania funkcjonalne**

* Połączenie z bazą danych
* Walidacja danych wejściowych, tak aby uniemożliwić wprowadzenie niepoprawnych danych
* Wyświetlanie odpowiednich komunikatów w przypadku wystąpienia błędów
* Dostęp do konkretnych funkcji przez konkretnych użytkowników

Obraz zawierający tekst, zrzut ekranu, Czcionka

Opis wygenerowany automatycznie  
  
*Rysunek 1. Panele widoku*

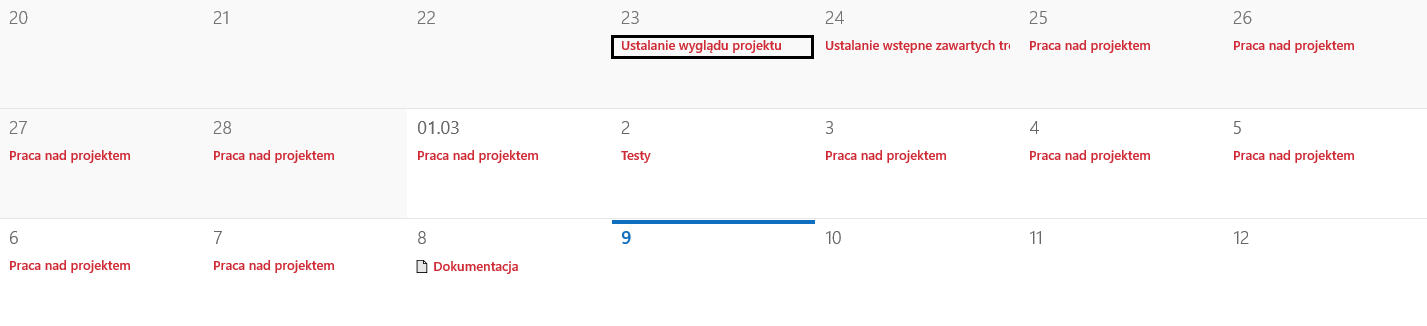
**Wymagania niefunkcjonalne**

* Program stworzony został w języku Java JDK 19.
* Program działa na systemach Windows.
* Program posiada proste i intuicyjne GUI
* Program jest stabilny nie powoduje żadnych problemów, błędów ani nieprzewidzianych anomalii.

1. **Opis techniczny projektu**

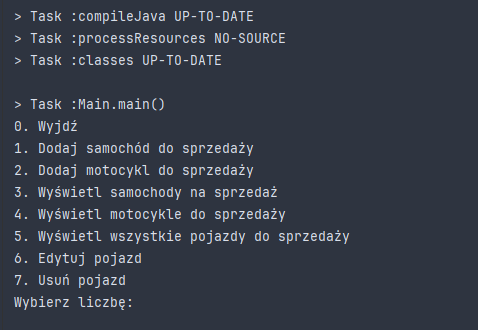
* Środowisko programistyczne Javy: IntelliJ IDEA Ultimate Edition, Java JDK 19
* Komputer z zainstalowanym systemem operacyjnym obsługujący język Java wraz z skonfigurowanym środowiskiem programistycznym.

1. **Harmonogram realizacji projektu**



1. **Prezentacja warstwy użytkowej projektu wraz z jej wizualizacją.**

Użytkownik o uruchomieniu aplikacji wyświetla im się ekran w którym musi wybrać jedną z opcji.

  
  
*Rysunek 2. Początek programu*

Dzięki klasą Menu, oraz Menu Option systemu sprzedaży pojazdów, użytkownik może dodawać, wyświetlać, edytować i usuwać pojazdy z oferty sprzedaży.

Klasa Menu Option znajduje się w pakiecie vehicledealer.menu.

Jest to klasa publiczna, co oznacza, że może być dostępna z innych pakietów. W skrócie, klasa MenuOption jest prostą strukturą danych reprezentującą pojedynczą opcję w menu. Przechowuje opis opcji i akcję do wykonania po jej wybraniu. Metoda execute umożliwia wykonanie związanej z opcją akcji, a getDescription umożliwia uzyskanie opisu opcji.

Natomiast klasa Menu obsługuje interakcję z użytkownikiem, wyświetlając menu i wykonując odpowiednie akcje w zależności od wyboru użytkownika. Są w niej metody takie jak menuOptions, która jest listą opcji menu. W konstruktorze klasy jest dodana domyślna opcja "Wyjdź", która wyświetla komunikat po wybraniu. Konstruktor przyjmuje listę opcji menu i dodaje je do istniejącej listy menuOptions. Metoda show uruchamia interaktywne menu.

W pętli do-while użytkownikowi wyświetlane są opcje menu, a następnie proszony jest o wybór. W przypadku wybrania opcji "0" (wyjście), pętla się kończy.

Metoda displayMenu wyświetla opcje menu wraz z ich numerami dla użytkownika.

Metoda executeOption wykonuje akcję związane z wybraną opcją. Jeżeli numer opcji mieści się w zakresie istniejących opcji, to wywoływana jest metoda execute związanej z nią klasy MenuOption.

W skrócie, klasa Menu obsługuje interakcję z użytkownikiem, wyświetlając menu i wykonując odpowiednie akcje w zależności od wyboru użytkownika.

4.1 VehicleSalesSystem

VehicleSalesSystem stanowi system sprzedaży pojazdów. Klasa zawiera pole db reprezentujące obiekt bazy danych pojazdów (implementujący interfejs VehiclesDatabase), oraz listy carList i motorcycleList przechowujące listy samochodów i motocykli.

Pole idGenerator jest obiektem klasy Random służącym do generowania unikalnych identyfikatorów dla nowych pojazdów.

Konstruktor inicjalizuje listy samochodów i motocykli pobierając dane z bazy danych.

# Metody do dodawania, wyświetlania, edytowania i usuwania pojazdów:

addNewCar() i addNewMotorcycle(): Dodają nowe pojazdy do listy.

displayAllToSale(), displayCarsToSale(), displayMotocyclesToSale(): Wyświetlają listy pojazdów na sprzedaż.

editVehicle(): Pozwala na edycję informacji o wybranym pojeździe.

deleteVehicle(): Usuwa wybrany pojazd z listy.

# Metoda addNewVehicle(VehicleType vehicleType):

Wywołuje addVehicle() z generowanym losowo identyfikatorem.

Metoda addVehicle(VehicleType vehicleType, int id):

Pozwala na dodanie nowego pojazdu do listy, wczytując dane od użytkownika.

Metoda getId():

Pobiera od użytkownika identyfikator pojazdu.

# Obsługa bazy danych:

Dane są wczytywane z bazy danych CSV (CsvVehiclesDatabase) przy inicjalizacji klasy, a także zapisywane po dodaniu, edycji lub usunięciu pojazdu.

Kod ten przedstawia prosty system sprzedaży pojazdów, który interaktywnie obsługuje dodawanie, wyświetlanie, edytowanie i usuwanie pojazdów, korzystając z list oraz bazy danych CSV.

4.2 Interfejs VehiclesDatabase

Interfejs zawiera deklaracje metod służących do obsługi danych związanych z samochodami (Car) i motocyklami (Motorcycle).

Interfejs VehiclesDatabase służy jako abstrakcja warstwy dostępu do danych dla pojazdów w systemie. Klasa implementująca, czyli CsvVehiclesDatabase będzie odpowiedzialna za konkretne operacje związane z pobieraniem i zapisywaniem danych do określonego źródła.

Korzystając z tego interfejsu, system sprzedaży pojazdów (VehicleSalesSystem), może być elastyczny pod względem sposobu przechowywania i pobierania danych o samochodach i motocyklach.

4.3 CsvVehiclesDatabase

CsvVehiclesDatabase, która implementuje interfejs VehiclesDatabase dla przechowywania danych o pojazdach w plikach CSV.

Stałe pola klasy definiują lokalizacje plików CSV, w których przechowywane są dane o samochodach (CARS\_FILE\_LOCATION) i motocyklach (MOTORCYCLES\_FILE\_LOCATION).

Metoda createVehicle

Tworzy obiekt pojazdu (Car lub Motorcycle) na podstawie danych w formie tablicy.

Obraz zawierający tekst, zrzut ekranu, oprogramowanie, Czcionka

Opis wygenerowany automatycznie

Natomiast metoda saveToCsv to metoda zapisująca listę pojazdów do pliku CSV.

4.4 Vehicle

Abstrakcyjna klasa, która reprezentuje ogólny pojazd w systemie dealerów pojazdów.

Klasa Vehicle jest oznaczona jako abstract, co oznacza, że nie może być bezpośrednio instancjonowana. Musi być rozszerzona przez konkretne klasy.

Implementuje interfejs CsvSerializable, co oznacza, że klasy dziedziczące po Vehicle muszą dostarczyć implementację metody toCsv().

Klasa posiada konstruktor, który przyjmuje parametry identyfikatora, marki, modelu i roku produkcji, inicjalizując odpowiednie pola.

Klasa Vehicle jest abstrakcyjna i stanowi bazę dla konkretnych typów pojazdów, takich jak Car i Motorcycle. Dzięki implementacji interfejsu CsvSerializable, każda klasa dziedzicząca po Vehicle musi dostarczyć metodę toCsv(), która przekształca obiekt pojazdu na reprezentację tekstową w formie linii CSV.

Uzywany jest tu również typ wyliczeniowy (enum) VehicleType do reprezentacji kategorii pojazdów w systemie dealerów pojazdów. W kodzie klasy Car i Motorcycle korzystają z tego typu wyliczeniowego, aby określić, czy są samochodami czy motocyklami.

Obraz zawierający tekst, zrzut ekranu, oprogramowanie, Oprogramowanie multimedialne

Opis wygenerowany automatycznie

1. **Zbiór informacji, system kontroli wersji**

Projekt został zrealizowany z wykorzystaniem systemu kontroli wersji Git.

1. **Podsumowanie**

Dokumentacja programu powstała w celu zaprezentowania możliwości programu. Spełnia on swoje założenia i jest prosty w obsłudze. Istnieje możliwość dodania nowych funkcji i takowe oczywiście zostaną dodane, program zostanie poddany masowej aktualizacji funkcji.

1. **Literatura**

Materiały źródłowe – wskazanie literatury i materiałów źródłowych wykorzystanych przy realizacji projektu.

1. <https://chat.openai.com/chat>
2. <https://www.youtube.com>
3. <https://learn.microsoft.com/>
4. <https://stackoverflow.com/>