|  |  |
| --- | --- |
| **Politechnika Świętokrzyska w Kielcach**  **Wydział Elektroniki, Automatyki i Informatyki** | |
| Przedmiot: **Programowanie obiektowe Java - projekt** | |
| Kierunek: **Informatyka  niestacjonarnie** **II rok II semestr** | **Zespół:**  **Klaudia Iwanowicz**  **Szymon Kasprzyk**  **Michał Gruszczyński**  **Oskar Pytlewski**  **Grupa: 11B** |
| Temat:  „Wypożyczalnia samochodów” |
| **SPRAWOZDANIE** |

**Spis treści**

1. WPROWADZENIE
   1. Opis programu ………………………………………………………………………3
   2. Cel programu ………………………………………………………………………..4
   3. Funkcjonalność ………………………………………………………………….…..4
2. KONFIGURACJA I URUCHOMIENIE
   1. Instalacja Javy ……………..……………………………………………………...…5
   2. Pobieranie i konfiguracja projektu …………………………………………………..5
   3. Uruchamianie programu …………………………………………………………….6
3. STRUKTURA DANYCH
   1. Klasa AddCarRating …………………………………………………………..…….9
   2. Klasa AddCars ……………………………………………………………..………13
   3. Klasa CalculateAmountOfRent ……………………………………………………16
   4. Klasa GUI ………………………………………………………………………….20
   5. Klasa MenuGui …………………………………………………………………….22
   6. Klasa MyTextField ……………………………………………………...…………23
   7. Klasa ShowCars ……………………………………………………………………25
   8. Klasa MyLoverPanel ……………………………………………………...……….26
   9. Klasa ShowRatingCars ……………………………………………….……………29

3.10 Klasa AddRentCar ……………………………………………………..…………34  
3.11 Klasa MyFrame …………………………………………………………………...38

1. DODAWANIE SAMOCHODÓW DO BAZY
   1. Marka ………………………………………………………………………………41
   2. Model ………………………………………………………………………………42
   3. Rok produkcji ………………………………………………………………...……43
   4. Cena …………………………………………………………………………..……44
   5. Weryfikacja …………………………………………………………...……………46
2. REZERWACJA DOSTĘPNYCH SAMOCHODÓW
   1. Dodawanie samochodów ……………………………………………………..……48
   2. Okres rezerwacji ………………………………………………………………...…49
   3. Wprowadzanie daty …………………………………………………………..……50
   4. Potwierdzanie wypożyczenia ……………………………....………………………51
3. KOSZTY WYNAJMU
   1. Wybór samochodu ……………………………………………………………...….53
   2. Okres wypożyczenia ……………………………………………………………….54
   3. Obliczanie kosztów wynajmu …………………………………………………...…55
4. SYSTEM OCEN SAMOCHODÓW
   1. Wybór samochodu …………………………………………………………………57
   2. Ocena pojazdu …………………………………………………………………..…58
5. WYŚWIETLANIE OCEN SAMOCHODÓW
   1. Spis wszystkich ocen ………………………………………………………………60
6. PODSUMOWANIE I PERSPEKTYWY ROZWOJU
   1. Podsumowanie funkcjonalności …………………………………………...………61
   2. Propozycje dalszych ulepszeń ………………………………………………..……62
7. **WPROWADZENIE**
   1. Opis programu

Program „Wypożyczalnie samochodów” jest projektem stworzonym w języku Java, który ma na celu zarządzanie systemem rezerwacji samochodów. Program umożliwia zarówno obsługę klienta, jak i zarządzanie dostępną flotą samochodów do wynajęcia.

W ramach tego projektu, stworzyliśmy aplikację okienkową, która zapewni interakcję użytkownika poprzez menu. Program przechowuję informację o samochodach, rezerwacjach oraz ocenach aut.

Program umożliwia klientom wyświetlanie dostępnych samochodów na podstawie podanego ID auta. Użytkownicy będą mogli dokonywać wypożyczenia wybranych samochodów na ustalony przez siebie okres czasu.

Dodatkowo program posiada system obliczania kosztów wynajmu, który przybliży klientom kwotę konkretnego samochodu na wybrany przez siebie okres czasu. Kolejnym systemem jest system oceny samochodu, dzięki czemu będzie można ocenić wybrane przez siebie auto w skali od 1 do 5 (1- najniżej, 5-najwyżej). Po dodanej ocenie można ją sprawdzić w zakładce „wyświetl oceny samochodów”.

Podsumowując, program "Wypożyczalnia samochodów" zapewni wygodne i efektywne zarządzanie wypożyczeniami samochodów, umożliwiając klientom łatwe wyszukiwanie, rezerwowanie, a także ocenianie i obliczanie kosztów wynajmu.

* 1. Cel programu

Celem naszego projektu „Wypożyczalnia samochodów” jest stworzenie wygodnego i kompletnego systemu do zarządzania procesem wypożyczania samochodów. Program ma na celu usprawnienie procesu wypożyczeń, umożliwiając skuteczne zarządzanie samochodami oraz rezerwacjami.

Główne cele projektu obejmują:

1. Ułatwienie wyszukania i zarezerwowania samochodów:

Program umożliwi klientom łatwe i szybkie wyszukanie dostępnego samochodu. Użytkownik będzie mógł rezerwować wybrany samochód online, zwiększając wygodę i dostępność usługi.

1. Łatwy i szybki podgląd ceny:

Program oferuje użytkownikom wcześniejszy podgląd ceny za wypożyczony samochód na określony okres czasu. Umożliwi to lepszy pogląd na ponoszone koszty oraz pomoże w doborze optymalnie dla klienta cenowo samochodu.

1. Efektywne zarządzanie rezerwacjami:

Program zapewnia łatwą obsługę rezerwacji samochodów poprzez dostęp do intuicyjnego interfejsu. Klient będzie mógł sprawdzić dostępność samochodu, wybrać dogodny termin i potwierdzić rezerwację.

1. Szybki widok oceny samochodu po wypożyczeniu:

Po zakończeniu wypożyczenia klient będzie miał możliwość oceny samochodu, aby podzielić się swoimi opiniami i doświadczeniem z innymi użytkownikami. Program umożliwi dodawanie i wyświetlanie ocen samochodów, co pozwoli innym klientom na podjęcie decyzji przy wyborze samochodu do wypożyczenia.

Celem naszego projektu jest dostarczenie funkcjonalnego, wydajnego i łatwego w obsłudze systemu, który usprawni proces wypożyczania/rezerwowania samochodów, umożliwi klientom łatwą ocenę samochodów po wypożyczeniu oraz zapewni szybką i intuicyjną obsługę rezerwacji auta.

* 1. Funkcjonalność

W ramach projektu głównym celem jest wykorzystanie technologii Java oraz jej możliwości do stworzenia kompletnego systemu obsługującego proces wypożyczania samochodów w wypożyczalni.

Wykorzystanie języka Java w projekcie ma kilka korzyści:

1. Obiektowość:

Java jest językiem programowania obiektowego, co umożliwia modelowanie różnych aspektów wypożyczalni samochodowej jako obiekty. Możemy tworzyć klasy reprezentujące samochody, rezerwacje, wypożyczenia itp.

1. Bogate biblioteki:

Java oferuje szeroki zakres bibliotek standardowych i zewnętrznych, które mogą być wykorzystane w projekcie. Możemy skorzystać z tych bibliotek, aby zaoszczędzić czas i wysiłek w implementacji niektórych funkcjonalności.

1. Platformy:

Java jest znana ze swojej platformy niezależnej od systemu operacyjnego. Oznacza to, że programy napisane w Javie mogą być uruchamiane na różnych systemach operacyjnych, takich jak Windows, macOS czy Linux, bez konieczności pisania osobnych wersji dla każdego z nich. To zapewnia elastyczność i możliwość uruchamiania programu na różnych platformach.

1. Bezpieczeństwo:

Java ma wiele wbudowanych mechanizmów bezpieczeństwa, które są istotne w przypadku projektu wypożyczalni samochodowej. Bezpieczeństwo danych klientów, ochrona przed nieautoryzowanym dostępem czy zabezpieczenia transmisji danych są istotne dla biznesu wypożyczalni.

1. **KONFIGURACJA I URUCHOMIENIE**
   1. Instalacja Javy

Instalacja Javy:

* Upewnij się, że masz zainstalowaną Jave na swoim systemie. Możesz pobrać i zainstalować najnowszą wersję Javy z oficjalnej strony Oracle lub skorzystać z dystrybucji OpenJDK (w wersji 20 lub nowszej).

Środowisko programistyczne:

* Wybierz środowisko programistyczne, które preferujesz do otworzenia projektów w Javie. Przykładowymi popularnymi są np. InrelliJ IDEA, Eclipse i NetBeans.
* Pobierz i zainstaluj wybrany program, postępując zgodnie z instrukcjami producenta.
  1. Pobieranie i konfiguracja projektu

Otwarcie projektu:

* Uruchom wybrane środowisko programistyczne (np.IntelliJ IDEA, Eclipse, NetBeans).
* Wypierz opcję „Otwórz projekt” lub „Importuj projekt” z menu.
* Przejdź do lokalizacji, w której znajduję się folder projektu „Wypożyczalnia samochodów”.
* Wybierz folder projektu i zaakceptuj otwarcie.

Sprawdzanie konfiguracji projektu:

* Przed rozpoczęciem kompilacji i uruchamiania projektu, upewnij się, że konfiguracja projektu jest poprawnie ustawiona.
* Sprawdź, czy masz zainstalowaną odpowiednią wersję Javy i skonfigurowane ścieżki dostępu do niej w środowisku programistycznym.

Kompilacja projektu:

* Wybierz opcję kompilacji lub zbuduj projekt w środowisku programistycznym
* Środowisko automatycznie skompiluje kod źródłowy projektu.
* Upewnij się, że w wyniku kompilacji nie ma żadnych błędów ani ostrzeżeń.
  1. Uruchamianie projektu:
* Znajdź klasę zawierającą metodę main w projekcie, która będzie punktem wejścia do programu. W tym przypadku, klasa Wypożyczalnia.java odpowiada za kompilację bazowej części w konsoli.
* Kliknij prawym przyciskiem myszy na tę klasę i wybierz opcję "Uruchom" lub "Uruchom jako aplikację". To spowoduje uruchomienie programu w trybie tekstowym.
* Alternatywnie, możesz wybrać opcję "Uruchom" lub "Uruchom konfigurację" z menu i wybrać odpowiednią konfigurację uruchamiającą dla projektu. Upewnij się, że wybierasz konfigurację związana z Wypożyczalnia.java.

Warto zauważyć, że choć klasa Wypożyczalnia.java odpowiada za kompilację bazowej części programu w konsoli, programem bazowym jest klasa GUI.java. Aby skorzystać z funkcjonalności graficznego interfejsu użytkownika, po uruchomieniu programu w trybie tekstowym, należy uruchomić również GUI.java z poziomu interfejsu.

Przykładowe punkty do uwzględnienia przy otwieraniu, kompilacji i uruchamianiu projektu:

* W przypadku IntelliJ IDEA: Otwórz projekt -> Wybierz folder projektu -> Kliknij na przycisk "Run" lub użyj skrótu klawiaturowego Shift+F10, aby uruchomić program.
* W przypadku Eclipse: Otwórz perspektywę Java -> Kliknij prawym przyciskiem myszy na projekt w widoku "Package Explorer" -> Wybierz "Run As" -> "Java Application".
* W przypadku NetBeans: Otwórz projekt -> Kliknij prawym przyciskiem myszy na projekt w widoku "Projects" -> Wybierz "Run" lub użyj skrótu klawiaturowego Shift+F6, aby uruchomić projekt.

Po wykonaniu tych kroków powinieneś być w stanie otworzyć, skompilować i uruchomić projekt "Wypożyczalnia samochodów".

1. **SRUKTURY DANYCH**

3.1 Klasa AddCarRating

import javax.swing.\*;  
import java.awt.\*;  
import java.awt.event.ActionEvent;  
import java.awt.event.ActionListener;  
  
import static javax.swing.JOptionPane.showMessageDialog;  
  
public class AddCarRating implements ActionListener {  
 private final MyFrame myFrameNext = new MyFrame();  
 private final MyLoverPanel panelEnd;  
 private final MyButton buttonIdPassRating;  
 private final MyTextField textFieldID;  
 private final JLabel labelCheckIdRating;  
 private final JComboBox comboBoxRating;  
  
 AddCarRating(){  
 //Initialization of variables  
 myFrameNext.backItem.addActionListener(this);  
 JPanel panelStart = new JPanel();  
 JPanel panelMiddle = new JPanel();  
 panelEnd = new MyLoverPanel(" Dodaj ocene: ");  
 JLabel label = new JLabel("Dodaj ocene samochoduu: ");  
 JLabel LabelId = new JLabel("Wybierz ID samochodu: ");  
 textFieldID = new MyTextField();  
 JLabel LB = new JLabel("");  
 buttonIdPassRating = new MyButton(" Sprawdz ");  
 buttonIdPassRating.addActionListener((ActionListener) this);  
 labelCheckIdRating = new JLabel("");  
 String[] rating1\_5 = {"1","2","3","4","5"};  
 comboBoxRating = new JComboBox<String>(rating1\_5);  
 //Changing elements in panelStart  
 panelStart.setBackground(new Color(0, 200, 0));  
 //Changing elements in panelMiddle  
 panelMiddle.setBackground(new Color(0, 200, 0));  
 panelMiddle.setBorder(BorderFactory.createEmptyBorder(30, 30, 10, 30));  
 panelMiddle.setLayout(new GridLayout(0, 2, 10, 5));  
 buttonIdPassRating.setBackground(new Color(50, 120, 200));  
 comboBoxRating.setBackground(Color.BLACK);  
 comboBoxRating.setForeground(Color.GREEN);  
 comboBoxRating.setFont(new Font("Arctic", Font.PLAIN, 50));  
 comboBoxRating.setSize(new Dimension(250, 40));  
 comboBoxRating.setVisible(false);  
 //Changing elements in panelEnd  
 panelEnd.buttonBack.addActionListener(this);  
 panelEnd.buttonSecond.addActionListener( this);  
 panelEnd.buttonSecond.setVisible(false);  
 //Add elements to panels  
 panelStart.add(label, BorderLayout.CENTER);  
 panelMiddle.add(LabelId);  
 panelMiddle.add(textFieldID);  
 panelMiddle.add(LB);  
 panelMiddle.add(buttonIdPassRating);  
 panelMiddle.add(labelCheckIdRating);  
 panelMiddle.add(comboBoxRating);  
 //Add elements to frames  
 myFrameNext.add(panelStart, BorderLayout.NORTH);  
 myFrameNext.add(panelMiddle,BorderLayout.CENTER);  
 myFrameNext.add(panelEnd, BorderLayout.SOUTH);  
 }  
  
  
 @Override  
 public void actionPerformed(ActionEvent e){  
 if(e.getSource() ==myFrameNext.backItem){  
 myFrameNext.dispose();  
 new MenuGui();  
 }  
 if(e.getSource()==buttonIdPassRating){  
 if (textFieldID.getText().isEmpty()) {  
 comboBoxRating.setVisible(false);  
 panelEnd.buttonSecond.setVisible(false);  
 labelCheckIdRating.setText("Podaj liczbe identyfikatora");  
 }  
 else {  
 int ID = Integer.parseInt(textFieldID.getText());  
 if (GUI.funcSearchCar(ID) != null) {  
 labelCheckIdRating.setText("Na ile oceniasz pojazd? ");  
 comboBoxRating.setVisible(true);  
 panelEnd.buttonSecond.setVisible(true);  
 }  
 else {  
 labelCheckIdRating.setText("Nie ma takiego ID ");  
 comboBoxRating.setVisible(false);  
 panelEnd.buttonSecond.setVisible(false);  
 }  
 }  
 }  
 //Lower buttons  
 if(e.getSource()==panelEnd.buttonBack){  
 myFrameNext.dispose();  
 new MenuGui();  
 }  
 if(e.getSource()==panelEnd.buttonSecond){  
 int ID = Integer.parseInt(textFieldID.getText());  
 GUI.Car car = GUI.funcSearchCar(ID);  
 car.addRating(comboBoxRating.getSelectedIndex()+1);  
 showMessageDialog(myFrameNext,"Dodano ocene samochodu do bazy");  
 myFrameNext.dispose();  
 new MenuGui();  
 }  
 }  
}

Opis:

* Klasa ‘AddCarRating’ odpowiada za dodawanie oceny dla samochodów po jego wypożyczeniu.
* Implementuje interfejs ActionListener, aby obsługiwać zdarzenia związane z interakcją użytkownika.

Konstruktory i pola:

* Klasa ma jeden konstruktor, który inicjalizuje wszystkie potrzebne pola i tworzy interfejs użytkownika.

Pola te obejmują:

* myFrameNext: obiekt klasy MyFrame, reprezentujący główne okno programu.
* panelEnd: obiekt klasy MyLoverPanel, reprezentujący panel zawierający przyciski do nawigacji.
* buttonIdPassRating: obiekt klasy MyButton, reprezentujący przycisk do sprawdzenia poprawności identyfikatora samochodu.
* textFieldID: obiekt klasy MyTextField, reprezentujący pole tekstowe do wprowadzenia identyfikatora samochodu.
* labelCheckIdRating: obiekt klasy JLabel, wyświetlający komunikaty dotyczące poprawności identyfikatora samochodu.
* comboBoxRating: obiekt klasy JComboBox, służący do wyboru oceny samochodu.

Metody:

* actionPerformed(ActionEvent e): Jest to metoda implementująca interfejs   
    
  ActionListener. Obsługuje zdarzenia, takie jak kliknięcie przycisków.
* Metoda reaguje na kliknięcie przycisków związanych z nawigacją (powrót do menu) oraz przycisków związanych z dodawaniem oceny.
* Sprawdza poprawność wprowadzonego identyfikatora samochodu.
* Dodaje ocenę samochodu do wewnętrznej bazy danych.

Ta klasa zajmuje się obsługą interfejsu użytkownika, umożliwia wprowadzenie identyfikatora samochodu i przypisanie mu oceny.

3.2 Klasa AddCars

import javax.swing.\*;  
import java.awt.\*;  
import java.awt.event.ActionEvent;  
import java.awt.event.ActionListener;  
import javax.swing.JOptionPane;  
import static javax.swing.JOptionPane.showMessageDialog;  
  
public class AddCars implements ActionListener {  
 private final MyFrame myFrameNext= new MyFrame();  
 private final MyLoverPanel panelEnd;  
 private final MyTextField myTextFieldMark, myTextFieldModel, myTextFieldYearOfProduction, myTextFieldPriceForDay;  
 private final JCheckBox checkBoxAddCar;  
 AddCars() {  
 //Initialization of variables  
 myFrameNext.backItem.addActionListener(this);  
 JPanel panelStart = new JPanel();  
 JPanel panelMiddle = new JPanel();  
 panelEnd = new MyLoverPanel(" Zatwierdz ");  
 JLabel titleLabel = new JLabel("Podaj parametry samochodu:");  
 JLabel Label1 = new JLabel("Podaj marke samochoduu: \t");  
 myTextFieldMark = new MyTextField();  
 JLabel Label2 = new JLabel("Podaj model samochodu: \t");  
 myTextFieldModel = new MyTextField();  
 myTextFieldYearOfProduction = new MyTextField();  
 JLabel Label4 = new JLabel("Podaj rok produkcji: \t");  
 JLabel Label3 = new JLabel("Podaj cene za dzien wynajmu: \t");  
 myTextFieldPriceForDay = new MyTextField();  
 checkBoxAddCar = new JCheckBox();  
 //Changing elements in panelStart  
 panelStart.setBackground(Color.GREEN);  
 //Changing elements in panelMiddle  
 panelMiddle.setBackground(new Color(0, 200, 0));  
 panelMiddle.setBorder(BorderFactory.createEmptyBorder(30, 30, 10, 30));  
 panelMiddle.setLayout(new GridLayout(0, 2, 10, 5));  
 Label1.setBackground(Color.GREEN);  
// myTextFieldMark.setText("OPEL");  
 Label2.setBackground(Color.GREEN);  
// myTextFieldModel.setText("ASTRA");  
 Label4.setBackground(Color.GREEN);  
// myTextFieldYearOfProduction.setText("2020");  
 Label3.setBackground(Color.GREEN);  
// myTextFieldPriceForDay.setText("100");  
 checkBoxAddCar.setText("Nie jestem robotem :)");  
 checkBoxAddCar.setBackground(Color.GREEN);  
 checkBoxAddCar.setFocusable(false);  
 panelMiddle.setBackground(Color.GREEN);  
 //Changing elements in panelEnd  
 panelEnd.setBackground(new Color(0, 200, 0));  
 panelEnd.setBorder(BorderFactory.createEmptyBorder(30, 30, 10, 30));  
 panelEnd.setLayout(new GridLayout(0, 4, 10, 5));  
 panelEnd.buttonBack.addActionListener(this);  
 panelEnd.buttonSecond.addActionListener((ActionListener) this);  
 //Add elements to panels  
 panelStart.add(titleLabel);  
 panelMiddle.add(Label1);  
 panelMiddle.add(myTextFieldMark);  
 panelMiddle.add(Label2);  
 panelMiddle.add(myTextFieldModel);  
 panelMiddle.add(Label4);  
 panelMiddle.add(myTextFieldYearOfProduction);  
 panelMiddle.add(Label3);  
 panelMiddle.add(myTextFieldPriceForDay);  
 panelMiddle.add(checkBoxAddCar);  
 //Add elements to frames  
 myFrameNext.add(panelStart, BorderLayout.NORTH);  
 myFrameNext.add(panelMiddle, BorderLayout.CENTER);  
 myFrameNext.add(panelEnd, BorderLayout.SOUTH);  
 }  
 public static void funcAddCar(String mark, String model, double price, short yearOfProduction, MyFrame frame) {  
 GUI.Car Car = new GUI.Car(mark, model, price, yearOfProduction);  
 GUI.Cars.add(Car);  
 showMessageDialog(frame,"Dodano samochod do bazy");  
 }  
 @Override  
 public void actionPerformed(ActionEvent e) {  
 if(e.getSource() == myFrameNext.backItem){  
 System.out.println("Nie powiodło sie");  
 myFrameNext.dispose();  
 new MenuGui();  
 }  
 if (e.getSource()==panelEnd.buttonSecond) {  
 if (!(!checkBoxAddCar.isSelected() | myTextFieldMark.getText().isEmpty() | myTextFieldModel.getText().isEmpty() | myTextFieldYearOfProduction.getText().isEmpty() | myTextFieldPriceForDay.getText().isEmpty())) {  
 double price = Double.parseDouble(myTextFieldPriceForDay.getText());  
 short year = Short.parseShort(myTextFieldYearOfProduction.getText());  
 funcAddCar(myTextFieldMark.getText(), myTextFieldModel.getText(),price,year,myFrameNext);  
 if (checkBoxAddCar.isSelected())  
 myFrameNext.dispose();  
 new MenuGui();  
 } else  
 JOptionPane.showMessageDialog(myFrameNext,"Uzupelnij puste pola");  
 }  
 if(e.getSource()==panelEnd.buttonBack){  
 myFrameNext.dispose();  
 new MenuGui();  
 }  
 }  
}

Opis:

* Klasa "AddCars" odpowiada za dodawanie nowych samochodów do systemu wypożyczalni.
* Implementuje interfejs ActionListener, aby obsługiwać zdarzenia związane z interakcją użytkownika.

Konstruktory i pola:

* Klasa ma jeden konstruktor, który inicjalizuje wszystkie potrzebne pola i tworzy interfejs użytkownika.

Pola te obejmują:

* myFrameNext: obiekt klasy MyFrame, reprezentujący główne okno programu.
* panelEnd: obiekt klasy MyLoverPanel, reprezentujący panel zawierający przyciski do nawigacji.
* myTextFieldMark, myTextFieldModel, myTextFieldYearOfProduction, myTextFieldPriceForDay: obiekty klasy MyTextField, służące do wprowadzania danych dotyczących nowego samochodu.
* checkBoxAddCar: obiekt klasy JCheckBox, służący do potwierdzenia, że użytkownik nie jest robotem.

Metody:

* actionPerformed(ActionEvent e): Jest to metoda implementująca interfejs ActionListener. Obsługuje zdarzenia, takie jak kliknięcie przycisków.
* Metoda reaguje na kliknięcie przycisków związanych z nawigacją (powrót do menu) oraz przycisku zatwierdzającego dodanie nowego samochodu.
* Sprawdza poprawność wprowadzonych danych dotyczących samochodu.
* Dodaje nowy samochód do wewnętrznej bazy danych.

Metoda funcAddCar:

* Metoda ta jest statyczną metodą klasy "AddCars".
* Służy do dodawania nowego samochodu do bazy danych.
* Tworzy obiekt klasy GUI.Car na podstawie wprowadzonych danych.
* Dodaje nowy samochód do listy samochodów w klasie GUI.Cars.
* Wyświetla komunikat potwierdzający dodanie samochodu.

Ta klasa zajmuje się obsługą interfejsu użytkownika, umożliwia wprowadzenie danych dotyczących nowego samochodu i dodanie go do systemu wypożyczalni.

3.3 Klasa CalculateAmountOfRent

import javax.swing.\*;  
import java.awt.\*;  
import java.awt.event.ActionEvent;  
import java.awt.event.ActionListener;  
  
public class CalculateAmountOfRent implements ActionListener {  
 private final MyFrame myFrameNext = new MyFrame();  
 private final MyLoverPanel panelEnd;  
 private final MyTextField textFieldID, textFieldDays;  
 private final MyButton buttonIdPass;  
 private final JLabel labelCheck, labelDays, labelCost;  
  
 CalculateAmountOfRent() {  
 //Initialization of variables  
 myFrameNext.backItem.addActionListener(this);  
 JPanel panelStart = new JPanel();  
 JPanel panel = new JPanel();  
 panelEnd = new MyLoverPanel(" Oblicz ");  
 JLabel label = new JLabel("Oblicz kwote wynajmu: ");  
 JLabel LabelId = new JLabel("Wybierz ID samochodu: ");  
 textFieldID = new MyTextField();  
 JLabel LB = new JLabel("");  
 buttonIdPass = new MyButton(" Sprawdz ");  
 labelCheck = new JLabel("");  
 labelDays = new JLabel("");  
 labelCost = new JLabel("");  
 JLabel LBCalculate = new JLabel("");  
 //Changing elements in panelStart  
 panelStart.setBackground(new Color(0, 200, 0));  
 //Changing elements in panelMiddle  
 panel.setBackground(new Color(0, 200, 0));  
 panel.setBorder(BorderFactory.createEmptyBorder(30, 30, 10, 30));  
 panel.setLayout(new GridLayout(0, 2, 10, 5));  
 buttonIdPass.addActionListener((ActionListener) this);  
 textFieldDays = new MyTextField();  
 textFieldDays.setVisible(false);  
 panelEnd.buttonSecond.addActionListener((ActionListener) this);  
 panelEnd.buttonSecond.setVisible(false);  
 //Changing elements in panelEnd  
 panelEnd.setBackground(new Color(0, 200, 0));  
 panelEnd.setBorder(BorderFactory.createEmptyBorder(30, 30, 10, 30));  
 panelEnd.setLayout(new GridLayout(0, 4, 10, 5));  
 panelEnd.buttonBack.addActionListener((ActionListener) this);  
 //Add elements to panels  
 panelStart.add(label, BorderLayout.CENTER);  
 panel.add(LabelId);  
 panel.add(textFieldID);  
 panel.add(LB);  
 panel.add(buttonIdPass);  
 panel.add(labelCheck);  
 panel.add(textFieldDays);  
 panel.add(labelDays);  
 panel.add(labelCost);  
 //Add elements to frames  
 myFrameNext.add(panelStart, BorderLayout.NORTH);  
 myFrameNext.add(panel, BorderLayout.CENTER);  
 myFrameNext.add(panelEnd, BorderLayout.SOUTH);  
 }  
  
 @Override  
 public void actionPerformed(ActionEvent e) {  
 if(e.getSource() == myFrameNext.backItem){  
 myFrameNext.dispose();  
 new MenuGui();  
 }  
 if(e.getSource()==buttonIdPass) {  
 if (textFieldID.getText().isEmpty()) {  
 textFieldDays.setVisible(false);  
 panelEnd.buttonSecond.setVisible(false);  
 labelCheck.setText("Podaj liczbe identyfikatora");  
 labelDays.setText("");  
 labelCost.setText("");  
 }  
 else {  
 int ID = Integer.parseInt(textFieldID.getText());  
 if (GUI.funcSearchCar(ID) != null) {  
  
 labelCheck.setText("Na ile dni chcesz wypozyczyc: ");  
 textFieldDays.setVisible(true);  
 panelEnd.buttonSecond.setVisible(true);  
  
 }  
 else {  
 labelCheck.setText("Nie ma takiego ID ");  
 textFieldDays.setVisible(false);  
 panelEnd.buttonSecond.setVisible(false);  
 labelDays.setText("");  
 labelCost.setText("");  
 }  
 }  
 }  
 if(e.getSource()==panelEnd.buttonSecond){  
 int ID = Integer.parseInt(textFieldID.getText());  
 GUI.Car Car = GUI.funcSearchCar(ID);  
 int days = Integer.parseInt(textFieldDays.getText());  
 double price = Car.funcCalculateTheCostOfRent(myFrameNext, days);  
 labelDays.setText("Kwota za " + Car.getMark() + " " + Car.getModel() + " wynosi: ");  
 labelCost.setText(String.format("%.2f PLN", price));  
 }  
 if(e.getSource()==panelEnd.buttonBack){  
 myFrameNext.dispose();  
 new MenuGui();  
 }  
 }  
}

Opis:

* Klasa "CalculateAmountOfRent" odpowiada za obliczanie kwoty wynajmu samochodu na podstawie wprowadzonych danych.
* Implementuje interfejs ActionListener, aby obsługiwać zdarzenia związane z interakcją użytkownika.

Konstruktory i pola:

* Klasa ma jeden konstruktor, który inicjalizuje wszystkie potrzebne pola i tworzy interfejs użytkownika.

Pola te obejmują:

* myFrameNext: obiekt klasy MyFrame, reprezentujący główne okno programu.
* panelEnd: obiekt klasy MyLoverPanel, reprezentujący panel zawierający przyciski do nawigacji.
* textFieldID, textFieldDays: obiekty klasy MyTextField, służące do wprowadzania danych, takich jak ID samochodu i liczba dni wynajmu.
* buttonIdPass: obiekt klasy MyButton, służący do potwierdzenia wprowadzenia ID samochodu.
* labelCheck, labelDays, labelCost: obiekty klasy JLabel, służące do wyświetlania komunikatów związanych z wprowadzanymi danymi i obliczoną kwotą wynajmu.

Metody:

* actionPerformed(ActionEvent e): Jest to metoda implementująca interfejs ActionListener. Obsługuje zdarzenia, takie jak kliknięcie przycisków.
* Metoda reaguje na kliknięcie przycisków związanych z nawigacją (powrót do menu) oraz przycisków do potwierdzenia wprowadzenia ID i obliczenia kwoty wynajmu.
* Sprawdza poprawność wprowadzonych danych i wywołuje odpowiednie metody z klasy GUI w celu wyszukania samochodu o podanym ID i obliczenia kwoty wynajmu.
* Wyświetla obliczoną kwotę wynajmu.

Ta klasa zajmuje się obsługą interfejsu użytkownika, umożliwia wprowadzenie ID samochodu oraz liczby dni wynajmu i oblicza kwotę wynajmu na podstawie tych danych.

3.4 Klasa GUI

import java.text.DateFormat;  
import java.text.ParseException;  
import java.text.SimpleDateFormat;  
import java.util.ArrayList;  
import java.util.Calendar;  
import java.util.Date;  
import java.sql.\*;  
import static javax.swing.JOptionPane.showMessageDialog;  
  
public class GUI{  
 public static void main(String[] args){ new GUI(); }  
 public GUI(){  
 //Car.createDatabase();  
 new MenuGui();  
 }  
 public static ArrayList<Car> Cars = new ArrayList<>();  
 private static int nextId = 1;  
  
 public static Car funcSearchCar(int id) {  
 for (Car Car : Cars) {  
 if (Car.getId() == id) {  
 return Car;  
 }  
 }  
 return null;  
 }  
 static class Car {  
 public static Connection connection;  
 private final int id;  
 private final String mark;  
 private final String model;  
 private final double price;  
 private final short yearOfProduction;  
 public final ArrayList<Integer> ratings;  
 public ArrayList<String> rentCar = new ArrayList<>();  
 public Car(String mark, String model, double price, short yearOfProduction) {  
 this.id = nextId++;  
 this.mark = mark;  
 this.model = model;  
 this.price = price;  
 this.yearOfProduction = yearOfProduction;  
 this.ratings = new ArrayList<>();  
 }  
 /\*  
 public static void createDatabase() {  
 try {  
 // Rejestrujemy sterownik JDBC  
 Class.forName("oracle.jdbc.driver.OracleDriver");  
  
 // Ustanowienie połączenia z bazą danych  
 connection = DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:xe", "username", "password");  
 } catch (ClassNotFoundException | SQLException e) {  
 e.printStackTrace();  
 }  
 }  
 public static void connectToDatabase() {  
 try {  
 // Ustanowienie połączenia z bazą danych  
 connection = DriverManager.getConnection("jdbc:mysql://localhost:3306/cars\_database", "borgon1999@gmail.com", "!Student2021");  
 } catch (SQLException e) {  
 e.printStackTrace();  
 }  
 }  
  
 public void insertIntoDatabase() {  
 try {  
 // Wstawienie danych samochodu do tabeli Car w bazie danych  
 PreparedStatement preparedStatement = connection.prepareStatement(  
 "INSERT INTO Car (id, mark, model, price, yearOfProduction) VALUES (?, ?, ?, ?, ?)");  
 preparedStatement.setInt(1, id);  
 preparedStatement.setString(2, mark);  
 preparedStatement.setString(3, model);  
 preparedStatement.setDouble(4, price);  
 preparedStatement.setShort(5, yearOfProduction);  
 preparedStatement.executeUpdate();  
  
 preparedStatement.close();  
 } catch (SQLException e) {  
 e.printStackTrace();  
 }  
 }  
 public static void addCarToDatabase(String mark, String model, double price, short yearOfProduction) {  
 connectToDatabase();  
  
 Car car = new Car(mark, model, price, yearOfProduction);  
 car.insertIntoDatabase();  
 }\*/  
  
  
 public int getId() { return id; }  
 public String getMark() {  
 return mark;  
 }  
 public String getModel() {  
 return model;  
 }  
 public double getPrice() {  
 return price;  
 }  
 public short getYearOfProduction() { return yearOfProduction;}  
 public double funcCalculateTheCostOfRent(MyFrame frame, int days) {  
 if(days < 7) return price \* days;  
 if(days < 14){  
 String messege = "Dodano rabat 10%. Za wynajem na " + days + " dni ";  
 showMessageDialog(frame, messege);  
 return price \* days \* 0.9;  
 }  
 String messege = "Dodano rabat 15%. Za wynajem na " + days + " dni";  
 showMessageDialog(frame, messege);  
 return price \* days \* 0.85;  
 }  
 public void addRating(int rating) {  
 ratings.add(rating);  
 }  
 public double averageRating() {  
 if (ratings.isEmpty()) {  
 return 0;  
 }  
 double sum = 0;  
 for (int price : ratings) {  
 sum += price;  
 }  
 return sum / ratings.size();  
 }  
 public void AddRentCarFromLoad(String rentDays, int days){  
 String add = rentDays + " " + String.format("%4d", days);  
 System.out.println(add.length());  
 rentCar.add(add);  
 }  
 public void AddRentCar(MyFrame frame, String rentDays, int days){  
 for(String daysInRent : rentCar){  
 if(checkTheDate(frame, daysInRent)){  
 String rent = daysInRent;  
 showMessageDialog(frame, rent);  
 return;  
 }  
 }  
 String add = rentDays + " " + String.format("%4d", days);  
 rentCar.add(add);  
 }  
  
 public boolean checkTheDate(MyFrame frame,String startRentDays){  
 String start = startRentDays.substring(0, 10);  
 DateFormat dateFormat = new SimpleDateFormat("dd.MM.yyyy");  
 Date startDate = null;  
 try {  
 startDate = dateFormat.parse(startRentDays.substring(0, 10));  
 }  
 catch (ParseException e) {  
 showMessageDialog(frame, "Ten termin jest zajety!");  
 throw new RuntimeException(e);  
 }  
 Calendar endDate = (Calendar) startDate.clone();  
 int days = Integer.parseInt(startRentDays.substring(startRentDays. length() - 4));  
 endDate.add(Calendar.DATE, days);  
 for(int i=0;i<days;i++){  
 endDate.add(Calendar.DATE, 1);  
 }  
 return true;  
 }  
 }

}

Opis:

* Klasa GUI jest główną klasą programu. Jej konstruktor inicjalizuje interfejs użytkownika i uruchamia aplikację.
* Wewnętrzna klasa Car reprezentuje samochód w wypożyczalni. Ma pola takie jak id, mark, model, price, yearOfProduction oraz listy ocen (ratings) i wynajmów (rentCar).
* Klasa Car posiada metody do obliczania kosztu wynajmu, dodawania ocen i zarządzania wynajmami.

Metody klasy GUI:

* funcSearchCar(int id): Metoda statyczna, która przeszukuje listę samochodów (Cars) i zwraca obiekt Car o podanym ID. Jeśli nie znajduje samochodu, zwraca wartość null.

Metody klasy Car:

* Car(String mark, String model, double price, short yearOfProduction): Konstruktor tworzy nowy obiekt Car i inicjalizuje jego pola.
* funcCalculateTheCostOfRent(MyFrame frame, int days): Metoda oblicza koszt wynajmu samochodu na podstawie liczby dni. Zwraca obliczoną wartość.
* addRating(int rating): Metoda dodaje ocenę samochodu do listy ocen (ratings).
* averageRating(): Metoda oblicza średnią ocen samochodu na podstawie listy ocen. Zwraca średnią wartość.
* AddRentCarFromLoad(String rentDays, int days): Metoda dodaje informacje o wynajmie samochodu na podstawie wczytanych danych (nie jest używana w tym kodzie).
* AddRentCar(MyFrame frame, String rentDays, int days): Metoda dodaje informacje o wynajmie samochodu na podstawie wprowadzonych danych. Sprawdza również dostępność terminu wynajmu.
* checkTheDate(MyFrame frame, String startRentDays): Metoda sprawdza dostępność terminu wynajmu na podstawie podanego daty rozpoczęcia wynajmu i liczby dni. Zwraca wartość logiczną w zależności od dostępności terminu.

3.5 Klasa MenuGui

import javax.swing.\*;  
import java.awt.\*;  
import java.awt.event.ActionEvent;  
import java.awt.event.ActionListener;  
  
public class MenuGui implements ActionListener {  
 private final MyFrame myFrameGUI;  
 private final JButton buttonAddCar,buttonShowCar,buttonReserveACars,buttonCalculateCost,buttonAddRatingCar,buttonShowCarsRating,exitButton;  
 MenuGui() {  
 myFrameGUI = new MyFrame();  
 myFrameGUI.editMenu.setVisible(false);  
 JLabel label = new JLabel("Wypozyczalnia samochodow firmy: Firma ");  
 buttonAddCar = new JButton("Dodaj samochod");  
 buttonShowCar = new JButton("Wyswietl samochody");  
 buttonReserveACars = new JButton("Rezerwuj samochod");  
 buttonCalculateCost = new JButton("Oblicz koszt wynajmu samochodu");  
 buttonAddRatingCar = new JButton("Ocen samochod");  
 buttonShowCarsRating = new JButton("Wyswietl oceny samochodow");  
 exitButton = new JButton("Wyjscie");  
 JPanel panel = new JPanel();  
 myFrameGUI.add(panel, BorderLayout.CENTER);  
 panel.setBackground(new Color(0, 200, 0));  
 panel.setBorder(BorderFactory.createEmptyBorder(30, 30, 10, 30));  
 panel.setLayout(new GridLayout(0, 1, 10, 5));  
 panel.add(label, BorderLayout.CENTER);  
 panel.add(buttonAddCar);  
 buttonAddCar.addActionListener(this);  
 buttonAddCar.setBorder(BorderFactory.createEtchedBorder());  
 buttonAddCar.setBackground(new Color(50, 120, 200));  
 panel.add(buttonShowCar);  
 buttonShowCar.addActionListener(this);  
 buttonShowCar.setBorder(BorderFactory.createEtchedBorder());  
 buttonShowCar.setBackground(new Color(50, 120, 200));  
 panel.add(buttonReserveACars);  
 buttonReserveACars.addActionListener(this);  
 buttonReserveACars.setBorder(BorderFactory.createEtchedBorder());  
 buttonReserveACars.setBackground(new Color(50, 120, 200));  
 panel.add(buttonCalculateCost);  
 buttonCalculateCost.addActionListener(this);  
 buttonCalculateCost.setBorder(BorderFactory.createEtchedBorder());  
 buttonCalculateCost.setBackground(new Color(50, 120, 200));  
 panel.add(buttonAddRatingCar);  
 buttonAddRatingCar.addActionListener(this);  
 buttonAddRatingCar.setBorder(BorderFactory.createEtchedBorder());  
 buttonAddRatingCar.setBackground(new Color(50, 120, 200));  
 panel.add(buttonShowCarsRating);  
 buttonShowCarsRating.addActionListener(this);  
 buttonShowCarsRating.setBorder(BorderFactory.createEtchedBorder());  
 buttonShowCarsRating.setBackground(new Color(50, 120, 200));  
 panel.add(exitButton);  
 exitButton.addActionListener(this);  
 exitButton.setBorder(BorderFactory.createEtchedBorder());  
 exitButton.setBackground(new Color(50, 120, 200));  
 }  
 @Override  
 public void actionPerformed(ActionEvent e) {  
 if(e.getSource()==buttonAddCar){  
 myFrameGUI.dispose();  
 new AddCars();  
 }  
 if(e.getSource()==buttonShowCar){  
 myFrameGUI.dispose();  
 new ShowCars();  
 }  
 if(e.getSource()==buttonReserveACars){  
 myFrameGUI.dispose();  
 new AddRentCar();  
 }  
 if(e.getSource()==buttonCalculateCost){  
 myFrameGUI.dispose();  
 new CalculateAmountOfRent();  
 }  
 if(e.getSource()==buttonAddRatingCar){  
 myFrameGUI.dispose();  
 new AddCarRating();  
 }  
 if(e.getSource()==buttonShowCarsRating){  
 myFrameGUI.dispose();  
 new ShowRatingsCars();  
 }  
 if (e.getSource() == exitButton)  
 System.exit(0);  
 }  
}

Opis:

* Klasa MenuGui tworzy interfejs użytkownika dla menu głównego aplikacji.
* Klasa implementuje interfejs ActionListener, aby obsługiwać zdarzenia przycisków.

Konstruktor:

* Konstruktor MenuGui() tworzy obiekt klasy MyFrame (niestandardowy obiekt JFrame) i inicjalizuje przyciski oraz etykiety dla interfejsu użytkownika.
* Przyciski są dodawane do panelu panel za pomocą menedżera układu GridLayout.
* Każdy przycisk ma przypisanego ActionListenera, który obsługuje zdarzenia kliknięcia przycisku.

Metoda actionPerformed(ActionEvent e):

* Ta metoda obsługuje zdarzenia kliknięcia przycisku.
* Na podstawie źródła zdarzenia (e.getSource()) sprawdzane jest, który przycisk został kliknięty.
* Na podstawie klikniętego przycisku, okno myFrameGUI jest zamykane, a otwierane jest nowe okno odpowiadające wybranej funkcjonalności.

Przykładowo, po kliknięciu przycisku "Dodaj samochód", bieżące okno zostaje zamknięte, a otwarte zostaje nowe okno klasy AddCars, które umożliwia dodawanie samochodów do wypożyczalni.

3.6 Klasa MyTextField

import javax.swing.\*;  
import java.awt.\*;  
  
public class MyTextField extends JTextField {  
 MyTextField(){  
 this.setSize(new Dimension(250, 40));  
 this.setFont(new Font("Arctic", Font.PLAIN, 50));  
 this.setForeground(Color.GREEN);  
 this.setBackground(Color.BLACK);  
 this.setCaretColor(Color.WHITE);  
 }  
}

Opis:

* Klasa MyTextField dziedziczy po klasie JTextField i dostarcza dostosowane ustawienia dla pola tekstowego.

Konstruktor:

* Konstruktor MyTextField() inicjalizuje pole tekstowe.

Ustawienia dla pola tekstowego obejmują:

* Rozmiar pola tekstowego (setSize(new Dimension(250, 40))), gdzie szerokość wynosi 250 pikseli, a wysokość 40 pikseli.
* Czcionkę tekstu (setFont(new Font("Arctic", Font.PLAIN, 50))), gdzie używana jest czcionka "Arctic" o stylu "PLAIN" i rozmiarze 50 punktów.
* Kolor tekstu (setForeground(Color.GREEN)), gdzie tekst będzie miał kolor zielony.
* Kolor tła pola tekstowego (setBackground(Color.BLACK)), gdzie tło pola tekstowego będzie miało kolor czarny.
* Kolor kursora (setCaretColor(Color.WHITE)), gdzie kursor w polu tekstowym będzie miał kolor biały.

Klasa MyTextField może być używana jako niestandardowe pole tekstowe w interfejsie użytkownika aplikacji, aby dostosować wygląd i zachowanie pola tekstowego.

3.7 Klasa ShowCars

import javax.swing.\*;  
import java.awt.\*;  
import java.awt.event.ActionEvent;  
import java.awt.event.ActionListener;  
import java.util.ArrayList;  
  
public class ShowCars implements ActionListener {  
 private final MyFrame myFrameNext = new MyFrame();  
 MyLoverPanel panelEnd;  
 final ArrayList<String> listCars;  
 final ArrayList<JLabel> labelListCars;  
 ShowCars(){  
 //Initialization of variables  
 myFrameNext.backItem.addActionListener(this);  
 JPanel panelStart = new JPanel();  
 JPanel panelMiddle = new JPanel();  
 panelEnd = new MyLoverPanel();  
 JLabel centerLabel = new JLabel("Obecne samochody:");  
 listCars = funcShowCars();  
 labelListCars = new ArrayList<>();  
 //Changing elements in panelStart  
 panelStart.setBackground(Color.GREEN);  
 //Changing elements in panel  
 panelMiddle.setBackground(Color.GREEN);  
 panelMiddle.setLayout(new BoxLayout(panelMiddle,BoxLayout.Y\_AXIS));  
 //Changing elements in panelEnd  
 panelEnd.buttonBack.addActionListener((ActionListener) this);  
 //Add elements to panels  
 panelStart.add(centerLabel);  
 for (String car : listCars) {  
 JLabel labelCar = new JLabel(car);  
 labelListCars.add(labelCar);  
 panelMiddle.add(labelCar);  
 }  
 panelEnd.add(panelEnd.buttonBack);  
 //Add elements to frames  
 myFrameNext.add(panelStart, BorderLayout.NORTH);  
 myFrameNext.add(panelMiddle,BorderLayout.CENTER);  
 myFrameNext.add(panelEnd, BorderLayout.SOUTH);  
 }  
 public static ArrayList<String> funcShowCars() {  
 ArrayList<String> listCars = new ArrayList<>();  
 if (GUI.Cars.isEmpty())  
 listCars.add("Nie dodano samochodow do bazy");  
 else{  
 for(GUI.Car car : GUI.Cars) {  
 listCars.add("ID: " + car.getId() + " " + car.getMark() + " " + car.getModel() + " " + car.getYearOfProduction()) ;  
 }  
 }  
 return listCars;  
 }  
 @Override  
 public void actionPerformed(ActionEvent e){  
 if(e.getSource() == myFrameNext.backItem){  
 myFrameNext.dispose();  
 new MenuGui();  
 }  
 if(e.getSource()==panelEnd.buttonBack){  
 myFrameNext.dispose();  
 new MenuGui();  
 }  
 }  
}

Opis:

* Klasa ShowCars implementuje interfejs ActionListener i odpowiedzialna jest za wyświetlanie informacji o samochodach w interfejsie użytkownika.

Konstruktor:

* Konstruktor klasy ShowCars inicjalizuje okno MyFrame oraz panele panelStart, panelMiddle i panelEnd. Tworzy również etykietę centerLabel i listę samochodów listCars przy użyciu metody funcShowCars(). Dodaje odpowiednie elementy do paneli i ramki.

Metody:

* Metoda funcShowCars() jest statyczną metodą, która zwraca listę samochodów w postaci napisów. Jeśli lista GUI.Cars (lista samochodów) jest pusta, dodaje informację o braku samochodów do listy. W przeciwnym razie dla każdego samochodu tworzy napis zawierający jego ID, markę, model i rok produkcji, a następnie dodaje ten napis do listy.
* Metoda actionPerformed() obsługuje akcje, takie jak kliknięcie przycisku "Back" w panelu panelEnd lub wybór opcji "Back" z menu myFrameNext.backItem. W obu przypadkach zamyka bieżące okno myFrameNext i tworzy nowe okno MenuGui().

Klasa ShowCars umożliwia wyświetlanie informacji o samochodach w interfejsie użytkownika oraz nawigację do innych funkcjonalności poprzez przyciski "Back".

3.8 Klasa MyLoverPanel

import javax.swing.\*;  
import java.awt.GridLayout;  
import java.awt.Color;  
  
public class MyLoverPanel extends JPanel{  
 static JButton buttonBack, buttonSecond;  
 MyLoverPanel(String secButton){  
 super();  
 this.setBackground(new Color(0, 235, 0));  
 this.setBorder(BorderFactory.createEmptyBorder(30, 30, 10, 30));  
 this.setLayout(new GridLayout(0, 4, 10, 5));  
 buttonBack = new JButton(" Powrot ");  
 buttonBack.setBackground(new Color(255,100,100));  
 buttonBack.setBorder(BorderFactory.createEtchedBorder());  
 buttonBack.setSize(250,20);  
 buttonSecond = new JButton(secButton);  
 buttonSecond.setBackground(new Color(0, 255, 0));  
 buttonSecond.setBorder(BorderFactory.createEtchedBorder());  
 buttonSecond.setSize(250,20);  
 JLabel lb1 = new JLabel("") , lb2 = new JLabel("");  
 this.add(buttonBack);  
 this.add(lb1);  
 this.add(lb2);  
 this.add(buttonSecond);  
 }  
 MyLoverPanel(){  
 this.setBackground(new Color(00, 235, 0));  
 this.setBorder(BorderFactory.createEmptyBorder(30, 30, 10, 30));  
 this.setLayout(new GridLayout(0, 4, 10, 5));  
 buttonBack = new JButton(" Powrot ");  
 buttonBack.setBackground(new Color(255,100,100));  
 buttonBack.setBorder(BorderFactory.createEtchedBorder());  
 buttonBack.setSize(250,20);  
 this.add(buttonBack);  
 }  
}

Konstruktory:

* Konstruktor MyLoverPanel(String secButton) inicjalizuje panel dolny z dwoma przyciskami.

Ustawienia dla panelu obejmują:

* Tło panelu (setBackground(new Color(0, 235, 0))), gdzie tło będzie miało kolor zielony.
* Marginesy panelu (setBorder(BorderFactory.createEmptyBorder(30, 30, 10, 30))).
* Układ panelu (setLayout(new GridLayout(0, 4, 10, 5))), gdzie elementy będą rozmieszczone w siatce o 4 kolumnach i zmiennej liczbie wierszy.
* Inicjalizowane są przyciski buttonBack i buttonSecond.

Ustawienia dla przycisku buttonBack obejmują:

* Tekst przycisku (" Powrót ").
* Tło przycisku (setBackground(new Color(255,100,100))), gdzie tło będzie miało kolor czerwony.
* Ramkę przycisku (setBorder(BorderFactory.createEtchedBorder())).
* Rozmiar przycisku (setSize(250,20)).

Ustawienia dla przycisku buttonSecond obejmują:

* Tekst przycisku (wartość przekazana jako argument secButton).
* Tło przycisku (setBackground(new Color(0, 255, 0))), gdzie tło będzie miało kolor zielony.
* Ramkę przycisku (setBorder(BorderFactory.createEtchedBorder())).
* Rozmiar przycisku (setSize(250,20)).
* Konstruktor MyLoverPanel() inicjalizuje panel dolny z jednym przyciskiem.
* Ustawienia dla panelu i przycisku buttonBack są takie same jak w poprzednim konstruktorze.
* Dodawany jest tylko przycisk buttonBack do panelu.

Klasa MyLoverPanel umożliwia tworzenie panelu dolnego z przyciskami w interfejsie użytkownika. Przyciski mogą być dostosowane pod względem tekstu, tła i rozmiaru.

3.9 Klasa ShowRatingCars

import javax.swing.\*;  
import java.awt.\*;  
import java.awt.event.ActionEvent;  
import java.awt.event.ActionListener;  
import java.util.ArrayList;  
  
public class ShowRatingsCars implements ActionListener {  
 private final MyFrame myFrameNext = new MyFrame();  
 private final MyLoverPanel panelEnd;  
 ShowRatingsCars(){  
 //Initialization of variables  
 myFrameNext.backItem.addActionListener(this);  
 JPanel panelStart = new JPanel();  
 JPanel panelMiddle = new JPanel();  
 panelEnd = new MyLoverPanel();  
 JLabel centerLabel = new JLabel("Oceny samochodow:");  
 ArrayList<String> ratingCars = funcRatingsCars();  
 ArrayList<JLabel> labelRatingsCars = new ArrayList<>();  
 //Changing elements in panelStart  
 panelStart.setBackground(Color.GREEN);  
 //Changing elements in panelMiddle  
 panelMiddle.setBackground(Color.GREEN);  
 panelMiddle.setLayout(new BoxLayout(panelMiddle,BoxLayout.Y\_AXIS));  
 //Changing elements in panelEnd  
 panelEnd.buttonBack.addActionListener(this);  
 //Add elements to panels  
 panelStart.add(centerLabel);  
 for (String car : ratingCars){  
 JLabel labelRatingCar = new JLabel(car);  
 labelRatingsCars.add(labelRatingCar);  
 panelMiddle.add(labelRatingCar);  
 }  
 //Add elements to frames  
 myFrameNext.add(panelStart, BorderLayout.NORTH);  
 myFrameNext.add(panelMiddle,BorderLayout.CENTER);  
 myFrameNext.add(panelEnd, BorderLayout.SOUTH);  
  
 }  
 private static ArrayList<String> funcRatingsCars(){  
 ArrayList<String> ratingsCars = new ArrayList<>();  
 double averageRatings;  
 if (GUI.Cars.isEmpty())  
 ratingsCars.add("Nie dodano samochodow do bazy");  
 else{  
 for(GUI.Car car : GUI.Cars) {  
 averageRatings = car.averageRating();  
 ratingsCars.add(String.format("ID: %d \t%s\t %s \t Ocena: %1.2f \n",car.getId(), car.getMark(), car.getModel(), averageRatings));  
 }  
 }  
 return ratingsCars;  
 }  
 @Override  
 public void actionPerformed(ActionEvent e){  
 if(e.getSource() == myFrameNext.backItem){  
 myFrameNext.dispose();  
 new MenuGui();  
 }  
 if(e.getSource()==panelEnd.buttonBack){  
 myFrameNext.dispose();  
 new MenuGui();  
 }  
 }  
}

Opis:

* Klasa ShowRatingsCars implementuje interfejs ActionListener i reprezentuje interfejs użytkownika do wyświetlania ocen samochodów.

Konstruktor:

Konstruktor ShowRatingsCars() inicjalizuje interfejs użytkownika do wyświetlania ocen samochodów.

* Inicjalizowane są zmienne i komponenty interfejsu, takie jak myFrameNext (obiekt klasy MyFrame), panelEnd (obiekt klasy MyLoverPanel), panelStart (obiekt klasy JPanel), panelMiddle (obiekt klasy JPanel) oraz centerLabel (obiekt klasy JLabel).
* Panel panelStart jest konfigurowany, ustawiając kolor tła na zielony.
* Panel panelMiddle jest konfigurowany, ustawiając kolor tła na zielony i układ na BoxLayout z pionowym rozmieszczeniem.
* Dodawane są elementy do paneli: centerLabel do panelStart, a także etykiety ocen samochodów do panelMiddle.
* Dodawane są panele do obiektu myFrameNext.

Metody:

Metoda funcRatingsCars() jest metodą statyczną, która zwraca listę ocen samochodów w postaci łańcuchów tekstowych.

* Tworzona jest lista ratingsCars.
* Dla każdego samochodu w liście GUI.Cars jest obliczana średnia ocena i dodawany jest odpowiedni łańcuch tekstowy do listy ratingsCars.
* Jeśli lista GUI.Cars jest pusta, dodawany jest odpowiedni komunikat do listy ratingsCars.
* Zwracana jest lista ratingsCars.

Metoda actionPerformed(ActionEvent e) obsługuje zdarzenia akcji, takie jak kliknięcie przycisku powrotu.

* Jeśli źródłem zdarzenia jest przycisk powrotu w menu górnym (myFrameNext.backItem), następuje zamknięcie obecnego okna i utworzenie nowego obiektu MenuGui() w celu powrotu do menu głównego.
* Jeśli źródłem zdarzenia jest przycisk powrotu w panelu dolnym (panelEnd.buttonBack), następuje zamknięcie obecnego okna i utworzenie nowego obiektu MenuGui() w celu powrotu do menu głównego.

Klasa ShowRatingsCars umożliwia wyświetlanie ocen samochodów w interfejsie użytkownika. Oceny samochodów są pobierane z listy GUI.Cars i wyświetlane w odpowiednich panel

3.10 Klasa AddRentCar

import javax.swing.\*;  
import java.awt.\*;  
import java.awt.event.ActionEvent;  
import java.awt.event.ActionListener;  
import java.text.DateFormat;  
import java.text.ParseException;  
import java.text.SimpleDateFormat;  
import java.util.Calendar;  
import java.util.Date;  
  
import static javax.swing.JOptionPane.showMessageDialog;  
  
public class AddRentCar implements ActionListener {  
 private final MyFrame myFrameNext = new MyFrame();  
 private final MyLoverPanel panelEnd;  
 private final JLabel labelStart, labelRentCheck, labelStartRent, labelRentStart, labelRentEnd;  
 private final MyTextField textFieldID, textFieldRentDays, textFieldStartRent;  
 private final MyButton buttonRentIdPass, buttonRentCalculate;  
 private final JCheckBox checkBoxAddCarFromToday;  
  
 AddRentCar() {  
 //Initialization of variables  
 myFrameNext.backItem.addActionListener(this);  
 JPanel panelStart = new JPanel();  
 JPanel panelMiddle = new JPanel();  
 panelEnd = new MyLoverPanel(" Dodaj rezerwacje ");  
 labelStart = new JLabel("Rezerwuj samochod: ");  
 JLabel LabelId = new JLabel("Wybierz ID samochodu: ");  
 textFieldID = new MyTextField();  
 buttonRentIdPass = new MyButton(" Sprawdz dostepnosc samochodu ");  
 labelRentCheck = new JLabel("");  
 textFieldRentDays = new MyTextField();  
 checkBoxAddCarFromToday = new JCheckBox("Czy wypozyczasz od dziasiejszego dnia?");  
 labelStartRent = new JLabel("\*Data poczatkowa: (format daty dd.MM.yyyy)");  
 textFieldStartRent = new MyTextField();  
 buttonRentCalculate = new MyButton(" Sprawdz dni w wypozyczeniu ");  
 labelRentStart = new JLabel("");  
 labelRentEnd = new JLabel("");  
 //Changing elements in panelStart  
 panelStart.setBackground(new Color(0, 200, 0));  
 //Changing elements in panel  
 panelMiddle.setBackground(new Color(0, 200, 0));  
 panelMiddle.setBorder(BorderFactory.createEmptyBorder(30, 30, 10, 30));  
 panelMiddle.setLayout(new GridLayout(0, 1, 10, 5));  
 textFieldID.setSize(new Dimension(250, 50));  
 textFieldID.setFont(new Font("Arctic", Font.PLAIN, 30));  
 buttonRentIdPass.addActionListener((ActionListener) this);  
 textFieldRentDays.setSize(new Dimension(250, 50));  
 textFieldRentDays.setFont(new Font("Arctic", Font.PLAIN, 30));  
 textFieldRentDays.setVisible(false);  
 checkBoxAddCarFromToday.setBackground(new Color(0, 200, 0));  
 checkBoxAddCarFromToday.setVisible(false);  
 checkBoxAddCarFromToday.setSize(new Dimension(250, 20));  
 labelStartRent.setVisible(false);  
 textFieldStartRent.setFont(new Font("Arctic", Font.PLAIN, 30));  
 textFieldStartRent.setVisible(false);  
 buttonRentCalculate.addActionListener((ActionListener) this);  
 buttonRentCalculate.setVisible(false);  
 //Changing elements in panelEnd  
 panelEnd.setBackground(new Color(0, 200, 0));  
 panelEnd.setBorder(BorderFactory.createEmptyBorder(30, 30, 10, 30));  
 panelEnd.setLayout(new GridLayout(0, 4, 10, 5));  
 panelEnd.buttonBack.addActionListener((ActionListener) this);  
 panelEnd.buttonSecond.addActionListener((ActionListener) this);  
 panelEnd.buttonSecond.setVisible(false);  
 //Add elements to panels  
 panelStart.add(labelStart, BorderLayout.CENTER);  
 panelMiddle.add(LabelId);  
 panelMiddle.add(textFieldID);  
 panelMiddle.add(buttonRentIdPass);  
 panelMiddle.add(labelRentCheck);  
 panelMiddle.add(textFieldRentDays);  
 panelMiddle.add(checkBoxAddCarFromToday);  
 panelMiddle.add(labelStartRent);  
 panelMiddle.add(textFieldStartRent);  
 panelMiddle.add(buttonRentCalculate);  
 panelMiddle.add(labelRentStart);  
 panelMiddle.add(labelRentEnd);  
 //Add elements to frames  
 myFrameNext.add(panelStart, BorderLayout.NORTH);  
 myFrameNext.add(panelMiddle, BorderLayout.CENTER);  
 myFrameNext.add(panelEnd, BorderLayout.SOUTH);  
 }  
 private String[] funcRentCar(Boolean fromToday,String textFieldStartRent, String day) {  
 Calendar startDate = Calendar.getInstance();  
 DateFormat dateFormat = new SimpleDateFormat("dd.MM.yyyy");  
 int days;  
 if(day.isEmpty()){  
 days=0;  
 }  
 else {  
 days = Integer.parseInt(day);  
 }  
 if (fromToday){}  
 else{  
 try {  
 Date date = dateFormat.parse(textFieldStartRent);  
 startDate.setTime(date);  
 }  
 catch (ParseException e) {  
 showMessageDialog(myFrameNext,"Nieprawidlowy format daty!");  
 return null;  
 }  
 }  
 Calendar endDate = (Calendar) startDate.clone();  
 endDate.add(Calendar.DATE, days);  
 return new String[] {dateFormat.format(startDate.getTime()), dateFormat.format(endDate.getTime())};  
 }  
  
 @Override  
 public void actionPerformed(ActionEvent e){  
 if(e.getSource() ==myFrameNext.backItem){  
 myFrameNext.dispose();  
 new MenuGui();  
 }  
 if(e.getSource()==buttonRentIdPass) {  
 try{  
 labelStart.setText("Rezerwuj samochod: ");  
 panelEnd.buttonSecond.setVisible(false);  
 textFieldRentDays.setVisible(false);  
 buttonRentCalculate.setVisible(false);  
 checkBoxAddCarFromToday.setVisible(false);  
 labelStartRent.setVisible(false);  
 textFieldStartRent.setVisible(false);  
 panelEnd.buttonSecond.setVisible(false);  
 labelRentStart.setText("");  
 labelRentEnd.setText("");  
 if (textFieldID.getText().isEmpty()) {  
 labelRentCheck.setText("Podaj liczbe identyfikatora");  
  
 }  
 else {  
 int ID = Integer.parseInt(textFieldID.getText());  
 GUI.Car car = GUI.funcSearchCar(ID);  
 if ( GUI.funcSearchCar(ID)!= null) {  
 labelStart.setText("Rezerwuj samochod: " + car.getMark()+" "+car.getModel()+ " " + car.getYearOfProduction());  
 labelRentCheck.setText("Na ile dni wypozyczasz: ");  
 textFieldRentDays.setVisible(true);  
 buttonRentCalculate.setVisible(true);  
 checkBoxAddCarFromToday.setVisible(true);  
 labelStartRent.setVisible(true);  
 textFieldStartRent.setVisible(true);  
 }  
 else {  
 labelRentCheck.setText("Nie ma takiego ID ");  
 }  
 }  
 }  
 catch (Exception e1){  
 showMessageDialog(myFrameNext, "To nie jest liczba!");  
 throw new RuntimeException(e1);  
 }  
  
 }  
  
 if(e.getSource()==buttonRentCalculate){  
 panelEnd.buttonSecond.setVisible(false);  
 if(textFieldRentDays.getText().isEmpty()){  
 showMessageDialog(myFrameNext,"Podaj ilosc dni! ");  
 }  
 else{  
 try{  
 String[] date = funcRentCar(!checkBoxAddCarFromToday.isSelected(),textFieldStartRent.getText(),textFieldRentDays.getText());  
 labelRentStart.setText("Rezerwacja zaczyna sie od: " + date[0]);  
 labelRentEnd.setText("Rezerwacja konczy sie: " + date[1]);  
 panelEnd.buttonSecond.setVisible(true);  
 }  
 catch (Exception e1){  
 showMessageDialog(myFrameNext, "To nie jest liczba!");  
 throw new RuntimeException(e1);  
 }  
 }  
 }  
 if(e.getSource()==panelEnd.buttonBack){  
 myFrameNext.dispose();  
 new MenuGui();  
 }  
 if (e.getSource() == panelEnd.buttonSecond) {  
 int ID = Integer.parseInt(textFieldID.getText());  
 GUI.Car car = GUI.funcSearchCar(ID);  
 car.AddRentCar(myFrameNext,labelRentStart.getText(),Integer.parseInt(textFieldRentDays.getText()));  
 showMessageDialog(myFrameNext,"Dodano rezerwacje! ");  
 myFrameNext.dispose();  
 new MenuGui();  
 }  
 }  
}

Opis:

* Klasa AddRentCar implementuje interfejs ActionListener i reprezentuje interfejs użytkownika do dodawania rezerwacji samochodów.

Konstruktor:

Konstruktor AddRentCar() inicjalizuje interfejs użytkownika do dodawania rezerwacji samochodów.

* Inicjalizowane są zmienne i komponenty interfejsu, takie jak myFrameNext (obiekt klasy MyFrame), panelEnd (obiekt klasy MyLoverPanel), labelStart, labelRentCheck, labelStartRent, labelRentStart, labelRentEnd (obiekty klasy JLabel), textFieldID, textFieldRentDays, textFieldStartRent (obiekty klasy MyTextField), buttonRentIdPass, buttonRentCalculate (obiekty klasy MyButton), checkBoxAddCarFromToday (obiekt klasy JCheckBox).
* Konfigurowane są elementy interfejsu, takie jak kolory tła, czcionki i widoczność.
* Dodawane są elementy do paneli: etykiety, pola tekstowe i przyciski.
* Dodawane są panele do obiektu myFrameNext.

Metody:

Metoda funcRentCar(Boolean fromToday, String textFieldStartRent, String day) jest prywatną metodą, która oblicza datę początkową i końcową rezerwacji.

* Tworzony jest obiekt startDate klasy Calendar i obiekt dateFormat klasy SimpleDateFormat.
* Jeśli parametr fromToday jest prawdziwy, to startDate jest ustawiany na bieżącą datę.
* W przeciwnym razie, następuje próba parsowania daty początkowej z tekstu textFieldStartRent i ustawienie startDate na tę wartość.
* Tworzony jest obiekt endDate, który jest klonem startDate, a następnie do endDate dodawana jest liczba dni podana w parametrze day.
* Zwracane są daty początkowa i końcowa w postaci tablicy łańcuchów tekstowych.

Metoda actionPerformed(ActionEvent e) obsługuje zdarzenia akcji, takie jak kliknięcie przycisku powrotu.

* Jeśli źródłem zdarzenia jest przycisk powrotu w menu górnym (myFrameNext.backItem), następuje zamknięcie obecnego okna i utworzenie nowego obiektu MenuGui() w celu powrotu do menu głównego.
* Jeśli źródłem zdarzenia jest przycisk buttonRentIdPass, następuje sprawdzenie dostępności samochodu na podstawie podanego identyfikatora.
* Jeśli źródłem zdarzenia jest przycisk buttonRentCalculate, następuje obliczenie daty początkowej i końcowej rezerwacji na podstawie podanej liczby dni.
* Jeśli źródłem zdarzenia jest przycisk powrotu w panelu dolnym (panelEnd.buttonBack), następuje zamknięcie obecnego okna i utworzenie nowego obiektu MenuGui() w celu powrotu do menu głównego.
* Jeśli źródłem zdarzenia jest przycisk buttonSecond w panelu dolnym, następuje dodanie rezerwacji samochodu na podstawie podanych danych.

Klasa AddRentCar umożliwia dodawanie rezerwacji samochodów w interfejsie użytkownika. Użytkownik może wybrać samochód na podstawie ID, podać liczbę dni wypożyczenia i określić datę początkową rezerwacji.

3.11 Klasa MyFrame

import javax.swing.\*;  
import java.awt.event.ActionEvent;  
import java.awt.event.ActionListener;  
import java.awt.event.KeyEvent;  
import java.io.BufferedWriter;  
import java.io.FileWriter;  
import java.io.IOException;  
import java.io.BufferedReader;  
import java.io.FileReader;  
  
public class MyFrame extends JFrame implements ActionListener {  
 private static int currentCarNumber = 1;  
 final JMenu fileMenu;  
 JMenu editMenu;  
 JMenu aboutMenu;  
 JMenuBar menuBar;  
 JMenuItem loadItem;  
 JMenuItem saveItem;  
 JMenuItem exitItem;  
 JMenuItem backItem;  
 JMenuItem aboutItem;  
 MyFrame(){  
 this.setTitle("Wypozyczalnia Samochodow"); //sets title of frame  
 this.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);  
 this.pack();  
 this.setVisible(true); //make frame visible  
 this.setSize(600,600);  
 ImageIcon image = new ImageIcon("lib/Picture/TitlePicture.png"); //Create an ImageIcon  
 this.setIconImage(image.getImage()); //Change icon of frame  
 menuBar = new JMenuBar();  
 fileMenu = new JMenu("File");  
 editMenu = new JMenu("Edit");  
 aboutMenu = new JMenu("About");  
 loadItem = new JMenuItem("Load");  
 saveItem = new JMenuItem("Save");  
 exitItem = new JMenuItem("Exit");  
 loadItem.addActionListener(this);  
 saveItem.addActionListener(this);  
 exitItem.addActionListener(this);  
 saveItem.setMnemonic(KeyEvent.VK\_S & KeyEvent.VK\_CONTROL);  
 fileMenu.add(loadItem);  
 fileMenu.add(saveItem);  
 fileMenu.add(exitItem);  
 backItem = new JMenuItem("Back");  
 backItem.addActionListener(this);  
 editMenu.add(backItem);  
 aboutItem = new JMenuItem("About");  
 aboutItem.addActionListener(this);  
 aboutMenu.add(aboutItem);  
 menuBar.add(fileMenu);  
 menuBar.add(editMenu);  
 menuBar.add(aboutMenu);  
 this.setJMenuBar(menuBar);  
 }  
 private void about(){  
 JOptionPane.showMessageDialog(this, "Program o nazwie Wypozyczalnia Samochodow zostal napisany na cele projektu na studia. \nSklad zespolu: \n Lider: <Szymon Kasprzyk>, \nPomocnik: <Michal Gruszczynski>, \nPowolna pomoc: Oskar Pytlewki, \nProzniak: klaudia iwanowicz");  
 }  
 private static void saveCarToFile() {  
 try {  
 String ratingStr="", rentStr="";  
 BufferedWriter writer = new BufferedWriter(new FileWriter("samochody.txt", true));  
 for (GUI.Car car: GUI.Cars) {  
// if(car.ratings.isEmpty())  
// ratingStr = "0";  
// else  
// for (Integer rating : car.ratings) {  
// ratingStr += ",," + rating.toString();  
// }  
// if(car.rentCar.isEmpty())  
// rentStr = "0";  
// else  
// for (String rent : car.rentCar){  
// rentStr += ",," + rent.substring(28);  
// }  
 writer.write(car.getMark() + "," + car.getModel() + "," + car.getPrice() + "," + car.getYearOfProduction() + "," +ratingStr + ","+rentStr);  
 writer.newLine();  
 }  
 writer.close();  
 currentCarNumber++;  
 } catch (IOException ex) {  
 ex.printStackTrace();  
 }  
 }  
 private static void loadDataFromFile() {  
 try {  
 String[] parametrs, ratings, rents;  
 BufferedReader reader = new BufferedReader(new FileReader("samochody.txt"));  
 String line;  
 while ((line = reader.readLine()) != null) {  
 parametrs = line.split(",");  
 GUI.Car Car = new GUI.Car(parametrs[0], parametrs[1], Double.parseDouble(parametrs[2]), Short.parseShort(parametrs[3]));  
 GUI.Cars.add(Car);  
// if(parametrs.length > 4) {  
// ratings = parametrs[4].split(",,");  
// for (int i = 0; i < ratings.length - 1; i++)  
// Car.addRating(Integer.parseInt(ratings[i]));  
// if(parametrs.length > 5){  
// rents = parametrs[5].split((".."));  
// for (int i = 0; i < rents.length - 1; i++){  
// Car.AddRentCarFromLoad(rents[i].substring(0,10), Integer.parseInt(rents[i].substring(11,14)));  
// }  
// }  
 }  
// }  
 reader.close();  
 } catch (IOException ex) {  
 ex.printStackTrace();  
 }  
 }  
  
  
 @Override  
 public void actionPerformed(ActionEvent e) {  
 if(e.getSource()==loadItem) {  
 loadDataFromFile();  
 }  
 if(e.getSource()==saveItem){  
 saveCarToFile();  
 }  
 if(e.getSource()==aboutItem)  
 about();  
 if(e.getSource()==exitItem)  
 System.exit(0);  
 }  
}

Opis:

* Klasa MyFrame dziedziczy po klasie JFrame i implementuje interfejs ActionListener. Reprezentuje główne okno interfejsu użytkownika programu "Wypożyczalnia Samochodów".

Konstruktor:

Konstruktor MyFrame() inicjalizuje główne okno interfejsu użytkownika.

* Ustawiane są tytuł i rozmiar okna.
* Tworzony jest obiekt ImageIcon z obrazem tytułowym.
* Ikona okna jest ustawiana na ten obraz.
* Inicjalizowane są komponenty menu, takie jak menuBar, fileMenu, editMenu, aboutMenu, loadItem, exitItem, backItem i aboutItem.
* Ustawiane są skróty klawiaturowe dla niektórych przycisków (setMnemonic(KeyEvent.VK\_S & KeyEvent.VK\_CONTROL)).
* Dodawane są przyciski menu do paska menu.
* Pasek menu jest ustawiany w oknie.

Metody:

Metoda about() jest prywatną metodą, która wyświetla okno dialogowe z informacjami o programie.

* Wyświetla okno dialogowe JOptionPane.showMessageDialog() z odpowiednim komunikatem.

Metoda actionPerformed(ActionEvent e) obsługuje zdarzenia akcji, takie jak kliknięcie przycisków w menu.

* Jeśli źródłem zdarzenia jest przycisk aboutItem, wywoływana jest metoda about() w celu wyświetlenia informacji o programie.
* Jeśli źródłem zdarzenia jest przycisk exitItem, program jest zamykany.

1. **DODAWANIE SAMOCHODÓW DO BAZY**

4.1 Marka

Włączając program „Wypożyczalnia samochodów” na starcie ukaże nam się menu, w którym znajduję się kilka opcji do wyboru. Pierwszą opcją jest opcja „Dodaj samochód” – klikając przekieruje nas do strony, gdzie będzie można dodać samochody.

Obraz zawierający tekst, zrzut ekranu, oprogramowanie, wyświetlacz

Opis wygenerowany automatycznie

By dodać samochód musimy podać konkretne parametry samochodu takie jak: marka, model, rok produkcji oraz cenę za dzień wynajmu. Pierwszym z parametrów jest marka samochodu – wpisujemy ją w wyznaczonym polu tekstowym.

Obraz zawierający tekst, zrzut ekranu, oprogramowanie, Oprogramowanie multimedialne

Opis wygenerowany automatycznie

Dla przykładu została wpisana marka ‘OPEL’ jak można zauważyć poniżej.

Obraz zawierający tekst, zrzut ekranu, oprogramowanie, System operacyjny

Opis wygenerowany automatycznie

* 1. Model

Kolejnym parametrem do wpisania jest model samochodu. Dla przykładu został wpisany model ‘ASTRA’ pokazany na poniższym zdjęciu.

Obraz zawierający tekst, zrzut ekranu, oprogramowanie, System operacyjny

Opis wygenerowany automatycznie

* 1. Rok produkcji

Następnym krokiem jest podanie roku produkcji samochodu.

Obraz zawierający tekst, zrzut ekranu, oprogramowanie, wyświetlacz

Opis wygenerowany automatycznie

* 1. Cena

Ostatnim krokiem do wpisania jest cena za dzień wynajmu samochodu.

Obraz zawierający tekst, zrzut ekranu, oprogramowanie, wyświetlacz

Opis wygenerowany automatycznie

* 1. Weryfikacja

Na koniec przechodzimy przez weryfikacje klikając przycisk „Nie jestem robotem”. Gdy to zrobimy możemy ostatecznie zatwierdzić dodawanie naszego samochodu.

Obraz zawierający tekst, zrzut ekranu, oprogramowanie, wyświetlacz

Opis wygenerowany automatycznie

Po dodaniu naszego auta możemy do zobaczyć go w zakładce „Wyświetl samochody” w menu głównym. Samochód został dodany z naszymi parametrami z wcześniejszych kroków oraz dostał swój unikatowy kod ID.

Obraz zawierający zrzut ekranu, tekst, komputer, oprogramowanie

Opis wygenerowany automatycznie

1. **REZERWACJA DOSTĘPNYCH SAMOCHODÓW**

5.1 Dodanie samochodów

Kolejną opcją naszego programu jest opcja „Rezerwuj samochód”, dzięki której możemy wypożyczyć samochód na dogodny termin dla nas..

Obraz zawierający tekst, zrzut ekranu, oprogramowanie, wyświetlacz

Opis wygenerowany automatycznie

Pierwszym krokiem jest wprowadzenie ID auta, które zamierzamy zarezerwować. W naszym przypadku jest to nasze auto, które wprowadziliśmy w punktach powyżej. Nasz OPEL ASTRA dostał numer ID = 1, co oznacza, że taki właśnie numer wpisujemy w poniższe pole. Następnie klikamy „Sprawdź dostępność samochodu”.

Obraz zawierający zrzut ekranu, tekst, oprogramowanie, wyświetlacz

Opis wygenerowany automatycznie

* 1. Okres rezerwacji

Jeśli wprowadziliśmy wszystkie dane poprawnie to możemy w następnym kroku podać ilość dni, na jaką chcemy wypożyczyć nasze auto. W naszym przypadku chcemy zarezerwować samochód na 4 dni oraz zaznaczamy opcję „Czy wypożyczasz od dzisiejszego dnia?” by rezerwacja była od dziś.

Obraz zawierający tekst, zrzut ekranu, oprogramowanie, Wielobarwność

Opis wygenerowany automatycznie

* 1. Wprowadzenie daty

Ostatnim krokiem do wypożyczenia jest podanie daty, od kiedy samochód ma być zarezerwowany. W naszym przykładzie podajemy datę oraz klikamy w przycisk „Sprawdź dni w wypożyczeniu”, dzięki czemu program pokaże nam od kiedy nasza rezerwacja zaczyna się i kiedy się kończy.

Obraz zawierający tekst, zrzut ekranu, oprogramowanie, System operacyjny

Opis wygenerowany automatycznie

* 1. Potwierdzenie rezerwacji

Gdy już wszystko zostało wprowadzone klikamy „Zatwierdź rezerwacje” co pozwoli nam sfinalizować wypożyczenie.

Obraz zawierający tekst, zrzut ekranu, oprogramowanie, System operacyjny

Opis wygenerowany automatycznie

Po kliknięciu dodania rezerwacji pokaże nam się informacja, że rezerwacja została dodana.

Obraz zawierający tekst, zrzut ekranu, Czcionka, numer

Opis wygenerowany automatycznie

1. **KOSZTY WYNAJMU**

6.1 Wybór samochodu

Kolejną pozycją w naszym menu jest „Oblicz koszt wynajmu samochodu”. Klient ma możliwość przed wypożyczeniem samochodu sprawdzić, jakie koszty poniesie przy wypożyczaniu auta na dany okres czasu.

Obraz zawierający tekst, zrzut ekranu, oprogramowanie, wyświetlacz

Opis wygenerowany automatycznie

Pierwszą rzeczą jaką musimy zrobić to wybranie samochodu, na którym chcemy obliczyć koszty wynajmu. Wpisujemy nasze wcześniej dodane auto OPEL ASTRA, które posiada ID=1 na potrzeby testów. Gdy już ID zostało wprowadzone – klikamy „Sprawdź”.

Obraz zawierający tekst, zrzut ekranu, oprogramowanie, System operacyjny

Opis wygenerowany automatycznie

* 1. Okres wypożyczenia

Następnie pojawi nam się zapytanie na ile dni chcemy wypożyczyć auto. Gdy już podamy ilość dni – klikamy „Oblicz”.

Obraz zawierający zrzut ekranu, tekst, oprogramowanie, Oprogramowanie multimedialne

Opis wygenerowany automatycznie

* 1. Obliczanie kosztów wynajmu

Program finalnie oblicza nam kwotę, którą musiał by zapłacić klient rezerwując OPLA na 4 dni. Następnie jeśli chcemy wyjść do menu głównego klikamy „Powrót”.

Obraz zawierający zrzut ekranu, tekst, oprogramowanie, Oprogramowanie multimedialne

Opis wygenerowany automatycznie

1. **SYSTEM OCENY SAMOCHODÓW**

7.1 Wybór samochodu

Wybierając w menu głównym „Oceń samochód” możemy po zakończonej rezerwacji podzielić się swoją opinią o samochodzie w skali od 1 do 5 (1-najniżej, 5-najlepiej).

Obraz zawierający tekst, zrzut ekranu, oprogramowanie, wyświetlacz

Opis wygenerowany automatycznie

Następnie tak jak we wcześniejszych etapach wpisujemy numer ID samochodu, które chcemy ocenić i klikamy „Sprawdź” by przejść dalej.

Obraz zawierający zrzut ekranu, tekst, oprogramowanie, System operacyjny

Opis wygenerowany automatycznie

* 1. Ocena pojazdu

Następnym krokiem jest ocena samochodu poprzez rozwinięcie paska po prawej stronie i wybór odpowiedniej oceny. Klikając „dodaj ocenę” pojawi się komunikat o jej dodaniu.

Obraz zawierający tekst, zrzut ekranu, oprogramowanie, Oprogramowanie multimedialne

Opis wygenerowany automatycznie

Obraz zawierający tekst, zrzut ekranu, Czcionka, numer

Opis wygenerowany automatycznie

1. **WYŚWIETLANIE OCEN SAMOCHODÓW**

8.1 Spis wszystkich ocen

Ostatnią opcją w naszym menu jest wyświetlenie wszystkich ocen klientów poprzez kliknięcie „Wyświetl oceny samochodów”.

Obraz zawierający tekst, zrzut ekranu, oprogramowanie, wyświetlacz

Opis wygenerowany automatycznie

Tutaj znajdziemy listę wszystkich ocenionych przez klientów samochodów.

Obraz zawierający zrzut ekranu, tekst, komputer, oprogramowanie

Opis wygenerowany automatycznie

1. **PODSUMOWANIE I PERSPEKTYWY ROZWOJU**

9.1 Podsumowanie funkcjonalności

Podsumowanie funkcjonalności przedstawia kluczowe możliwości systemu wypożyczalni samochodów. Poniżej znajdują się główne funkcje i cechy programu:

* System dodawania samochodów: Program umożliwia dodawanie nowych samochodów do bazy danych wypożyczalni. Informacje o samochodach obejmują markę, model, rok produkcji, cenę oraz inne szczegóły.
* Wyświetlanie dostępnych samochodów: Użytkownicy mogą przeglądać dostępne samochody w wypożyczalni, uzyskując informacje o marce, modelu, roczniku. To umożliwia łatwe znalezienie odpowiedniego pojazdu.
* Rezerwacja samochodów: Użytkownicy mają możliwość dokonywania rezerwacji samochodów na określone daty. System zapewnia sprawne zarządzanie rezerwacjami i dostępnością samochodów.
* Obliczanie kosztów wynajmu auta: Program automatycznie oblicza koszty wynajmu auta na podstawie wybranego samochodu, okresu wypożyczenia oraz innych czynników. Użytkownik otrzymuje jasne informacje dotyczące opłat za wypożyczenie.
* System oceny auta: Użytkownicy mogą oceniać samochody po zakończonym wypożyczeniu. System umożliwia wystawienie oceny oraz dodanie opinii na temat doświadczeń związanych z danym samochodem.
* Wyświetlanie wszystkich ocen: Program umożliwia przeglądanie wszystkich ocen wystawionych przez użytkowników dla poszczególnych samochodów. To pozwala innym użytkownikom na zapoznanie się z opiniami i ocenami innych użytkowników przed dokonaniem wyboru samochodu.
  1. Propozycje dalszych ulepszeń

Pomimo zrealizowania głównych funkcjonalności, istnieje wiele możliwości rozwoju i ulepszenia programu wypożyczalni samochodów. Oto kilka z tych możliwości:

* Baza klientów: Rozbudowa systemu o zaawansowaną bazę danych klientów, umożliwiającą przechowywanie dodatkowych informacji o klientach, takich jak preferencje, historia wypożyczeń, dane kontaktowe itp. To pozwoli na lepsze zarządzanie relacjami z klientami oraz dostosowanie oferty do ich potrzeb.
* Możliwość dodawania komentarzy do ocen: Rozszerzenie systemu oceniania samochodów o możliwość dodawania komentarzy i opinii użytkowników. To umożliwi bardziej szczegółowe recenzje i pozwoli innym użytkownikom na uzyskanie dodatkowych informacji na temat doświadczeń związanych z danym samochodem.
* Dodanie dodatkowo płatnego wyposażenia samochodu: Wprowadzenie możliwości dodatkowego płatnego wyposażenia samochodu, takiego jak nawigacja GPS, fotelik dla dziecka, bagażnik na rowery itp. Umożliwi to klientom personalizację wypożyczonych pojazdów i zwiększy elastyczność oferty wypożyczalni.
* Filtrowanie i sortowanie aut: Rozbudowa funkcji filtrowania i sortowania podczas przeglądania dostępnych samochodów. Użytkownicy będą mogli filtrować samochody według preferencji, takich jak marka, model, cena, rok produkcji, rodzaj paliwa itp. Dodatkowo, możliwość sortowania wyników pozwoli na szybkie znalezienie najlepszego pasującego pojazdu.

Te możliwości rozwoju i ulepszenia programu wypożyczalni samochodów mają na celu rozszerzenie funkcjonalności, poprawę personalizacji, usprawnienie procesów i zwiększenie satysfakcji użytkowników. Implementacja tych ulepszeń przyczyni się do lepszej obsługi klientów i bardziej efektywnego zarządzania wypożyczalnią.