|  |  |
| --- | --- |
|  | **AKADEMIA GÓRNICZO-HUTNICZA** |

Dokumentacja do projektu

**Car Rental**

z przedmiotu

**Programowanie obiektowe**

Elektronika rok II

*Robert Zubek*

poniedziałek 17:10

prowadzący: Rafał Frączek

08.01.2024

# Opis projektu

Projekt służy do pomocy w zarządzaniu wypożyczalnią samochodową. Główną funkcjonalnością jest system wynajmu- możliwe jest zalogowanie jako wypożyczający i dodanie prośby. Następnie admin ze swojego kontekstu może wypożyczyć auto lub odmówić. Oprócz tego występują funkcje poboczne omówione poniżej.

# Project description

Project was made to help in managing a car rental. The main functionality of the program is a renting system- it is possible to login as a borrower and send a request. Next admin, from his context, can approve or decline it. Despite that, there are side functions discussed below.

# Instrukcja użytkownika

W tym punkcie należy umieścić instrukcję użytkowania programu. Może być to na przykład opis poszczególnych menu w programie. W przypadku gry należy opisać zasady gry. Opcjonalnie można wstawić zrzuty ekranu. Jeśli uruchomienie programu wymaga wykonania jakiś niestandardowych lub dodatkowych czynności (na przykład uruchomienie serwera baz danych itp.) to należy zamieścić tę informację.

# Kompilacja

Wystarcza standardowa kompilacja. Projekt działa na dowolnym systemie operacyjnym.

# Pliki źródłowe

Projekt składa się z następujących plików źródłowych:

* *Admin.h –* zawiera klasę *Admin*,
* *User.h* – zawiera klasę *User*,
* *Car.h –* zawiera klasę *Car*,
* *Rent.h –* zawiera klasę *Rent,*
* *adminSaver.h –* zawiera klasę *adminSaver,*
* *userSaver.h –* zawiera klasę *userSaver*,
* *carSaver.h –* zawiera klasę *carSaver*,
* *rentSaver.h –* zawiera klasę *rentSaver.*

# Zależności

brak

# Opis klas

W projekcie utworzono następujące klasy:

* User – reprezentuje użytkownika:
  + bool change\_password(string password) – zmienia obecne hasło na inne,
  + void remember\_password(int pesel, string document\_number) – przypomina aktualne hasło,
  + bool login(string password) – loguje użytkownika,
  + bool logOut(void) – wylogowywuje użytkownika.
* Admin – reprezentuje admina, dziedziczy z klasy User, stworzona jedynie dla czytelności kodu.
* Car – reprezentuje samochód do wypożyczenia:
  + void rent(int number) – zmienia stan samochodu na wypożyczony przez danego użytkownika,
  + void free(void) – zmienia stan samochodu na dostępny.
* Rent – reprezentuje wypożyczenie.
* userSaver – służy do zapisu i odczytu danych na temat użytkowników:
  + void save(vector <User&> users) – zapisuje dane z wektora użytkowników do pliku txt,
  + vector<User> read(vector<User>& users)– odczytuje dane na temat użytkowników z pliku txt i zapisuje do wektora.
* adminSaver – służy do zapisu i odczytu danych na temat adminów:
  + void save(vector <Admin&> admins) – zapisuje dane z wektora adminów do pliku txt,
  + vector<Admin> read(vector<Admin>& admins)– odczytuje dane na temat adminów z pliku txt i zapisuje do wektora.
* carSaver – służy do zapisu i odczytu danych na temat samochodów:
  + void save(vector <Car&> cars) – zapisuje dane z wektora samochodów do pliku txt,
  + vector<Car> read(vector<Car>& cars)– odczytuje dane na temat samochodów z pliku txt i zapisuje do wektora.
* rentSaver – służy do zapisu i odczytu danych na temat wypożyczeń:
  + void save(vector <Rent&> rents) – zapisuje dane z wektora wypożyczeń do pliku txt,
  + vector<Rent> read(vector<Rent>& rents)– odczytuje dane na temat wypożyczeń z pliku txt i zapisuje do wektora.

# Zasoby

W projekcie wykorzystywane są następujące pliki zasobów:

* users.txt – plik zawierający dane użytkowników. Struktura pliku:
  + jedna linia = jeden użytkownik w kolejności:
  + nazwisko,
  + wiek,
  + pesel,
  + numer dokumentu,
  + hasło,
  + numer użytkownika,
  + stan użytkownika (aktywny, dezaktywowany).
* admins.txt – plik zawierający dane adminów. Struktura pliku:
  + jedna linia = jeden admin w kolejności:
  + nazwisko,
  + wiek,
  + pesel,
  + numer dokumentu,
  + hasło,
  + numer użytkownika,
  + stan admina (aktywny, dezaktywowany).
* cars.txt – plik zawierający dane samochodów. Struktura pliku:
  + jedna linia = jeden samochód w kolejności:
  + status,
  + model,
  + opis,
  + cena.
* rents.txt – plik zawierający dane wypożyczeń. Struktura pliku:
  + jedna linia = jedno wypożyczenie w kolejności:
  + status,
  + zakres dat,
  + numer wypożyczającego użytkownika,
  + model samochodu,
  + numer wypożyczenia.

# Dalszy rozwój i ulepszenia

W przyszłości dobrym pomysłem byłoby zautomatyzowanie zakresu dat wypożyczeni i obliczanie kwoty wypożyczenia. Obecnie o te rzeczy musi ręcznie dbać admin przy akceptacji. W przypadku błędu admina, możliwe jest wypożyczenie tego samego modelu samochodu dwóm użytkownikom jednocześnie. Również wszystkie dane w programie muszą być wprowadzane w postaci jednego wyrazu z powodu sposobu zapisu, co może być niedogodne.

# Inne

brak