Rejestr Zarejestrowanych Pojazdów

Kacper Baryłowicz

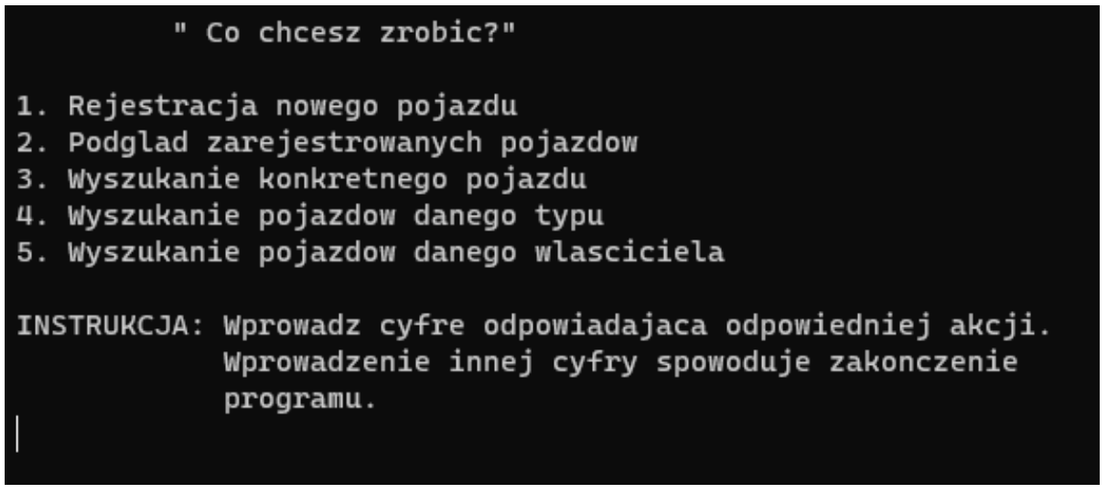
SPIS TREŚCI

Opis Programu

Program przechowuje zarejestrowane pojazdy w pliku baza.txt. Użytkownik może rejestrować nowe pojazdy, przeglądać już wcześniej zarejestrowane pojazdy, zmieniać właścicieli pojazdów, wyszukiwać kokretne pojazdy i usuwać pojazdy. Możliwe jest również robienie wydruków wybranych pojazdów do pliku wydruk.txt. Program korzysta z wcześniej zapisanych pojazdów, których dane mogą nadpisać automatycznie część cech rejestrowanego pojazdu.

Działanie programu

Rejestracja nowego pojazdu;



Po uruchomieniu programu mamy do wyboru 5 opcji;

1. Rejestracja nowego pojazdu

2. Podgląd zarejestrowanych pojazdów

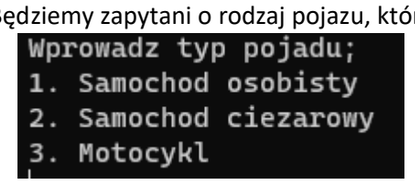
3. Wyszukanie kokretnego pojazdu.

4. Wyszukanie pojazdów danego typu.

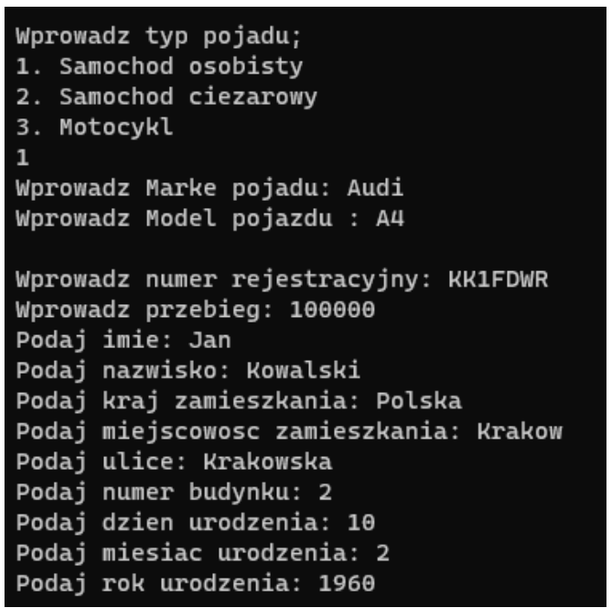
5. Wyszukanie pojazdów danego właścieciela

Gdy wybierzemy inną opcje niż jedną z powyższych np. 6 to program się zakończy i wyświetli komunikat.

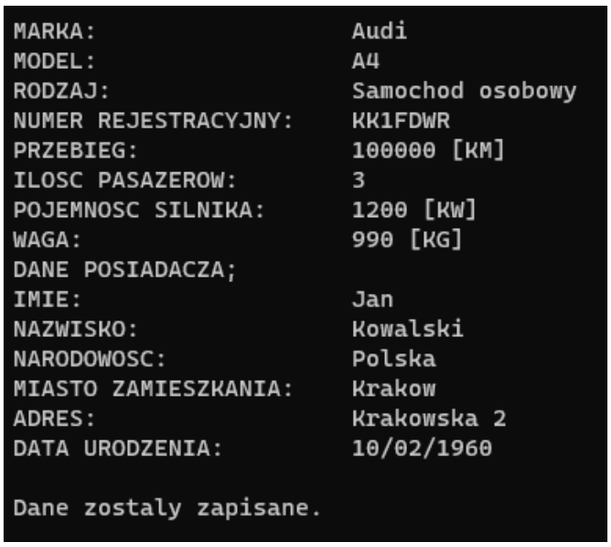
Po wybraniu opcji nr 1. Będziemy zapytani o rodzaj pojazu, który chcemy zarejestrować.



Po wybraniu opcji np. 1 ukaże się szereg danych do wprowadzenia. Takich jak: marka, model, numer rejestracyjny, przebieg, imię, nazwisko, kraj zamieszkania, miejscowość zamieszkania, ulica, numer, budynku, dzień urodzenia, miesiąc urodzeniam rok urodzenia (dane mogą różnić w zależności od wcześniejszego wyboru typu pojazdu).



Po wprowadzeniu danych, zostaną one wyświetlone na ekranie, oraz dane zostaną zapisane w pliku tekstowym baza.txt.



Podgląd zarejestrownych pojazdów;

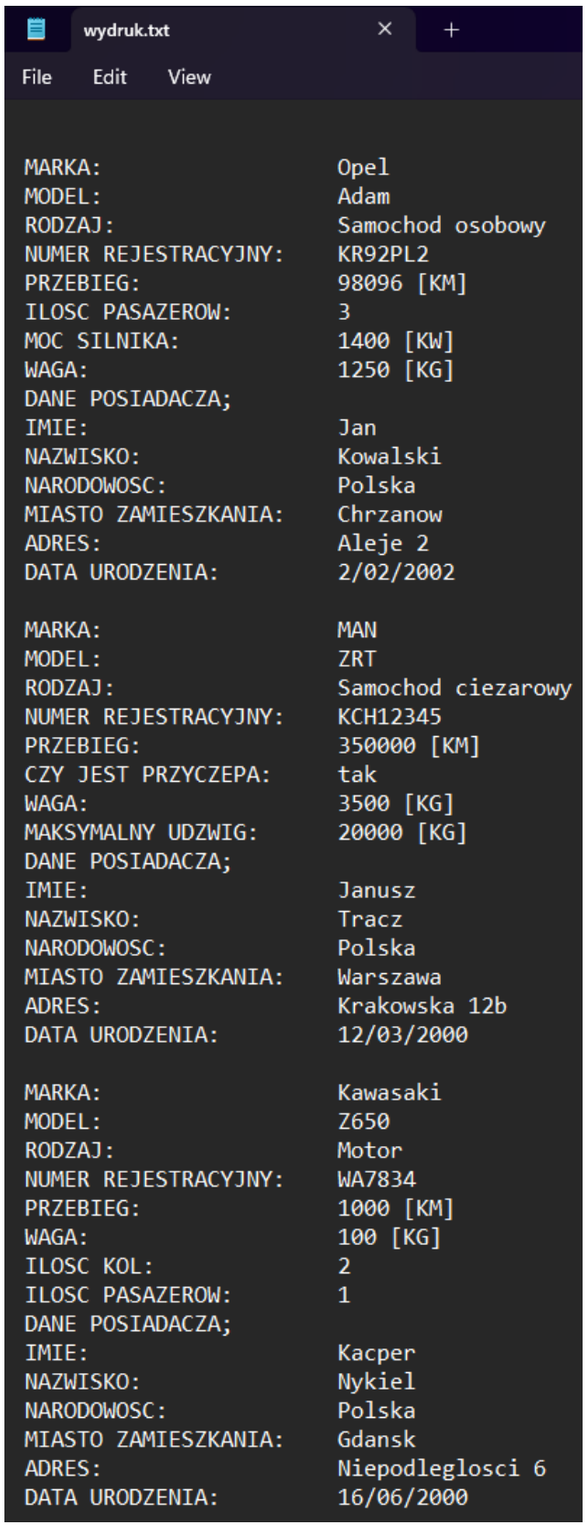
Gdy dane się zapiszemy program wróci do menu głównego, gdzie będziemy mogli ponownie zarejestrować pojazd, albo wykonać inne czynności jak np po wybraniu opcji nr. 2, gdzie ukażą nam się wszystkie zarejestrowane wcześniej pojazdy.

```
MARKA:                  Opel
MODEL:                  Adam
RODZAJ:                 Samochod osobowy
NUMER REJESTRACYJNY:    KR92PL2
PRZEBIEG:               98096 [KM]
ILOSC PASAZEROW:        3
POJEMNOSC SILNIKA:      1400 [KW]
WAGA:                   1250 [KG]
DANE POSIADACZA;
IMIE:                   Jan
NAZWISKO:               Kowalski
NARODOWOSC:             Polska
MIASTO ZAMIESZKANIA:    Chrzanow
ADRES:                  Aleje 2
DATA URODZENIA:         2/02/2002

MARKA:                  MAN
MODEL:                  ZRT
RODZAJ:                 Samochod ciezarowy
NUMER REJESTRACYJNY:    KCH12345
PRZEBIEG:               350000 [KM]
CZY JEST PRZYCZEPA:     tak
WAGA:                   3500 [KG]
MAKSYMALNY UDZWIG:      20000 [KG]
DANE POSIADACZA;
IMIE:                   Janusz
NAZWISKO:               Tracz
NARODOWOSC:             Polska
MIASTO ZAMIESZKANIA:    Warszawa
ADRES:                  Krakowska 12b
DATA URODZENIA:         12/03/2000

MARKA:                  Kawasaki
MODEL:                  Z650
RODZAJ:                 Motor
NUMER REJESTRACYJNY:    WA7834
PRZEBIEG:               1000 [KM]
WAGA:                   100 [KG]
ILOSC KOL:              2
ILOSC PASAZEROW:        1
DANE POSIADACZA;
IMIE:                   Kacper
NAZWISKO:               Nykiel
NARODOWOSC:             Polska
MIASTO ZAMIESZKANIA:    Gdansk
ADRES:                  Niepodleglosci 6
DATA URODZENIA:         16/06/2000
```

Następnie program zapyta czy chcemy wydrukować dane i po wpisaniu “TAK” program zapisze dane w pliku wydruk.txt.



Gdy wpiszemy “NIE” dane się nie zapiszą do pliku. Niezależnie od wybranej opcji program powróci do menu głównego.

Wyszukanie kokretnego pojazdu;

Po wybraniu opcji nr.3 w menu głównym, będziemy mogli wyszukać kokretny pojazd wpisując jego numer rejestracyjny.

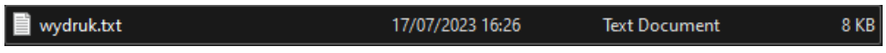
```
WPROWADZ NUMER REJESTRACYJNY :KK23W4
```

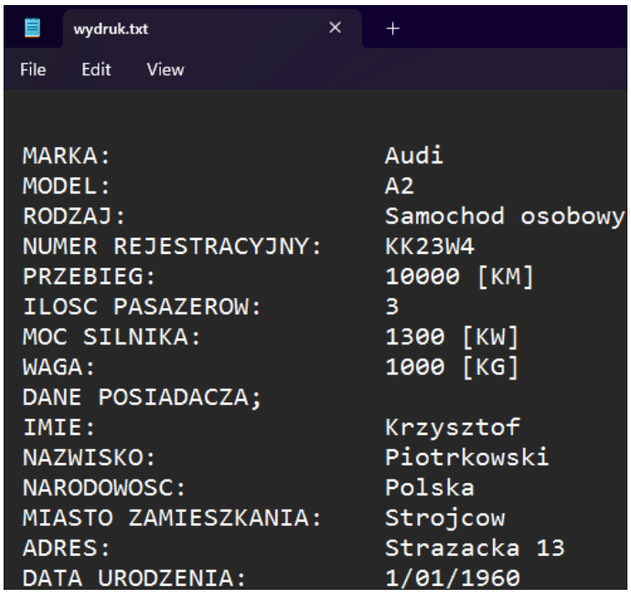
Po wprowadzeniu poprawnego numeru rejestracyjnego pokażą się dane pojazdu, oraz akcje, które możemy wykonać. W przeciwnym razie, jeśli numer rejestracyjny będzie niepoprawny, to pokaże się komunikat o niepoprawnych danych I program wróci do menu głównego.

```
MARKA:                  Audi
MODEL:                  A2
RODZAJ:                 Samochod osobowy
NUMER REJESTRACYJNY:    KK23W4
PRZEBIEG:               10000 [KM]
ILOSC PASAZEROW:        3
MOC SILNIKA:            1300 [KW]
WAGA:                   1000 [KG]
DANE POSIADACZA;
IMIE:                   Krzysztof
NAZWISKO:               Piotrkowski
NARODOWOSC:             Polska
MIASTO ZAMIESZKANIA:    Strojcow
ADRES:                  Strazacka 13
DATA URODZENIA:         1/01/1960

Co chcesz zrobi? z tymi danymi?
1. Wydruk
2. Zmiana wlasciciela
3. usun
4. nic
```

Gdy wybierzemy opcje nr.1 dane tego pojazdu zostaną zapisane do pliku wydruk.txt.





Jednak, gdy wybierzemy opcje nr.2. Będziemy poproszeni o wprowadzenie: imienia, nazwiska, narodowości, miasta zamieszkania, ulicy, numeru budynku, dzień urodzenia, miesiąc urodzenia, rok urodzenia.

```
Wprowadz IMIE: Jan
Wprowadz NAZWISKO: Kowalski
Wprowadz KRAJ ZAMIESZKANIA: Polska
Wprowadz MIEJSCOWOSC: Krakow
Wprowadz ULICE: Krakowska
Wprowadz NUMER BUDYNKU: 2
Wprowadz DZIEN URODZENIA: 2
Wprowadz MIESIAC URODZENIA: 2
Wprowadz ROK URODZENIA: 1999
```

I dane zostaną podmienione.

```
MARKA:                  Audi
MODEL:                  A2
RODZAJ:                 Samochod osobowy
NUMER REJESTRACYJNY:    KK23W4
PRZEBIEG:               10000 [KM]
ILOSC PASAZEROW:        3
MOC SILNIKA:            1300 [KW]
WAGA:                   1000 [KG]
DANE POSIADACZA;
IMIE:                   Jan
NAZWISKO:               Kowalski
NARODOWOSC:             Polska
MIASTO ZAMIESZKANIA:    Krakow
ADRES:                  Krakowska 2
DATA URODZENIA:         2/2/1999
```

Po wybraniu opcji nr.3 pojazd zostanie usunięty.

```
WPROWADZ NUMER REJESTRACYJNY :KK23W4
niepoprawnie wprowadzone dane...
```

Gdy wybierzemy opcje nr.4 nic się nie stanie.

Niezależnie od wybranej opcji program wróci ostatecznie do menu głównego.

Wyszukanie pojazdów danego typu;

Po wybraniu tej opcji zostaniemy zapytani o typ pojazdów jaki chcemy wyszukać.

```
Wprowadz typ pojadu;
1. Samochod osobisty
2. Samochod ciezarowy
3. Motocykl
```

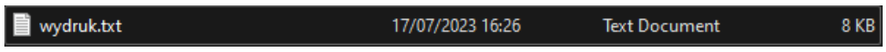
Gdy wybierzemy np opcje nr.1 ukażą się wszystkie zapisane samochy osobiste.

```
MARKA:                  Opel
MODEL:                  Adam
RODZAJ:                 Samochod osobowy
NUMER REJESTRACYJNY:    KR92PL2
PRZEBIEG:               98096 [KM]
ILOSC PASAZEROW:        3
MOC SILNIKA:            1400 [KW]
WAGA:                   1250 [KG]
DANE POSIADACZA;
IMIE:                   Jan
NAZWISKO:               Kowalski
NARODOWOSC:             Polska
MIASTO ZAMIESZKANIA:    Chrzanow
ADRES:                  Aleje 2
DATA URODZENIA:         2/02/2002

MARKA:                  Opel
MODEL:                  Adam
RODZAJ:                 Samochod osobowy
NUMER REJESTRACYJNY:    KR92PL3
PRZEBIEG:               98096 [KM]
ILOSC PASAZEROW:        3
MOC SILNIKA:            1400 [KW]
WAGA:                   1250 [KG]
DANE POSIADACZA;
IMIE:                   Janusz
NAZWISKO:               Milewski
NARODOWOSC:             Polska
MIASTO ZAMIESZKANIA:    Krakow
ADRES:                  Raclawicka 20
DATA URODZENIA:         20/02/2000

MARKA:                  Toyota
MODEL:                  Yaris
RODZAJ:                 Samochod osobowy
NUMER REJESTRACYJNY:    PO54KRS
PRZEBIEG:               98050 [KM]
ILOSC PASAZEROW:        3
MOC SILNIKA:            1200 [KW]
WAGA:                   1450 [KG]
DANE POSIADACZA;
IMIE:                   Alex
NAZWISKO:               Jonson
NARODOWOSC:             USA
MIASTO ZAMIESZKANIA:    Alabama
ADRES:                  April 245d
DATA URODZENIA:         10/02/1976
```

Następnie zostaniemy zapytani czy chcemy wydrukować dane. Po wpisaniu “TAK” dane zostaną zapisane w pliku wydruk.txt.

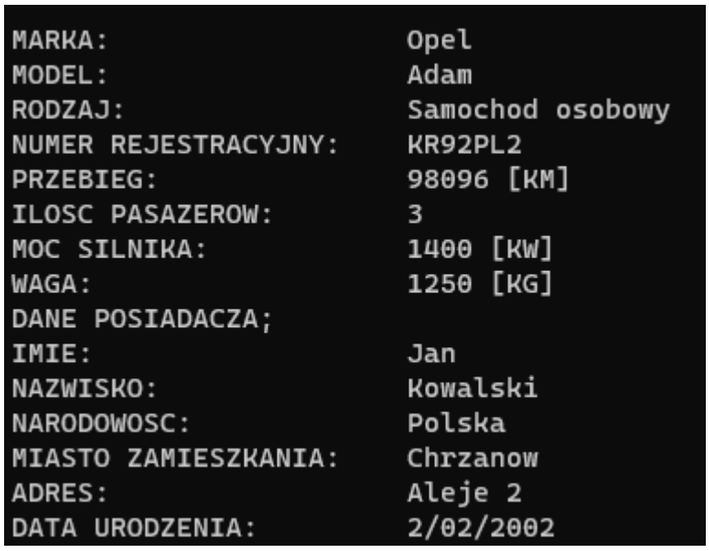


A następnie program powróci do menu głównego.

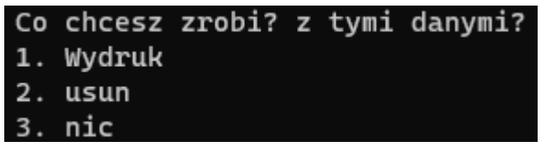
Wyszukanie pojazdów danego właściciela;

Gdy wybierzemy tę opcję, będziemy znaleźć pojazdy danego właściciela po wprowadzeniu danych takich jak: imie, nazwisko, dzień urodzenia, miesiąc urodzenia I rok urodzenia.

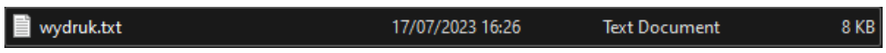
Po wprowadzeniu danych ukażą nam się pojazdy tego właściciela.



Oraz zostanie wyświetlona lista akcji do zrobienia z tymi danymi.



Po wybraniu opcji nr.1 dane zostaną zapisane do pliku wydruk.txt.



Gdy wybierzemy opcje nr.2 dane zostaną usunięte.

Po wybraniu opcji nr.3 nic się nie stanie.

Niezależnie od wybranej opcji program powróci do menu głównego.

Rejestr Zarejestrowanych Pojazdów

Wygenerowano przez Doxygen 1.9.6

Rozdział 1

Indeks hierarchiczny

1.1 Hierarchia klas

Ta lista dziedziczenia posortowana jest z grubsza, choć nie całkowicie, alfabetycznie:

Rozdział 2

Indeks klas

2.1 Lista klas

Tutaj znajdują się klasy, struktury, unie i interfejsy wraz z ich krótkimi opisami:

Rozdział 3

Indeks plików

3.1 Lista plików

Tutaj znajduje się lista wszystkich udokumentowanych plików z ich krótkimi opisami:

Rozdział 4

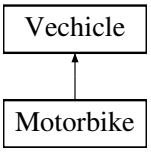
Dokumentacja klas

4.1 Dokumentacja klasy Motorbike

Klasa dla pojazdu typu Motor.

`#include <Motorbike.h>`

Diagram dziedziczenia dla Motorbike



Metody publiczne

- **Motorbike** (std::string model_, std::string marka_, int weight_, int amountOfWheels_, int amountOf↩ Passagers_, std::string type_, std::string registrationNumber_, int mileage_, std::string name_, std::string surname_, std::string homeCountry_, std::string cityOfResidence_, std::string street_, std::string number↩ OfBuilding_, int dayOfbirth_, int monthOfBirth_, int yearOfBirth_)
- void setMarka (std::string n)

  Funkcja ustawia marke pojazdu.
- void **printMarka** ()

  Funkcja wyświetla marke pojazdu na ekranie.
- void setModel (std::string n)

  Funkcja ustawia model pojazdu.
- void **printModel** ()

  Funkcja wyświetla model pojazdu na ekranie.
- int setWeight (int n)

  Funkcja ustawia wagę pojazdu.
- int getWeight ()

  Funkcja zwraca wagę pojazdu.
- void **printWeight** ()

  Funkcja wyświetla wage pojazdu na ekranie.
- int setAmountOfWheels (int n)

Funkcja ustawia liczbę kół pojazdu.

- int getAmountOfWheels ()

  Funkcja zwraca liczbę kół pojazdu.

- void **printAmountOfWheels** ()

  Funkcja wyświetla liczbę kół pojazdu na ekranie.

- int setAmountOfPassagers (int n)

  Funkcja ustawia liczbę pasażerów pojazdu.

- int getAmountOfPassagers ()

  Funkcja zwraca liczbę pasażerów pojazdu.

- void **printAmountOfPassegers** ()

  Funkcja wyświetla liczbę pasażerów pojazdu na ekranie.

- virtual void printData ()

  Fukcja wyświetla na ekranie wszystkie dane pojazdu.

- virtual void saveVechicle ()

  Fukcja zapisuje dane pojazdu.

Metody publiczne dziedziczone z Vechicle

- **Vechicle** (std::string type_, std::string registrationNumber_, int mileage_, std::string name_, std::string lastName_, std::string homeCountry_, std::string cityOfResidence_, std::string streetName_, std::string numberOfBuilding_, int dayOfBirth_, int monthOfBirth_, int yearOfBirth_)
- void **setRegistrationNumber** (std::string n)

  Fukcja ustwia numer rejestracyjny pojazu.

- void **printRegistrationNumber** ()

  Fukcja wyświetla na ekranie numer rejestracyjny pojazu.

- void **setType** (std::string n)

  Fukcja ustwia typ pojazdu.

- void **printType** ()

  Fukcja wyświetla na ekranie typ pojazu.

- void **setOwner** (std::string _name, std::string _lastName, std::string _homecountry, std::string _cityOf↩ Residence, std::string _streetName, std::string _numberOfBuilding, int _dayOfBirth, int _monthOfBirth, int _yearOfBirth)

  Fukcja ustwia właściciela pojazdu.

- void **typeOwner** ()

  Fukcja pozwala wprowadzic właściciela pojazdu przez użytkownika.

- int **setMileage** (int n)

  Fukcja ustwia przebieg pojazdu.

- int getMileage ()

  Fukcja zwraca przebieg pojazdu.

- void **printMileage** ()

  Fukcja wyświetla na ekranie przebieg pojazdu.

- virtual void printData ()=0

  Fukcja wyświetla na ekranie wszystkie dane pojazdu.

- virtual void saveVechicle ()=0

  Fukcja zapisuje dane pojazdu.

Przyjaciele

- std::istream & **operator**>> (std::istream &s, Motorbike &p)

Dodatkowe Dziedziczone Składowe

Atrybuty chronione dziedziczone z Vechicle

- std::string **type**
- std::string **registrationNumber**
- int **mileage**
- Owner **owner**

4.1.1 Opis szczegółowy

Klasa dla pojazdu typu Motor.

4.1.2 Dokumentacja funkcji składowych

4.1.2.1 getAmountOfPassagers()

```
int Motorbike::getAmountOfPassagers ( )
```

Funkcja zwraca liczbę pasażerów pojazdu.

Zwraca

int liczba pasażerów pojazdu.

4.1.2.2 getAmountOfWheels()

```
int Motorbike::getAmountOfWheels ( )
```

Funkcja zwraca liczbę kół pojazdu.

Zwraca

int liczba kół pojazdu.

4.1.2.3 getWeight()

```
int Motorbike::getWeight ( )
```

Funkcja zwraca wagę pojazdu.

Zwraca

int waga pojazdu.

4.1.2.4 printData()

```
void Motorbike::printData ( )  [virtual]
```

Fukcja wyświetla na ekranie wszystkie dane pojazdu.

Implementuje Vechicle.

4.1.2.5 saveVechicle()

```
void Motorbike::saveVechicle ( )  [virtual]
```

Fukcja zapisuje dane pojazdu.

Implementuje Vechicle.

4.1.2.6 setAmountOfPassagers()

```
int Motorbike::setAmountOfPassagers (
            int n )
```

Funkcja ustawia liczbę pasażerów pojazdu.

Parametry

n	int liczba pasażerów pojazdu

4.1.2.7 setAmountOfWheels()

```
int Motorbike::setAmountOfWheels (
            int n )
```

Funkcja ustawia liczbę kół pojazdu.

Parametry

n	int liczba kół pojazdu.

4.1.2.8 setMarka()

```
void Motorbike::setMarka (
            std::string n )
```

Funkcja ustawia marke pojazdu.

Parametry

n	string marka pojazdu.

4.1.2.9 setModel()

```
void Motorbike::setModel (
            std::string n )
```

Funkcja ustawia model pojazdu.

Parametry

n	string model pojazdu.

4.1.2.10 setWeight()

```
int Motorbike::setWeight (
            int n )
```

Funkcja ustawia wagę pojazdu.

Parametry

n	int waga pojazdu.

Dokumentacja dla tej klasy została wygenerowana z plików:

- Motorbike.h
- Motorbike.cpp

4.2 Dokumentacja klasy Owner

Klasa dla właściciela pojazdu.

```
#include <Owner.h>
```

Metody publiczne

- **Owner** (std::string name_, std::string lastName_, std::string homeCountry_, std::string cityOfResidence_, std::string streetName_, std::string numberOfBuilding_, int dayOfBirth_, int monthOfBirth_, int yearOfBirth_)
- void setName (std::string n)

  Fukcja zapisuje imie właściciela.
- void **printName** ()

  Fukcja wyświetla imie właściciela na ekranie.
- void **getName** (std::string &name_)

  Fukcja zwraca imie właściciela.
- void setLastName (std::string n)

  Fukcja zapisuje nazwisko właściciela.
- void **printLastName** ()

  Fukcja wyświetla nazwisko właściciela na ekranie.
- void **getLastName** (std::string &lastName_)

  Fukcja zwraca nazwisko właściciela.
- void setHomeCountry (std::string n)

  Fukcja zapisuje kraj pochodzenia właściciela.
- void **printHomeCoutry** ()

  Fukcja wyświetla kraj pochodzenia właściciela na ekranie.
- void **getHomeCountry** (std::string &homeCountry_)

  Fukcja zwraca kraj pochodzenia właściciela.
- void setCityOfResidence (std::string n)

  Fukcja zapisuje miejscowość zamieszkania właściciela.
- void **printCityOfResidence** ()

  Fukcja wyświetla miejscowość zamieszkania właściciela na ekranie.
- void **getCityOfResidence** (std::string &cityOfResidence_)

  Fukcja zwraca miejscowość zamieszkania właściciela.
- void setStreetName (std::string n)

  Fukcja zapisuje ulice właściciela.
- void **printStreetName** ()

  Fukcja wyświetla ulice właściciela na ekranie.
- void **getStreetName** (std::string &streetName_)

  Fukcja zwraca ulice właściciela.
- void setNumberOfBuilding (std::string n)

  Fukcja zapisuje numer budynku właściciela.
- void **printNumberOfBuilding** ()

  Fukcja wyświetla numer budynku właściciela na ekranie.
- void **getNumberOfBuilding** (std::string &numberOfBuilding_)

  Fukcja zwraca numer budynku właściciela.
- int setDayOfBirth (int n)

  Fukcja zapisuje dzień urodzenia właściciela.
- int getDayOfBirth ()

  Fukcja zwraca dzień urodzenia właściciela.
- void **printDayOfBirth** ()

  Fukcja wyświetla dzień urodzenia właściciela na ekranie.
- int setMonthOfBirth (int n)

  Fukcja zapisuje miesiąc urodzenia właściciela.
- int getMonthOfBirth ()

  Fukcja zwraca miesiąc urodzenia właściciela.
- void **printMonthOfBirth** ()

Fukcja wyświetla miesiąc urodzenia właściciela na ekranie.

- int setYearOfBirth (int n)

  Fukcja zapisuje rok urodzenia właściciela.
- int getYearOfBirth ()

  Fukcja zwraca rok urodzenia właściciela.
- void **printYearOfBirth** ()

  Fukcja wyświetla rok urodzenia właściciela na ekranie.

4.2.1 Opis szczegółowy

Klasa dla właściciela pojazdu.

4.2.2 Dokumentacja funkcji składowych

4.2.2.1 getDayOfBirth()

```
int Owner::getDayOfBirth ( )
```

Fukcja zwraca dzień urodzenia właściciela.

Zwraca

int dzień urodzenia

4.2.2.2 getMonthOfBirth()

```
int Owner::getMonthOfBirth ( )
```

Fukcja zwraca miesiąc urodzenia właściciela.

Zwraca

int miesiąc urodzenia

4.2.2.3 getYearOfBirth()

```
int Owner::getYearOfBirth ( )
```

Fukcja zwraca rok urodzenia właściciela.

Zwraca

int rok urodzenia

4.2.2.4 setCityOfResidence()

```
void Owner::setCityOfResidence (
            std::string n )
```

Fukcja zapisuje miejscowość zamieszkania właściciela.

Parametry

n	string miejscowość zamieszkania właściciela.

4.2.2.5 setDayOfBirth()

```
int Owner::setDayOfBirth (
            int n )
```

Fukcja zapisuje dzień urodzenia właściciela.

Parametry

n	int dzień urodzenia właściciela.

4.2.2.6 setHomeCountry()

```
void Owner::setHomeCountry (
            std::string n )
```

Fukcja zapisuje kraj pochodzenia właściciela.

Parametry

n	string kraj pochodzenia właściciela.

4.2.2.7 setLastName()

```
void Owner::setLastName (
            std::string n )
```

Fukcja zapisuje nazwisko właściciela.

Parametry

n	string nazwisko właściciela.

4.2.2.8 setMonthOfBirth()

```
int Owner::setMonthOfBirth (
            int n )
```

Fukcja zapisuje miesiąc urodzenia właściciela.

Parametry

n	int miesiąc urodzenia właściciela.

4.2.2.9 setName()

```
void Owner::setName (
            std::string n )
```

Fukcja zapisuje imie właściciela.

Parametry

n	string imie właściciela.

4.2.2.10 setNumberOfBuilding()

```
void Owner::setNumberOfBuilding (
            std::string n )
```

Fukcja zapisuje numer budynku właściciela.

Parametry

n	string numer budynku właściciela.

4.2.2.11 setStreetName()

```
void Owner::setStreetName (
            std::string n )
```

Fukcja zapisuje ulice właściciela.

Parametry

n	string ulice właściciela.

4.2.2.12 setYearOfBirth()

```
int Owner::setYearOfBirth (
            int n )
```

Fukcja zapisuje rok urodzenia właściciela.

Parametry

n	int rok urodzenia właściciela.

Dokumentacja dla tej klasy została wygenerowana z plików:

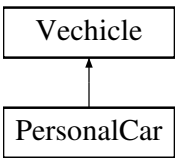
- Owner.h
- Owner.cpp

4.3 Dokumentacja klasy PersonalCar

Klasa dla pojazdu typu Samochód Osobowy.

```
#include <PersonalCar.h>
```

Diagram dziedziczenia dla PersonalCar



Metody publiczne

- **PersonalCar** (std::string model_, std::string marka_, int numberOfPassengers_, int engineCapacity_↩ , int weight_, std::string type_, std::string registrationNumber_, int mileage_, std::string name_, std::string surname_, std::string homeCountry_, std::string city_, std::string street_, std::string numberOfBuilding_, int dayOfBirth_, int monthOfBirth_, int yearOfBirth_)
- void setMarka (std::string n)

  Fukcja ustwia marka pojazu.
- void **getMarka** (std::string &n)

  Funkcja zwraca marke pojazdu.
- void **printMarka** ()

Funkcja wyświetla marke pojazdu na ekranie.

- void setModel (std::string n)

  Funkcja ustawia model pojazdu.
- void **getModel** (std::string &n)

  Funkcja zwraca model pojazdu.
- void **printModel** ()

  Funkcja wyświetla model pojazdu na ekranie.
- int setNumberOfPassegers (int n)

  Funkcja ustawia ilość pasażerów pojazdu.
- int getNumberOfPassegers ()

  Funkcja zwraca ilość pasażerów pojazdu.
- void **printNumberOfPassegers** ()

  Funkcja wyświetla ilość pasażerów pojazdu na ekranie.
- int setEngineCapacity (int n)

  Funkcja ustawia pojemność silnika pojazdu.
- int getEngineCapacity ()

  Funkcja zwraca pojemność silnika pojazdu.
- void **printEngineCapacity** ()

  Funkcja wyświetla pojemność silnika pojazdu na ekranie.
- int setWeight (int n)

  Funkcja ustawia wagę pojazdu.
- int getWeight ()

  Funkcja zwraca wagę pojazdu.
- void **printWeight** ()

  Funkcja wyświetla wagę pojazdu na ekranie.
- virtual void printData ()

  Fukcja wyświetla na ekranie wszystkie dane pojazdu.
- virtual void saveVechicle ()

  Fukcja zapisuje dane pojazdu.

Metody publiczne dziedziczone z Vechicle

- **Vechicle** (std::string type_, std::string registrationNumber_, int mileage_, std::string name_, std::string lastName_, std::string homeCountry_, std::string cityOfResidence_, std::string streetName_, std::string numberOfBuilding_, int dayOfBirth_, int monthOfBirth_, int yearOfBirth_)
- void **setRegistrationNumber** (std::string n)

  Fukcja ustwia numer rejestracyjny pojazu.
- void **printRegistrationNumber** ()

  Fukcja wyświetla na ekranie numer rejestracyjny pojazu.
- void **setType** (std::string n)

  Fukcja ustwia typ pojazdu.
- void **printType** ()

  Fukcja wyświetla na ekranie typ pojazu.
- void **setOwner** (std::string _name, std::string _lastName, std::string _homecountry, std::string _cityOf↩ Residence, std::string _streetName, std::string _numberOfBuilding, int _dayOfBirth, int _monthOfBirth, int _yearOfBirth)

  Fukcja ustwia właściciela pojazdu.
- void **typeOwner** ()

  Fukcja pozwala wprowadzic właściciela pojazdu przez użytkownika.
- int **setMileage** (int n)

  Fukcja ustwia przebieg pojazdu.

- int getMileage ()

  Fukcja zwraca przebieg pojazdu.
- void **printMileage** ()

  Fukcja wyświetla na ekranie przebieg pojazdu.
- virtual void printData ()=0

  Fukcja wyświetla na ekranie wszystkie dane pojazdu.
- virtual void saveVechicle ()=0

  Fukcja zapisuje dane pojazdu.

Przyjaciele

- std::istream & **operator**>> (std::istream &s, PersonalCar &p)

Dodatkowe Dziedziczone Składowe

Atrybuty chronione dziedziczone z Vechicle

- std::string **type**
- std::string **registrationNumber**
- int **mileage**
- Owner **owner**

4.3.1 Opis szczegółowy

Klasa dla pojazdu typu Samochód Osobowy.

4.3.2 Dokumentacja funkcji składowych

4.3.2.1 getEngineCapacity()

```
int PersonalCar::getEngineCapacity ( )
```

Funkcja zwraca pojemność silnika pojazdu.

Zwraca

int pojemność silnika pojazdu.

4.3.2.2 getNumberOfPassegers()

```
int PersonalCar::getNumberOfPassegers ( )
```

Funkcja zwraca ilość pasażerów pojazdu.

Zwraca

int ilość pasażerów pojazdu.

4.3.2.3 getWeight()

```
int PersonalCar::getWeight ( )
```

Funkcja zwraca wagę pojazdu.

Zwraca

int wagę pojazdu.

4.3.2.4 printData()

```
void PersonalCar::printData ( )  [virtual]
```

Fukcja wyświetla na ekranie wszystkie dane pojazdu.

Implementuje Vechicle.

4.3.2.5 saveVechicle()

```
void PersonalCar::saveVechicle ( )  [virtual]
```

Fukcja zapisuje dane pojazdu.

Implementuje Vechicle.

4.3.2.6 setEngineCapacity()

```
int PersonalCar::setEngineCapacity (
            int n )
```

Funkcja ustawia pojemność silnika pojazdu.

Parametry

n	int pojemność silnika pojazdu.

4.3.2.7 setMarka()

```
void PersonalCar::setMarka (
            std::string n )
```

Fukcja ustwia marka pojazu.

Parametry

n	string marka pojazdu.

4.3.2.8 setModel()

```
void PersonalCar::setModel (
            std::string n )
```

Funkcja ustawia model pojazdu.

Parametry

n	string model pojazdu.

4.3.2.9 setNumberOfPassegers()

```
int PersonalCar::setNumberOfPassegers (
            int n )
```

Funkcja ustawia ilość pasażerów pojazdu.

Parametry

n	int ilość pasażerów pojazdu.

4.3.2.10 setWeight()

```
int PersonalCar::setWeight (
            int n )
```

Funkcja ustawia wagę pojazdu.

Parametry

n	int wagę pojazdu.

Dokumentacja dla tej klasy została wygenerowana z plików:

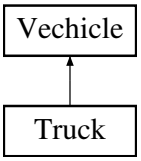
- PersonalCar.h
- PersonalCar.cpp

4.4 Dokumentacja klasy Truck

Klasa dla pojazdu typu Samochód Ciężarowy.

```
#include <Truck.h>
```

Diagram dziedziczenia dla Truck



Metody publiczne

- **Truck** (std::string model_, std::string marka_, std::string isThereATrailer_, int weight_, int maximumLoad↩ _, std::string type_, std::string registrationNumber_, int mileage_, std::string name_, std::string surname_, std::string homeCountry_, std::string cityOfResidence_, std::string street_, std::string numberOfBuilding_, int dayOfBirth_, int mounthOfBirth_, int yearOfBirth_)
- void setMarka (std::string n)

  Funkcja ustawia marke pojazdu.
- void **printMarka** ()

  Funkcja wyświetla marke pojazdu na ekranie.
- void setModel (std::string n)

  Funkcja ustawia model pojazdu.
- void **printModel** ()

  Funkcja wyświetla model pojazdu na ekranie.
- int setWeight (int n)

  Funkcja ustawia wagę pojazdu.
- int getWeight ()

  Funkcja zwraca wagę pojazdu.
- void **printWeight** ()

Funkcja wyświetla wagę pojazdu na ekranie.

- void setIsThereATrailer (std::string n)

  Funkcja ustawia czy pojazd zawiera przyczepę.
- void **printIsThereATrailer** ()

  Funkcja wyświetla czy pojazd zawiera przyczepę na ekranie.
- int setMaximumLoad (int n)

  Funkcja ustawia maksymalną ładowność pojazdu.
- int getMaximumLoad ()

  Funkcja maksymalną ładowność pojazdu.
- void **printMaximumLoad** ()

  Funkcja wyświetla maksymalną ładowność pojazdu na ekranie.
- virtual void printData ()

  Fukcja wyświetla na ekranie wszystkie dane pojazdu.
- virtual void saveVechicle ()

  Fukcja zapisuje dane pojazdu.

Metody publiczne dziedziczone z Vechicle

- **Vechicle** (std::string type_, std::string registrationNumber_, int mileage_, std::string name_, std::string lastName_, std::string homeCountry_, std::string cityOfResidence_, std::string streetName_, std::string numberOfBuilding_, int dayOfBirth_, int monthOfBirth_, int yearOfBirth_)
- void **setRegistrationNumber** (std::string n)

  Fukcja ustwia numer rejestracyjny pojazu.
- void **printRegistrationNumber** ()

  Fukcja wyświetla na ekranie numer rejestracyjny pojazu.
- void **setType** (std::string n)

  Fukcja ustwia typ pojazdu.
- void **printType** ()

  Fukcja wyświetla na ekranie typ pojazu.
- void **setOwner** (std::string _name, std::string _lastName, std::string _homecountry, std::string _cityOf↩ Residence, std::string _streetName, std::string _numberOfBuilding, int _dayOfBirth, int _monthOfBirth, int _yearOfBirth)

  Fukcja ustwia właściciela pojazdu.
- void **typeOwner** ()

  Fukcja pozwala wprowadzic właściciela pojazdu przez użytkownika.
- int **setMileage** (int n)

  Fukcja ustwia przebieg pojazdu.
- int getMileage ()

  Fukcja zwraca przebieg pojazdu.
- void **printMileage** ()

  Fukcja wyświetla na ekranie przebieg pojazdu.
- virtual void printData ()=0

  Fukcja wyświetla na ekranie wszystkie dane pojazdu.
- virtual void saveVechicle ()=0

  Fukcja zapisuje dane pojazdu.

Przyjaciele

- std::istream & **operator**>> (std::istream &s, Truck &p)

Dodatkowe Dziedziczone Składowe

Atrybuty chronione dziedziczone z Vechicle

- std::string **type**
- std::string **registrationNumber**
- int **mileage**
- Owner **owner**

4.4.1 Opis szczegółowy

Klasa dla pojazdu typu Samochód Ciężarowy.

4.4.2 Dokumentacja funkcji składowych

4.4.2.1 getMaximumLoad()

```
int Truck::getMaximumLoad ( )
```

Funkcja maksymalną ładowność pojazdu.

Zwraca

int maksymalna ładowność pojazdu.

4.4.2.2 getWeight()

```
int Truck::getWeight ( )
```

Funkcja zwraca wagę pojazdu.

Zwraca

int wagę pojazdu.

4.4.2.3 printData()

```
void Truck::printData ( )  [virtual]
```

Fukcja wyświetla na ekranie wszystkie dane pojazdu.

Implementuje Vechicle.

4.4.2.4 saveVechicle()

```
void Truck::saveVechicle ( )  [virtual]
```

Fukcja zapisuje dane pojazdu.

Implementuje Vechicle.

4.4.2.5 setIsThereATrailer()

```
void Truck::setIsThereATrailer (
            std::string n )
```

Funkcja ustawia czy pojazd zawiera przyczepę.

Parametry

n	string czy pojazd zawiera przyczepę.

4.4.2.6 setMarka()

```
void Truck::setMarka (
            std::string n )
```

Funkcja ustawia marke pojazdu.

Parametry

n	string marka pojazdu.

4.4.2.7 setMaximumLoad()

```
int Truck::setMaximumLoad (
            int n )
```

Funkcja ustawia maksymalną ładowność pojazdu.

Parametry

n	int maksymalną ładowność pojazdu.

4.4.2.8 setModel()

```
void Truck::setModel (
            std::string n )
```

Funkcja ustawia model pojazdu.

Parametry

n	string model pojazdu.

4.4.2.9 setWeight()

```
int Truck::setWeight (
            int n )
```

Funkcja ustawia wagę pojazdu.

Parametry

n	int wagę pojazdu.

Dokumentacja dla tej klasy została wygenerowana z plików:

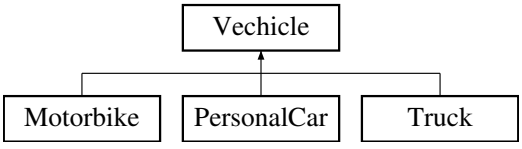
- Truck.h
- Truck.cpp

4.5 Dokumentacja klasy Vechicle

Klasa jest wspólna dla wszystkich typów pojazdów. Są wniej zapisane podstawowe cechy i parametry dla pojazdów.

```
#include <Vechicle.h>
```

Diagram dziedziczenia dla Vechicle



Metody publiczne

- **Vechicle** (std::string type_, std::string registrationNumber_, int mileage_, std::string name_, std::string lastName_, std::string homeCountry_, std::string cityOfResidence_, std::string streetName_, std::string numberOfBuilding_, int dayOfBirth_, int monthOfBirth_, int yearOfBirth_)
- void **setRegistrationNumber** (std::string n)

  Fukcja ustwia numer rejestracyjny pojazu.
- void **printRegistrationNumber** ()

  Fukcja wyświetla na ekranie numer rejestracyjny pojazu.
- void **setType** (std::string n)

  Fukcja ustwia typ pojazdu.
- void **printType** ()

  Fukcja wyświetla na ekranie typ pojazu.
- void **setOwner** (std::string _name, std::string _lastName, std::string _homecountry, std::string _cityOf↩ Residence, std::string _streetName, std::string _numberOfBuilding, int _dayOfBirth, int _monthOfBirth, int _yearOfBirth)

  Fukcja ustwia właściciela pojazdu.
- void **typeOwner** ()

  Fukcja pozwala wprowadzic właściciela pojazdu przez użytkownika.
- int **setMileage** (int n)

  Fukcja ustwia przebieg pojazdu.
- int getMileage ()

  Fukcja zwraca przebieg pojazdu.
- void **printMileage** ()

  Fukcja wyświetla na ekranie przebieg pojazdu.
- virtual void printData ()=0

  Fukcja wyświetla na ekranie wszystkie dane pojazdu.
- virtual void saveVechicle ()=0

  Fukcja zapisuje dane pojazdu.

Atrybuty chronione

- std::string **type**
- std::string **registrationNumber**
- int **mileage**
- Owner **owner**

4.5.1 Opis szczegółowy

Klasa jest wspólna dla wszystkich typów pojazdów. Są wniej zapisane podstawowe cechy i parametry dla pojazdów.

4.5.2 Dokumentacja funkcji składowych

4.5.2.1 getMileage()

```
int Vechicle::getMileage ( )
```

Fukcja zwraca przebieg pojazdu.

Zwraca

zwraca int przebieg pojazdu

4.5.2.2 printData()

```
virtual void Vechicle::printData ( )  [pure virtual]
```

Fukcja wyświetla na ekranie wszystkie dane pojazdu.

Implementowany w Motorbike, PersonalCar i Truck.

4.5.2.3 saveVechicle()

```
virtual void Vechicle::saveVechicle ( )  [pure virtual]
```

Fukcja zapisuje dane pojazdu.

Implementowany w Motorbike, PersonalCar i Truck.

Dokumentacja dla tej klasy została wygenerowana z plików:

- Vechicle.h
- Vechicle.cpp

Rozdział 5

Dokumentacja plików

5.1 ChangeOwner.h

```
00001 #pragma once
00002 #include <iostream>
00003
00013 void changeOwnerRegistrationNumber(std::string registrationNumber);
```

5.2 CheckTypesOfCars.h

```
00001 #pragma once
00002 #include <iostream>
00003
00015 void checkTypesOfCars(std::string marka, std::string model, std::string type, bool& WasType,
      std::string& firstInf, std::string& secondInf, std::string& thirdInf);
```

5.3 Delete.h

```
00001 #pragma once
00002 #include <iostream>
00003
00011 void deleteVechicleRegistrationNumber(std::string registrationNumber);
```

5.4 MainMenu.h

```
00001 #pragma once
00002
00006 void drawMainMenu();
00007
00011 void drawEnd();
00012
00016 void drawDoYouWantToPrint();
00017
00021 void enterRegistrationNumber();
00022
00026 void drawWhatYouWantToDoNextRegistrationNumber();
00027
00031 void enterTypeOfCar();
00032
00036 void drawWhatYoiWantToDoNextUser();
00037
00041 void printQuestionMarka();
00042
00046 void printQuestionModel();
00047
00051 void printSaveData();
00052
00057 void printType(std::string n);
```

5.5 Motorbike.h

```
00001 #pragma once
00002 #include "Vechicle.h"
00003 #include <iostream>
00004 #include <string>
00005
00009 class Motorbike : public Vechicle
00010 {
00011     std::string model;
00012     std::string marka;
00013     int weight;
00014     int amountOfWheels;
00015     int amountOfPassagers;
00016 public:
00017     Motorbike();
00018     Motorbike(std::string model_, std::string marka_, int weight_, int amountOfWheels_, int
      amountOfPassagers_, std::string type_, std::string registrationNumber_, int mileage_, std::string
      name_, std::string surname_, std::string homeCountry_, std::string cityOfResidence_, std::string
      street_, std::string numberOfBuilding_, int dayOfbirth_, int monthOfBirth_, int yearOfBirth_);
00019     ~Motorbike();
00020
00025     void setMarka(std::string n);
00026
00030     void printMarka();
00031
00036     void setModel(std::string n);
00037
00041     void printModel();
00042
00047     int setWeight(int n);
00048
00053     int getWeight();
00054
00058     void printWeight();
00059
00064     int setAmountOfWheels(int n);
00065
00070     int getAmountOfWheels();
00071
00075     void printAmountOfWheels();
00076
00081     int setAmountOfPassagers(int n);
00082
00087     int getAmountOfPassagers();
00088
00092     void printAmountOfPassegers();
00093
00097     virtual void printData();
00098
00102     virtual void saveVechicle();
00103
00104     friend std::istream& operator » (std::istream& s, Motorbike& p);
00105 };
00106
```

5.6 Owner.h

```
00001 #pragma once
00002 #include <iostream>
00003 #include <string>
00004
00008 class Owner
00009 {
00010     std::string name;
00011     std::string lastName;
00012     std::string homeCountry;
00013     std::string cityOfResidence;
00014     std::string streetName;
00015     std::string numberOfBuilding;
00016     int dayOfBirth;
00017     int monthOfBirth;
00018     int yearOfBirth;
00019 public:
00020     Owner();
00021     Owner(std::string name_, std::string lastName_, std::string homeCountry_, std::string
      cityOfResidence_, std::string streetName_, std::string numberOfBuilding_, int dayOfBirth_, int
      monthOfBirth_, int yearOfBirth_);
00022     ~Owner();
00023
00028     void setName(std::string n);
00029
00033     void printName();
```

```
00034
00038     void getName(std::string& name_);
00039
00044     void setLastName(std::string n);
00045
00049     void printLastName();
00050
00054     void getLastName(std::string& lastName_);
00055
00060     void setHomeCountry(std::string n);
00061
00065     void printHomeCoutry();
00066
00070     void getHomeCountry(std::string& homeCountry_);
00071
00076     void setCityOfResidence(std::string n);
00077
00081     void printCityOfResidence();
00082
00086     void getCityOfResidence(std::string& cityOfResidence_);
00087
00092     void setStreetName(std::string n);
00093
00097     void printStreetName();
00098
00102     void getStreetName(std::string& streetName_);
00103
00108     void setNumberOfBuilding(std::string n);
00109
00113     void printNumberOfBuilding();
00114
00118     void getNumberOfBuilding(std::string& numberOfBuilding_);
00119
00124     int setDayOfBirth(int n);
00125
00130     int getDayOfBirth();
00131
00135     void printDayOfBirth();
00136
00141     int setMonthOfBirth(int n);
00142
00147     int getMonthOfBirth();
00148
00152     void printMonthOfBirth();
00153
00158     int setYearOfBirth(int n);
00159
00164     int getYearOfBirth();
00165
00169     void printYearOfBirth();
00170 };
```

5.7 PersonalCar.h

```
00001 #pragma once
00002 #include <string>
00003 #include "Vechicle.h"
00004
00008 class PersonalCar : public Vechicle
00009 {
00010     std::string model;
00011     std::string marka;
00012     int numberOfPassegers;
00013     int engineCapacity;
00014     int weight;
00015 public:
00016     PersonalCar();
00017     PersonalCar(std::string model_, std::string marka_, int numberOfPassengers_, int engineCapacity_,
      int weight_, std::string type_, std::string registrationNumber_, int mileage_, std::string name_,
      std::string surname_, std::string homeCountry_, std::string city_, std::string street_, std::string
      numberOfBuilding_, int dayOfBirth_, int monthOfBirth_, int yearOfBirth_);
00018     ~PersonalCar();
00019
00024     void setMarka(std::string n);
00025
00029     void getMarka(std::string& n);
00030
00034     void printMarka();
00035
00040     void setModel(std::string n);
00041
00045     void getModel(std::string& n);
00046
```

```
00050     void printModel();
00051
00056     int setNumberOfPassegers(int n);
00057
00062     int getNumberOfPassegers();
00063
00067     void printNumberOfPassegers();
00068
00073     int setEngineCapacity(int n);
00074
00079     int getEngineCapacity();
00080
00084     void printEngineCapacity();
00085
00090     int setWeight(int n);
00091
00096     int getWeight();
00097
00101     void printWeight();
00102
00106     virtual void printData();
00107
00111     virtual void saveVechicle();
00112
00113     friend std::istream& operator » (std::istream& s, PersonalCar& p);
00114 };
00115
```

5.8 Print.h

```
00001 #pragma once
00002 #include <iostream>
00003
00011 void printAllBase();
00012
00020 void printRegistrationNumber(std::string registrationNumber);
00021
00029 void printTypeOfCar(int type);
00030
00031
00039 void printUserCars(std::string name, std::string surname, std::string dayOfBirth, std::string
      monthOfBirth, std::string yearOfBirth);
```

5.9 ReadBase.h

```
00001 #pragma once
00002 #include <iostream>
00003
00010 void readBase();
00011
00019 void readBaseTypeOfCar(int choice);
00020
00027 void readBaseUser(std::string& name, std::string &surname, std::string &dayOfBirth, std::string&
      monthOfBirth, std::string& yearOfBirth, bool& ifCorrect);
```

5.10 Registration.h

```
00001 #pragma once
00002
00003 void newRegister();
```

5.11 Search.h

```
00001 #pragma once
00002 #include <iostream>
00003 #include <vector>
00004 #include <fstream>
00005
00014 void searchRegistrationNumber(std::string registrationNumber, bool& correct);
```

5.12 Security.h

```
00001 #pragma once
00002 #include <iostream>
00003
00007 void cinInt(int& variable);
```

5.13 Truck.h

```
00001 #pragma once
00002 #include <string>
00003 #include "Vechicle.h"
00004
00008 class Truck : public Vechicle
00009 {
00010     std::string model;
00011     std::string marka;
00012     std::string isThereATrailer;
00013     int weight;
00014     int maximumLoad;
00015 public:
00016     Truck();
00017     Truck(std::string model_, std::string marka_, std::string isThereATrailer_, int weight_, int
      maximumLoad_, std::string type_, std::string registrationNumber_, int mileage_, std::string name_,
      std::string surname_, std::string homeCountry_, std::string cityOfResidence_, std::string street_,
      std::string numberOfBuilding_, int dayOfBirth_, int mounthOfBirth_, int yearOfBirth_);
00018     ~Truck();
00019
00024     void setMarka(std::string n);
00025
00029     void printMarka();
00030
00035     void setModel(std::string n);
00036
00040     void printModel();
00041
00046     int setWeight(int n);
00047
00052     int getWeight();
00053
00057     void printWeight();
00058
00063     void setIsThereATrailer(std::string n);
00064
00068     void printIsThereATrailer();
00069
00074     int setMaximumLoad(int n);
00075
00080     int getMaximumLoad();
00081
00085     void printMaximumLoad();
00086
00087
00091     virtual void printData();
00092
00096     virtual void saveVechicle();
00097
00098     friend std::istream& operator » (std::istream& s, Truck& p);
00099 };
00100
```

5.14 Vechicle.h

```
00001 #pragma once
00002 #include "Owner.h"
00003 #include <iostream>
00004 #include <string>
00005
00009 class Vechicle
00010 {
00011 protected:
00012     std::string type;
00013     std::string registrationNumber;
00014     int mileage;
00015     Owner owner;
00016 public:
00017     Vechicle();
```

```
00018     Vechicle(std::string type_, std::string registrationNumber_, int mileage_, std::string name_,
      std::string lastName_, std::string homeCountry_, std::string cityOfResidence_, std::string
      streetName_, std::string numberOfBuilding_, int dayOfBirth_, int monthOfBirth_, int yearOfBirth_);
00019     ~Vechicle();
00020
00024     void setRegistrationNumber(std::string n);
00025
00029     void printRegistrationNumber();
00030
00034     void setType(std::string n);
00035
00039     void printType();
00040
00044     void setOwner(std::string _name, std::string _lastName, std::string _homecountry, std::string
      _cityOfResidence, std::string _streetName, std::string _numberOfBuilding, int _dayOfBirth, int
      _monthOfBirth, int _yearOfBirth);
00045
00049     void typeOwner();
00050
00054     int setMileage(int n);
00055
00060     int getMileage();
00061
00065     void printMileage();
00066
00070     virtual void printData() = 0;
00071
00075     virtual void saveVechicle() = 0;
00076 };
00077
```

Skorowidz