Wiktor Komorowski

PROE

sem. 2

mgr inż. Marcin Kamil Bączyk

**Aplikacja/Moduł do obsługi dystrybutorów na stacji benzynowej GUI**

**2. Interface aplikacji**

- Uruchamiany w trybie okienkowym z pobraniem danych podczas działania programu

Wybór odstawy lub tankowania

Wybór rodzaju paliwa

Wybór ilości paliwa

**3. Oczekiwany wynik działania programu**

-Sposób prezentowania wyników

Każdorazowo wypisywana jest do pliku oraz na ekran ilość zatankowanego paliwa, należność do zapłaty i stan paliwa na stacji.

**4. Analiza obiektów występujących w trakcie działania programu**

-Fuel

Reprezentuje paliwo i zawiera metody takie jak tankowanie samochodu czy dostawę paliwa

-Disel

Obiekt zawiera informacje o cenie na dm3 i ilość dostępnego tego rodzaju paliwa

-Petrol

Obiekt zawiera informacje o cenie na dm3 i ilość dostępnego tego rodzaju paliwa

-Customer

Obiekt przetrzymuje aktualne informacje o stanie tankowania oraz należności za nie

-raportGenerator

Obiekt generyjący raport z działania programu(spis obsłużonych osób)

-Distibutor

Opowiedzialna za proces tankowania

-SupplyMenu

Okienko Menu dostaw

-Gas

Obiekt zawiera informacje o cenie na dm3 i ilość dostępnego tego rodzaju paliwa

-Refuel

Okienko menu tankowania

-RefuelnigSequence

Okienko do przeprowadzania procesu tankowania

-SupplySequence

Okienko do przeprowadzania procesu dostaw

-MainWindow

Okienko zawierające menu główne

-Raport

Okienko wyświetlające raport na bieżąco

-Bill

Okienko wyświetlające pojedynczy paragon

-Authenticator

Okienko logowania do menu dostaw

**5. Przypadki użycia**

Pokazanie menu->Analiza danych wejściowych->Przekierowanie ich do odpowiednich obiektów->Wyświetlenie działania oraz aktualizacja raportu->Pokazanie menu…

**6. Statyczna analiza typów i interfejsów poszczególnych typów**

-Fuel

-Abstakcyjna

-Klasa bazowa

-zawiera metody wirtualne

-Disel

-Zawiera pola

float petrolStationLevel-ilość dostępnego paliwa

cost(price) – cena za dm3

string - nazwa

-Korzysta z metod klasy Fuel

- Petrol

-Zawiera pola

float petrolStationLevel-ilość dostępnego paliwa

cost(price) – cena za dm3

string - nazwa

-Korzysta z metod klasy Fuel

-Gas

-Zawiera pola

float petrolStationLevel-ilość dostępnego paliwa

float cost(price) – cena za dm3

string - nazwa

-Korzysta z metod klasy Fuel

-Customer

-Zwiera pola

refuelLevel-ilość zatankowanego paliwa

-Zawiera metodę

Refuel

-raportGenerator

-Zawiera metody

makeReFuelingRaport– dodaje do pliku kolejne paragony

makeSypplyRaport – dodaj do pliku kolejne dostawy

-MainWindow

-Zawiera metody

alocateData-alokuje wszystkie potrzebne obiekty

on\_refuel\_clicked-obsługuje przycisk wyboru tankowania

on\_resource\_clicked-obsługuje przycisk wyboru dostawy

-Authentocator

-Zawiera metody

on\_confirme\_clicked-obsługuje przycisk zatwierdzenia hasła

-Raport

-Zwiera metody

printRefuel-wypisuje na ekran raport z tankowania

printSypply-wypisuje na ekran raport z dostawy

-Bill

-Zawiera metody

printBill-wypisuje na ekran pojedynczy paragon

Spis klas:

**Authentication**

#ifndef AUTHENTICATION\_H

#define AUTHENTICATION\_H

#include <QDialog>

#include <memory>

#include "supplymenu.h"

namespace **Ui** {

class **Authentication**;

}

class **Authentication** : public QDialog

{

Q\_OBJECT

public:

explicit **Authentication**(std::shared\_ptr<SupplyMenu> a, QWidget \*parent = nullptr);

~***Authentication***();

private slots:

void **on\_Comfirm\_clicked**();

private:

Ui::Authentication \*ui;

std::shared\_ptr<SupplyMenu> supplyMenu;

};

#endif // AUTHENTICATION\_H

**Bill**

#ifndef BILL\_H

#define BILL\_H

#include <QDialog>

#include <memory>

#include <string>

#include <QString>

#include "customer.h"

namespace **Ui** {

class **Bill**;

}

class **Bill** : public QDialog

{

Q\_OBJECT

public:

explicit **Bill**(std::shared\_ptr<Customer> a, QWidget \*parent = nullptr);

void **getCustomerParameters**();

void **printBill**();

~***Bill***();

private slots:

void **on\_pushButton\_pay\_clicked**();

private:

Ui::Bill \*ui;

std::shared\_ptr<Customer> customer;

double customerFee, customerFuel;

std::string fuelName;

};

#endif // BILL\_H

**Customer**

#ifndef CUSTOMER\_H

#define CUSTOMER\_H

#include <string>

class **Customer**

{

private:

double bill, fuelTaken;

std::string fuelName;

public:

**Customer**(std::string a="Nazwa paliwa" ,double b=0, double c=0);

void **saveBill**(std::string a, double b, double c);

double **giveFuelTaken**();

std::string **giveFuelName**();

double **giveBill**();

};

#endif // CUSTOMER\_H

**Disel**

#ifndef DISEL\_H

#define DISEL\_H

#include "fuel.h"

#include <string>

class **Disel** : public Fuel

{

private:

double stationFuleLevel, price;

std::string fuelName;

public:

**Disel**(double a=1000, double b=3.89, std::string c="Disel");

virtual void ***giveFuel***(double howMuch);

virtual void ***loadFuel***(double howMuch);

virtual double ***giveFuelStatus***();

virtual double ***giveFuelPrice***();

virtual std::string ***giveFuelName***();

};

#endif // DISEL\_H

**Distributor**

#ifndef DISTRIBUTOR\_H

#define DISTRIBUTOR\_H

#include "fuel.h"

#include <chrono>

#include <memory>

class **Distributor**

{

private:

std::chrono::steady\_clock::time\_point start;

std::chrono::steady\_clock::time\_point stop;

double reFuelingTime, reFuelingSpeed, stationFuelLevel, fuelTaken, calculatedFee;

std::shared\_ptr<Fuel> fuelType;

public:

**Distributor**(std::shared\_ptr<Fuel> a=nullptr, double b=0, double c=1.35, double d=0, double e=0, double f=0);

void **setFuelType**(std::shared\_ptr<Fuel> fuelType\_);

void **getStationFuelLevel**();

float **giveStationFuelLevel**();

void **startRefueling**();

void **stopRefueling**();

void **wipeAll**();

double **giveFuelTaken**();

double **giveCalculatedFee**();

};

#endif // DISTRIBUTOR\_H

**Fuel**

#ifndef FUEL\_H

#define FUEL\_H

#include <string>

class **Fuel**

{

public:

virtual void ***giveFuel***(double howMuch)=0;

virtual void ***loadFuel***(double howMuch)=0;

virtual double ***giveFuelStatus***()=0;

virtual double ***giveFuelPrice***()=0;

virtual std::string ***giveFuelName***()=0;

virtual ~***Fuel***(){};

};

#endif // FUEL\_H

**Gas**

#ifndef GAS\_H

#define GAS\_H

#include "fuel.h"

#include <string>

class **Gas** : public Fuel

{

private:

double stationFuleLevel, price;

std::string fuelName;

public:

**Gas**(double a=1000, double b=2.48, std::string c="Gaz");

virtual void ***giveFuel***(double howMuch);

virtual void ***loadFuel***(double howMuch);

virtual double ***giveFuelStatus***();

virtual double ***giveFuelPrice***();

virtual std::string ***giveFuelName***();

};

#endif // GAS\_H

**MainWindow**

#ifndef MAINWINDOW\_H

#define MAINWINDOW\_H

#include <QMainWindow>

#include <memory>

#include "refuel.h"

#include "authentication.h"

QT\_BEGIN\_NAMESPACE

namespace **Ui** { class **MainWindow**; }

QT\_END\_NAMESPACE

class **MainWindow** : public QMainWindow

{

Q\_OBJECT

public:

**MainWindow**(QWidget \*parent = nullptr);

void **alocateData**();

~***MainWindow***();

private slots:

void **on\_refuel\_clicked**();

void **on\_resource\_clicked**();

private:

Ui::MainWindow \*ui;

std::shared\_ptr<Customer> customer;

std::shared\_ptr<Distributor> distributor;

std::shared\_ptr<Refuel> refuel;

std::shared\_ptr<SupplyMenu> supplyMenu;

std::shared\_ptr<Authentication> authentication;

std::shared\_ptr<RefuelingSequence> refuelingSequence;

std::shared\_ptr<Petrol> petrol;

std::shared\_ptr<Gas> gas;

std::shared\_ptr<Disel> disel;

std::shared\_ptr<Raport> raport;

std::shared\_ptr<SupplySequence> supplySequence;

};

#endif // MAINWINDOW\_H

**Petrol**

#ifndef PETROL\_H

#define PETROL\_H

#include "fuel.h"

#include <string>

class **Petrol** : public Fuel

{

private:

double stationFuleLevel, price;

std::string fuelName;

public:

**Petrol**(double a=1000, double b= 4.39, std::string c="Benzyna");

virtual void ***giveFuel***(double howMuch);

virtual void ***loadFuel***(double howMuch);

virtual double ***giveFuelStatus***();

virtual double ***giveFuelPrice***();

virtual std::string ***giveFuelName***();

};

#endif // PETROL\_H

**Raport**

#ifndef RAPORT\_H

#define RAPORT\_H

#include <QDialog>

#include <QString>

namespace **Ui** {

class **Raport**;

}

class **Raport** : public QDialog

{

Q\_OBJECT

public:

explicit **Raport**(QWidget \*parent = nullptr);

void **printRefuel**(QString fuelName, double customerFuel, double customerFee, int i, double fuelStatus);

void **printSupply**(QString fuelName, double supplyValue, int i, double fuelStatus);

~***Raport***();

private:

Ui::Raport \*ui;

};

#endif // RAPORT\_H

**RaportGenerator**

#ifndef RAPORTGENERATOR\_H

#define RAPORTGENERATOR\_H

#include "customer.h"

#include "fuel.h"

#include "raport.h"

//#include "supply.hpp"

#include <memory>

#include <QString>

#include <string>

#include <fstream>

#include <iostream>

#include <iomanip>

class **RaportGenerator**

{

private:

std::shared\_ptr<Fuel> fuelType;

std::shared\_ptr<Customer> customer;

std::shared\_ptr<Raport> raport;

//std::shared\_ptr<Supply> supply;

std::fstream file;

std::string fileName, fuelName;

double customerFee, customerFuel;

int i,j;

public:

**RaportGenerator**(std::shared\_ptr<Raport> c=nullptr, std::shared\_ptr<Customer> a=nullptr,

std::string d="C:/Users/wikos/Desktop/raport.txt", int e=1, std::string f="Brak", std::shared\_ptr<Fuel> g=nullptr, int h=1);

void **setFuelType**(std::shared\_ptr<Fuel> fuelType=nullptr);

void **getCustomerParameters**();

void **makeReFuelingRaport**();

void **makeSypplyRaport**(double supplyValue);

};

#endif // RAPORTGENERATOR\_H

**Refuel**

#ifndef REFUEL\_H

#define REFUEL\_H

#include <QDialog>

#include <memory>

#include "refuelingsequence.h"

#include "gas.h"

#include "petrol.h"

#include "disel.h"

namespace **Ui** {

class **Refuel**;

}

class **Refuel** : public QDialog

{

Q\_OBJECT

public:

explicit **Refuel**(std::shared\_ptr<RefuelingSequence> a=nullptr,

std::shared\_ptr<Petrol> b=nullptr, std::shared\_ptr<Gas> d=nullptr,

std::shared\_ptr<Disel> e=nullptr, QWidget \*parent = nullptr);

~***Refuel***();

private slots:

void **on\_Petrol\_clicked**();

void **on\_Gas\_clicked**();

void **on\_Disel\_clicked**();

private:

Ui::Refuel \*ui;

std::shared\_ptr<RefuelingSequence> refuelingSequnce;

std::shared\_ptr<Petrol> petrol;

std::shared\_ptr<Gas> gas;

std::shared\_ptr<Disel> disel;

};

#endif // REFUEL\_H

**RefuelingSequence**

#ifndef REFUELINGSEQUENCE\_H

#define REFUELINGSEQUENCE\_H

#include <QDialog>

#include "fuel.h"

#include "customer.h"

#include "distributor.h"

#include "raport.h"

#include "raportgenerator.h"

#include "bill.h"

namespace **Ui** {

class **RefuelingSequence**;

}

class **RefuelingSequence** : public QDialog

{

Q\_OBJECT

public:

explicit **RefuelingSequence**(std::shared\_ptr<Raport> a, QWidget \*parent = nullptr);

void **setFuelType**(std::shared\_ptr<Fuel> fuelType);

~***RefuelingSequence***();

private slots:

void **on\_pushButton\_refuel\_pressed**();

void **on\_pushButton\_refuel\_released**();

void **on\_pushButton\_finish\_clicked**();

private:

Ui::RefuelingSequence \*ui;

std::shared\_ptr<Fuel> fuelType;

std::shared\_ptr<Distributor> distributor;

std::shared\_ptr<Customer> customer;

std::shared\_ptr<Raport> raport;

std::shared\_ptr<RaportGenerator> raportgenerator;

std::shared\_ptr<Bill> bill;

};

#endif // REFUELINGSEQUENCE\_H

**SupplyMenu**

#ifndef SUPPLYMENU\_H

#define SUPPLYMENU\_H

#include <QDialog>

#include <memory>

#include "petrol.h"

#include "gas.h"

#include "disel.h"

#include "supplysequence.h"

namespace **Ui** {

class **SupplyMenu**;

}

class **SupplyMenu** : public QDialog

{

Q\_OBJECT

public:

explicit **SupplyMenu**(std::shared\_ptr<SupplySequence> a=nullptr, std::shared\_ptr<Petrol> b=nullptr, std::shared\_ptr<Gas> d=nullptr,

std::shared\_ptr<Disel> e=nullptr, QWidget \*parent = nullptr);

~***SupplyMenu***();

private slots:

void **on\_pushButton\_Petrol\_clicked**();

void **on\_pushButton\_Gas\_clicked**();

void **on\_pushButton\_Disel\_clicked**();

private:

Ui::SupplyMenu \*ui;

std::shared\_ptr<SupplySequence> supplySequenece;

std::shared\_ptr<Petrol> petrol;

std::shared\_ptr<Gas> gas;

std::shared\_ptr<Disel> disel;

};

#endif // SUPPLYMENU\_H

**SupplySequence**

#ifndef SUPPLYSEQUENCE\_H

#define SUPPLYSEQUENCE\_H

#include <QDialog>

#include <QString>

#include <QMessageBox>

#include "fuel.h"

#include "raport.h"

#include "raportgenerator.h"

namespace **Ui** {

class **SupplySequence**;

}

class **SupplySequence** : public QDialog

{

Q\_OBJECT

public:

explicit **SupplySequence**(std::shared\_ptr<Raport> a=nullptr, QWidget \*parent = nullptr);

void **setFuelType**(std::shared\_ptr<Fuel> fuelType);

~***SupplySequence***();

private slots:

void **on\_pushButton\_confirm\_clicked**();

private:

Ui::SupplySequence \*ui;

std::shared\_ptr<Fuel> fuelType;

std::shared\_ptr<Raport> raport;

std::shared\_ptr<RaportGenerator> raportGenerator;

};

#endif // SUPPLYSEQUENCE\_H

Relacje klas:

Petrol, Disel i Gas dziedziczą po Fuel

BillGenerator wykorzystuje metody klasy Customer i Fuel

Distributor wykorzystuje metody klasy Fuel

MainWindow wykorzustuje metody wszystkich klas

RaportGenerator wykorzystuje metody klas Customer, Fuel

Opis ważnych elementów:

Metoda distributor->reFuel();

Metoda wykorzystuje zawartość biblioteki „std::chrono”. Wykorzystuje ją do wyliczenia ilości zatankowanego paliwa. Robi to na podstawie czasu, który upłyną od wciśnięcia klawisza „tankuj”, aż do jego puszczenia. Potem mnoży uzyskany czas przez szybkość z jaką dystrybutor podaj paliwo(zadanym w konstruktorze) otrzymuje wynik, który następnie jest podawany do kolejnych klas.

Instrukcja:

Tankowanie:

Po uruchomieniu należy wybrać opcję tankowania, a następnie rodzaj paliwa, który chce się zatankować. Następnie trzeba przytrzymać przycisk tankuj . Potem program zaczyna liczyć czas, należy odczekać tyle ile chcemy i ponownie wykonać w.w. czynności(jest to wypisywane na ekran podczas działania programu). Na końcu program doda nasz rachunek do raportu i ponownie wyświetli Menu Główne.

Dostawa:

Gdy wyświetli się Menu Główne należy wybrać opcję dostawy i wprowadzić hasło. Pojawi się prośba o wybór rodzaju paliwa, którego dostawę chcemy przeprowadzić, a następnie prośba o wprowadzenie ilości paliwa w jaka jest w dostawie. Na końcu program doda wpis o dostawie do raportu i ponownie wyświetli Menu Główne.

Wyniki działania programu:

Zawartość pliku raport.txt: