**МИНИСТЕРСТВО ЦИФРОВОГО РАЗВИТИЯ,**

**СВЯЗИ И МАССОВЫХ КОММУНИКАЦИЙ РОССИЙСКОЙ ФЕДЕРАЦИИ**

**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ**

**«САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ТЕЛЕКОММУНИКАЦИЙ ИМ. ПРОФ. М.А. БОНЧ-БРУЕВИЧА» (СПбГУТ)**

Факультет: Инфокоммуникационных сетей и систем Кафедра: Защищенных систем связи

Дисциплина: Защищенные операционные системы

ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №7.1

Система управления транспортными средствами

*(тема отчета)*

Направление/специальность подготовки

10.03.01 Информационная безопасность

*(код и наименование направления/специальности)*

Студент:

Емельянов Н.Р. ИКБ-32

*(Ф.И.О., № группы) (подпись)*

Михайлов И.Д. ИКБ-32

*(Ф.И.О., № группы) (подпись)*

Преподаватель:

Асс. Смирнов Д.Н.

*(Должность, Ф.И.О. преподавателя) (подпись)*

Санкт-Петербург 2024

**Цель**

Написать систему управления транспортными средствами.

**Задачи**

Основные задачи на лабораторную работу:

1. Определить базовый класс "Транспортное средство", содержащий общую информацию (марка, модель, год выпуска и т. д.).
2. Создать подклассы для различных видов транспортных средств, таких как "Автомобиль", "Мотоцикл" и "Грузовик", которые наследуют базовый класс.
3. Реализовать методы для расчета расхода топлива, проведения технического обслуживания и других операций, специфичных для каждого типа транспортного средства.

**Ход выполнения**

Шаги выполнения лабораторной работы:

1. Создаем макет в QTDesigner;
2. Экспортируем макет в формате .ui;
3. Конвертируем макет из .ui в .py;
4. Подключаем макет к основному файлу при помощи import;
5. Определяем класс Vehicle с методами init, ride, tire\_service, gas\_station, display\_info для определения возможных действий с транспортными средствами;
6. Определяем подкласс Motocycle, который наследуется от класса Vehicle и переопределяет методы: display\_info и ride;
7. Аналогично пункту 6 переопределяются методы для подклассов Car и Truck;
8. Определяем класс ExampleApp, который наследует от QtWidgets.QMainWindow и bank.Ui\_MainWindow для создания оконного приложения. В этом классе определены методы для обработки взаимодействия пользователя с интерфейсом.

**Листинг программы**

import sys

from PyQt5 import QtWidgets

import bank

class BankAccount:

def \_\_init\_\_(self):

self.cash = 0

def deposit(self, money):

if money > 0:

self.cash += money

return "Текущий баланс: " + str(self.cash)

else:

return "Попытка взлома зафиксирована"

def withdraw(self, money):

if (self.cash - money) > 0:

self.cash -= money

out = "Текущий баланс: " + str(self.cash)

else:

out = "Недостаточно средств"

return out

class SavingsAccount(BankAccount):

def \_\_init\_\_(self):

super().\_\_init\_\_()

def withdraw(self, money):

if (self.cash - money) > 0:

self.cash -= money + (money \* 0.01)

out = (

"Комиссия: " + str(money \* 0.01) + "\nТекущий баланс: " + str(self.cash)

)

else:

out = "Недостаточно средств"

return out

class CurrentAccount(BankAccount):

def \_\_init\_\_(self):

super().\_\_init\_\_()

def deposit(self, money):

if money > 0:

self.cash += money \* 0.99

return (

"Комиссия: " + str(money \* 0.01) + "\nТекущий баланс: " + str(self.cash)

)

else:

return "Попытка взлома зафиксирована"

class ExampleApp(QtWidgets.QMainWindow, bank.Ui\_MainWindow):

def \_\_init\_\_(self):

super().\_\_init\_\_()

self.setupUi(self)

self.bank = BankAccount()

self.savings = SavingsAccount()

self.current = CurrentAccount()

self.selectAcc.addItems(

["Банковский счет", "Сберегательный счет", "Текущий счет"]

)

self.selectAcc.activated.connect(self.activated)

self.pushWith.clicked.connect(self.withdrawal)

self.pushDep.clicked.connect(self.deposit)

self.\_\_bank\_acc\_selected()

def activated(self, index):

if index == 0:

self.\_\_bank\_acc\_selected()

elif index == 1:

self.\_\_savings\_acc\_selected()

elif index == 2:

self.\_\_current\_acc\_selected()

def \_\_bank\_acc\_selected(self):

self.label.setText("Текущий баланс: " + str(self.bank.cash))

def \_\_savings\_acc\_selected(self):

self.label.setText("Текущий баланс: " + str(self.savings.cash))

def \_\_current\_acc\_selected(self):

self.label.setText("Текущий баланс: " + str(self.current.cash))

def withdrawal(self):

curIndex = self.selectAcc.currentIndex()

if curIndex == 0:

message = self.bank.withdraw(self.spinWith.value())

elif curIndex == 1:

message = self.savings.withdraw(self.spinWith.value())

elif curIndex == 2:

message = self.current.withdraw(self.spinWith.value())

else:

message = 'error'

self.label.setText(message)

def deposit(self):

curIndex = self.selectAcc.currentIndex()

if curIndex == 0:

message = self.bank.deposit(self.spinDep.value())

if curIndex == 1:

message = self.savings.deposit(self.spinDep.value())

if curIndex == 2:

message = self.current.deposit(self.spinDep.value())

self.label.setText(message)

def main():

app = QtWidgets.QApplication(sys.argv)

window = ExampleApp()

window.show()

app.exec\_()

main()

На рисунках 1, 2, 3 представлены окна с различными типами транспортных средств:

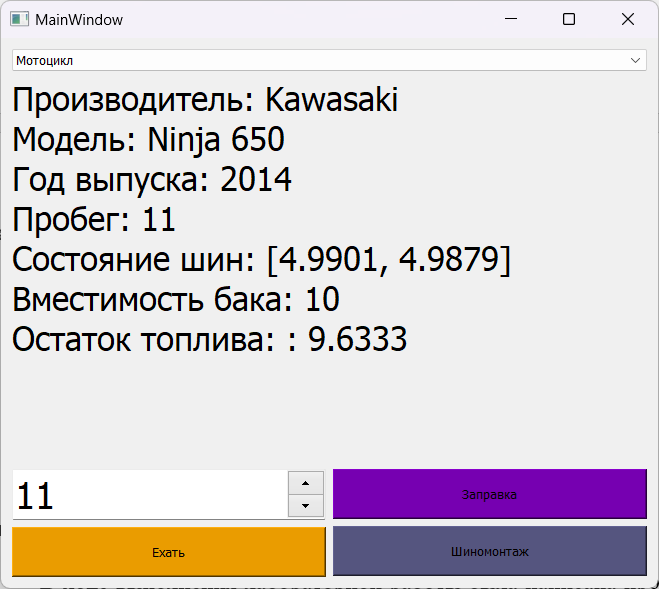


Рис. 1 – Окно с Мотоциклом

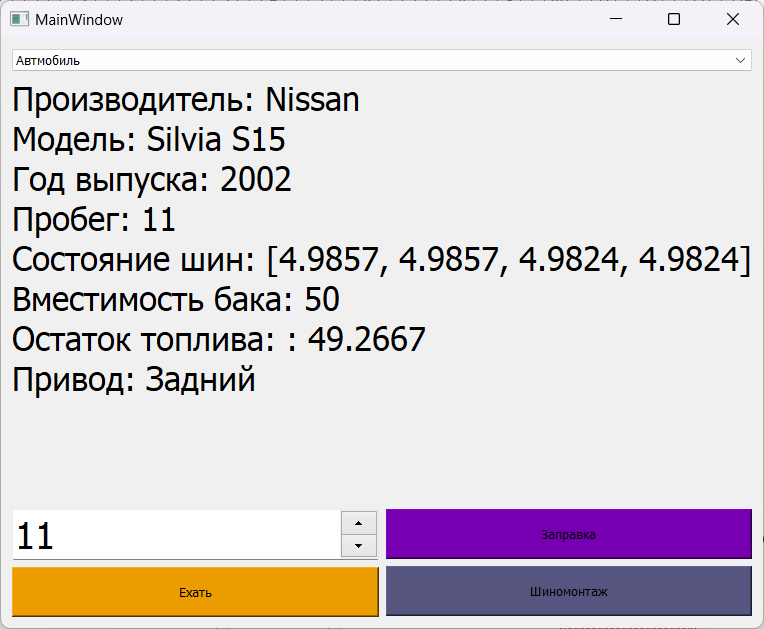


Рис. 2 – Окно с Автомобилем

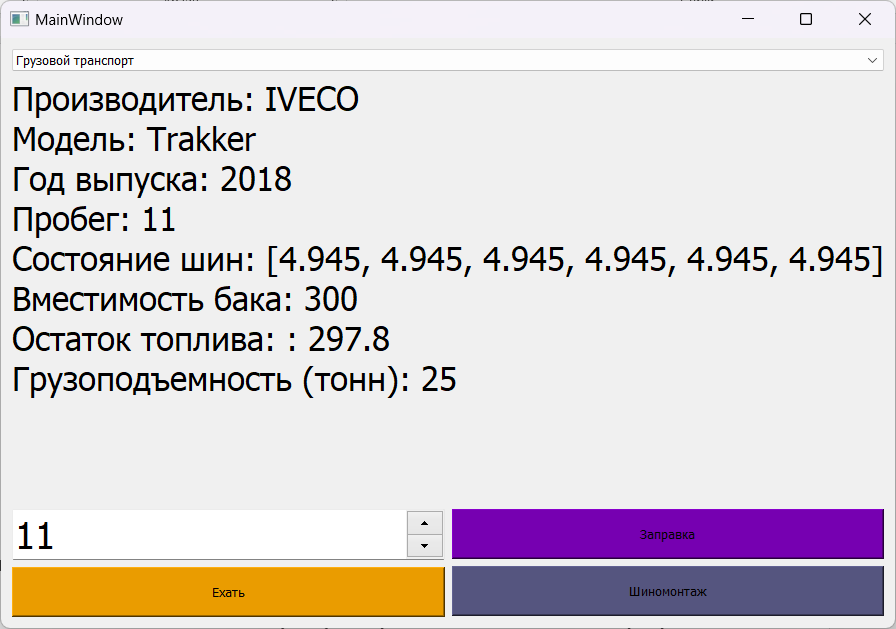


Рис. 3 – Окно с грузовым транспортом

**Вывод**

В ходе выполнения лабораторной работы была написана система управления транспортными средствами. Это было сделано в ходе выполнения следующих шагов:

1. Импортировали необходимые библиотеки
2. Создали класс для работы с разными типами транспортных средств (Vehicle)
3. Создали подклассы для мотоцикла, автомобиля и грузового транспорта (Motorcycle, Car, Truck);
4. Определили методы для каждого из них;
5. Создали класс ExampleApp для оконного приложения и прописывали взаимодействие с пользователем.