Міністерство освіти і науки України

Харківський національний університет радіоелектроніки

Кафедра Програмної інженерії

КУРСОВА РОБОТА

ПОЯСНЮВАЛЬНА ЗАПИСКА

з дисципліни «Основи об’єктно-орієнтованого програмування»

КАСА АВТОВОКЗАЛУ

Керівник, ст. викл. Черепанова Ю.Ю.

Студент гр. ПЗПІ-16-5 Брестовицький Р.М

Комісія:

проф. \_\_\_\_\_\_\_\_\_\_\_\_\_\_ Бондарєв В.М.

ст. викл. \_\_\_\_\_\_\_\_\_\_\_\_\_\_ Черепанова Ю.Ю.

ас. \_\_\_\_\_\_\_\_\_\_\_\_\_\_ Аксьоненко Б.Ф.

Харків 2017

**ХАРКІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ РАДІОЕЛЕКТРОНІКИ**

Кафедра: ***Програмної інженерії***

Дисципліна: ***Об’єктно-орієнтоване програмування***

Напрям: ***Програмна інженерія***

Курс: 1 . Група: ПЗПІ-16-5 . Семестр: 2 .

***ЗАВДАННЯ***

***на курсовий проект студента***

***Брестовицького Романа Миколайовича***

(Прізвище, Ім’я, По батькові)

1) Тема проекту: *Каса автовокзалу*

2) Тема здачі студентом закінченого проекту: **«*31» - травня - 2017 р.***

3) Вихідні дані до проекту:

*Специфікація програми, методичні вказівки до виконання курсової роботи*

4) Зміст розрахунково-пояснювальної записки:

*Вступ, специфікація програми, проектна специфікація, інструкція користувача, висновки*

5) Перелік графічного матеріалу:

*Схема об’єктної моделі, алгоритми, приклади екранних форм*

**КАЛЕНДАРНИЙ ПЛАН**

|  |  |  |
| --- | --- | --- |
| *№* | *Назва етапу* | *Термін виконання* |
| 1 | Видача теми, узгодження і затвердження теми. | 15-02-2017 р. |
| 2 | Формулювання вимог до програми. | 15-02-2017 – 22-02-2017 р. |
| 3 | Розробка підсистеми зберігання, пошуку, видалення автобусів. | 22-02-2017 – 09-03-2017 р. |
| 4 | Розробка підсистеми зберігання, пошуку, повернення квитків. | 09-03-2017 – 23-03-2017 р. |
| 5 | Розробка інтерфейсу. | 23-03-2017 – 27-03-2017 р. |
| 6 | Розробка функцій зберігання та завантаження даних. | 27-03-2017 – 10-04-2017 р. |
| 7 | Тестування і доопрацювання розробленої програмної системи. | 10-04-2017 – 28-04-2017 р. |
| 8 | Оформлення пояснювальної записки, додатків, графічного матеріалу. | 28-04-2017 – 29-05-2017 р. |
| 9 | Захист | 29-05-2017 – 31-05-2017 р. |

Студент \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Керівник \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Черепанова Ю.Ю.

« 15 » лютого   2017 року

РЕФЕРАТ

Пояснювальна записка до курсової роботи: 40 с., 16 рис., 3 додатки. Кількість частин: 2. Кількість джерел з переліку посилань: 5.

Мета роботи – проектування та створення системи для можливості обробки даних каси автостанції.

Методи розробки базуються на використанні середи розробки Microsoft Visual Studio 2016, Windows Forms, платформи .NET Framework 4.5, мови програмування C#.

У результаті розробки отримано програму в якій можливо зберігати данні рейсів (кількість вільних місць, час відправлення, проміжні зупинки, кінцевий пункт, день відправлення ), данні квитків ( ім’я та фамілія пасажира, номер автобуса, день відправлення, кінцевий пункт). Під час роботи програма зберігає всі необхідні данні для подальшої обробки, що робить її стійкою від закриття робочого вікна.

ПРОГРАМА, КУРСОВА, ОБ’ЄКТНО-ОРІЄНТОВАНЕ ПРОГРАМУВАННЯ, МОВА ПРОГРАМУВАННЯ C#, ПЛАТФОРМА .NET, КАСА АВТОВОКЗАЛУ.

ЗМІСТ

Вступ……………………………………………………………………………….6

1 Специфікація програми…………………………………………………………7

* 1. Основні вимоги та функції програми………………………………….7
  2. Опис інтерфейсу………………………………………………………..7

2 Проектна специфікація………………………………………………………...12

2.1 Об’єктна модель програми……………………………………………12

2.2 Реалізація функцій програми…………………………………………15

3 Інструкція користувача………………………………………………………...20

Висновки…………………………………………………………………....……24

Перелік посилань………………………………………………………….……..25

Додаток А – Метод пошуку рейсів «showBus» ……………………………….26

Додаток Б – Метод оновлення автобусів «updateBus»………………………..30

Додаток С – Метод додавання квитка «addTicket»……………………………34

ВСТУП

Людина завжди мала бажання освоювати нові місцевості, подорожувати, торгувати. Для поліпшення цього процесу створили транспорт, а з часом цілі системи, які дали можливість долати великі дистанції за більш короткий час, ці системи стали куди складніші зі всілякими розв’язками та об’єднаннями. Одною з ходових галузей є автобусні перевезення. Велика кількість рейсового транспорту долає простори кожну секунду. Та чи легко слідкувати за всіма транспортними засобами? Чи легко перевірити та запам’ятати всіх пасажирів?

Для покращення роботи цієї галузі з’явилася ідея створення системи, яка б допомагала слідкувати за відправлення автобусів та за усіма пасажирами. Бо для обслуговування автостанції необхідна велика кількість персоналу.

Розробка програмних систем має важливе значення, адже з часом ці системи зможуть поліпшити працю людини. Надаючи перевагу системам ми зможемо зменшити кількість помилок, які залежать від людського фактору.

В даній роботі стала задача створити систему, яка зможе формувати автобусні рейси та контролювати продаж квитків. Зберігати всю інформацію локально та надавати її при необхідності.

Мета даної курсової роботи освоїти мову програмування С#, навчитися об’єктно-орієнтованому програмуванню, розвити навички роботи в Windows Forms.

Задачі роботи:

- Розробка системи зчитування та запису даних;

- Створення логіки роботи системи;

- Робота з користувачем;

- Розробка логіки додавання, пошуку та видалення інформації;

- Тестування готового продукту;

- Відпрацьовування навичок оновлення продукту;

- Створення стійкості програми.

1 СПЕЦИФІКАЦІЯ ПРОГРАМИ

1.1 Основні вимоги та функції програми

Система каси автовокзалу має задачу полегшити робочий процес на автостанціях. Має своє значення для касира, який зможе контролювати рейси автобусів та потік квитків.

Серед основних функцій можна виділити наступні:

* Додавання нового автобуса до системи;
* Редагування вже наявного автобуса;
* Видалення старих автобусів;
* Пошук необхідного транспорту;
* Запис в базу проданих квитків;
* Пошук необхідних квитків;
* Повернення квитків в касу;
* Можливість зберігати всю інформацію.
  1. Опис інтерфейсу

При вході в програму перед нами з’являтиметься вікно контролю автобусів, у якому одразу можна буде додати новий автобус, якщо в базі вже записані рейси, то можна знайти необхідний маршрут. При необхідності ми зможемо замінити вже наявну інформацію або видалити списані маршрути.

При завантаженні перед нами з’являються (див. рис 1.1):

* Місце для запису:

1. Номеру автобуса;
2. Додаткових станцій;
3. Кінцевого пункту;
4. Загальної кількості посадових місць;
5. Часу відправлення;
6. Дня відправлення.

* Кнопки:

1. «Добавить»;
2. «Найти»;
3. «Удалить»;
4. «Изменить»;
5. «Очистить таблицу»;
6. «Очистить» (належить до області виводу);
7. «Перейти».

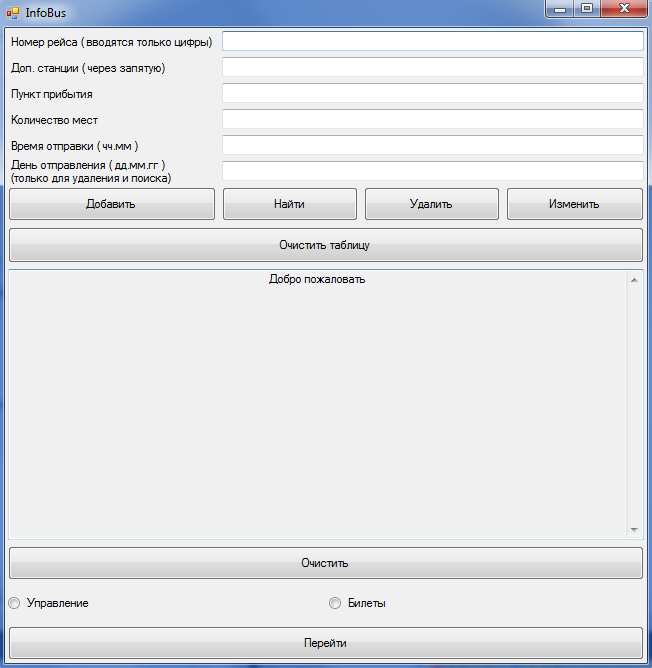


Рисунок 1.1 – Вікно контролю автобусів

На стартовому вікні ми будемо побачити ще два елементи контролю «Управление» та «Билеты».

У нижній частині буде перехід в інший відділ, який наддасть нам можливість продажу, пошуку та поверненню квитків (див. рис. 1.2). При переході у відділ квитків ми побачимо:

* Місце для запису:

1. Прізвища покупця;
2. Ім’я;
3. Пункт до якого збирається користувач;
4. Номер білету;
5. Номер потрібного рейсу;
6. День відправлення;

* Кнопки:

1. «Добавить»;
2. «Найти»;
3. «Вернуть»;
4. «Очистить таблицу»;
5. «Очистить» (має значення для області виводу);
6. «Перейти».

* Радіобатони:

1. «Управление»;
2. «Билеты».

Під час використання програми користувач іноді намагається зробити дії, які можуть нашкодити дії програми, для цього система має деякий захист у вигляді попереджень (див. малюнок 1.2).

При переході між меню обробки автобусів і редакції квитків, мають зберігатися всі данні, які буди внесені в попередні поля.

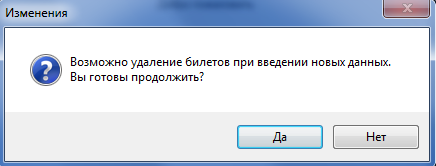


Рисунок 1.2 – Попередження перед виконанням змін

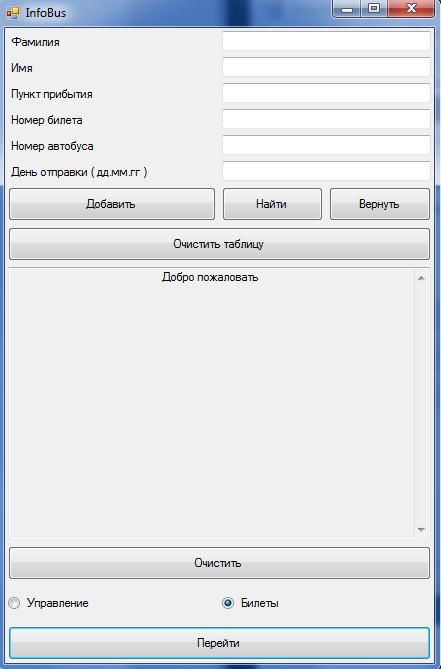


Рисунок 1.3 – Вікно контролю квитків

Під час спроби видалити всіх данних буде з’являтися запит на спеціальний запис, тільки після того як користувач введе потрібний символ свориться повне видалення.

2 ПРОЕКТНА СПЕЦИФІКАЦІЯ

2.1 Об’єктна модель програми

Програма складається з декількох класів, вікна та файлів, які створюються під час першого запуску системи і оновлюються впродовж роботи програми. Класи взаємодіють між собою та надають нам очікуваний результат [1].

На рисунку 2.1 зображено архітектуру проекту (всі наявні класи, службові файли та посилання).

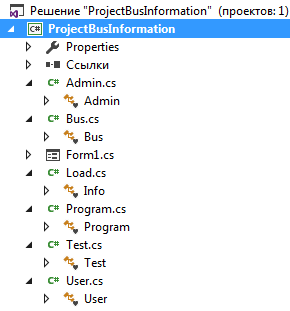


Рисунок 2.1 – Архітектура проекту

Почнемо з класу Bus, він необхідний для створення файлів, у які ми записуємо (з допомогою того ж класу) всю необхідну інформацію щодо автобусів.

На рисунку 2.2 зображено змінні, які реалізують всю необхідну інформацію о рейсах:

* Номер рейсу;
* Додаткові станції;
* Кінцевий пункт;
* Загальна кількість посадових місць;
* Кількість вільних місць, які можна купувати;
* Час відправлення;
* День відправлення.

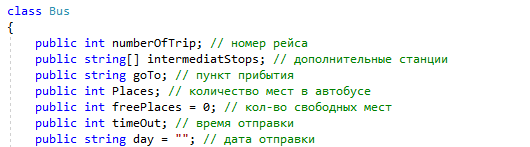


Рисунок 2.2 – Змінні класу Bus

Цей клас реалізує запис та зчитування файлів, в яких зберігається інформація рейсів, що дає змогу закрити вікно, але не втрати данні.

В класі перевизначено метод ToString(), для того, щоб при завантаженні потрібного маршруту легко можна вивести всі необхідні дані у зручному розташуванні, як показано на рисунку 2.3.

Для пошуку потрібних рейсів створено статичний метод, який відкриває директорію, у якій зберігаються дані о всіх автобусних рейсах, після чого завантажує всі записані дані та повертає масив з екземплярів класу Bus (див. рис. 2.4).

Клас User дуже схожий на клас Bus, але він реалізує запис, зчитування файлів, які зберігають інформацію про квитки. Цей клас має такі змінні (див. рис 2.5):

* Номер білету;
* Прізвище;
* Ім’я;
* Кінцевий пункт;
* Номер автобуса;
* День відправлення.

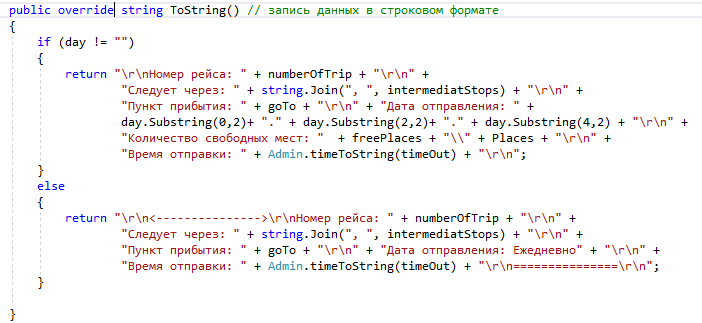


Рисунок 2.3 – Перевизначений метод ToString() в класі Bus

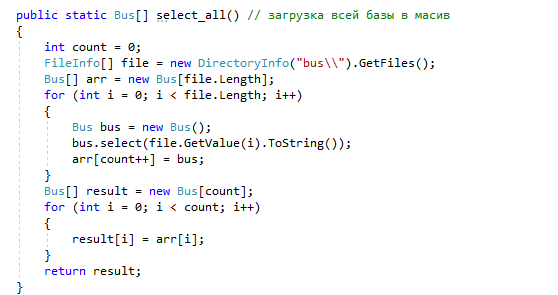


Рисунок 2.4 – Метод, який повертає всі наявні автобуси

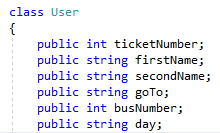


Рисунок 2.5 Змінні які належать класу User

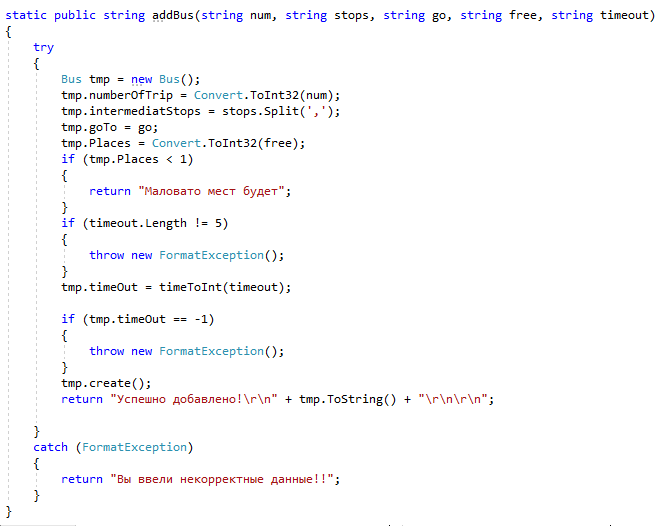
Усі необхідні функції, які обробляють процес викликані користувачем перевіряються спочатку у класі BusInfo (наслідує влаштований клас Form), а затим передаються у статичний клас Admin, який вже контролює усі процеси, які пов’язані зі зміною вхідних даних, з оновленням вже наявних, та з виводом вихідних даних [2].

2.2 Реалізація функцій програми

Більш детально розглянемо клас Admin, який є головним класом. Він отримує запити від класу BusInfo і вхідні данні (користувача) та викликає необхідні методи з інших класі та з самого себе (див. рис. 2.6).

При створенні нового автобуса, статичний метод addBus використовує метод переводу часу в числовий вид (див. рис. 2.7) та визиває метод запису нового автобуса до файлу (tmp.creat()).

Коли ми створюємо пошук необхідного автобуса, нам передаються всі необхідні данні з полів для вводу, які використовував користувач(див. рис. 2.8). Після того, як ми отримали необхідну інформацію, ми завантажуємо всі наявні автобуси. Далі, перевіряємо кожний рейс на наявність необхідної для пасажира зупинки, якщо це не є критерієм пошуку, то ми перевіряємо усю іншу надану інформацію. У будь-якому випадку будуть представлені тільки автобуси з наявністю вільних місць (або буде показано загальну інформацію маршруту). Коли необхідні автобуси знайдені, то вони сортуються за зростанням часу відправлення, результат передається назад до головної форми. Повний код методу розміщено у додатку А.



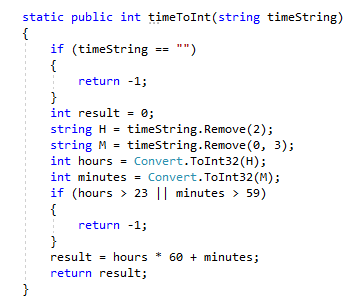
Рисунок 2.6 – Метод створення нового автобуса

Рисунок 2.7 – Метод перетворення часу в числовий формат

Щоб оновити автобус, користувач дає згоду на можливість видалення квитків, які не будуть підходити під критерії нових даних маршруту. Обов’язковою інформацією є номер рейсу. Всі квитки, які можуть буди видалені записуються в пам'ять, після чого, автобус зі старою інформацією видаляється. Далі, створюється в статичній пам’яті новий автобус і ми туди за допомогою методу додавання квитків записуємо наші збережені білети. Квитки, яких не задовольняє нові данні автобуса, зберігаються і повертаються назад до головної форми для виводу [3]. Ознайомитися з кодом можна в додатку Б.

Видалення автобусів відбувається в такій послідовності(див. рис. 2.8):

1. Завантаження всіх автобусів;
2. Пошук маршрутів для видалення;
3. Запис тих рейсів, які необхідно видалити;
4. Пошук квитків, які необхідно повернути (вони запам’ятовуються для повернення в головну форму);
5. Видалення квитків та автобусів з бази;
6. Повернення результатів о квитках, які були видалені.

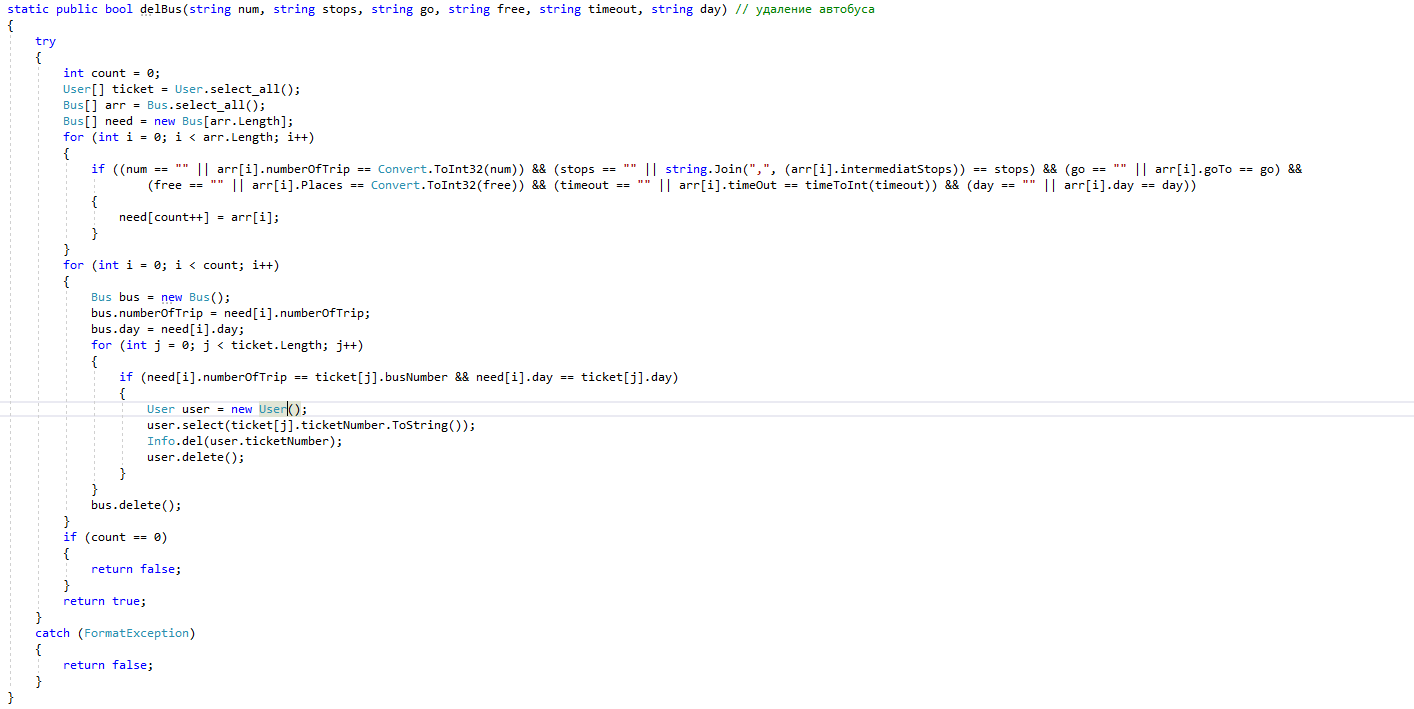


Рисунок 2.8 – Метод видалення автобусів

При намірах купити квиток, необхідно ввести до програми персональні дані пасажира, номер білету, номер потрібного автобуса та пункт прибуття. Після цього, система перевіряє чи правильно введена дата відправлення, чи не намагається пасажир купити квиток на автобус, який вже відправився. Система перевіряє, щоб персональний код квитка не був зайнятий кимось іншим. По даним номера автобуса, цей автобус завантажується та перевіряється на наявність вільним місць і на можливість пасажира доїхати до бажаного пункту. Якщо все збігається, то квиток записується до бази, а метод повертає позитивний результат та данні квитка [4]. Повний текст метода розміщено в додатку С.

При необхідності пошуку вже наявних квитків, використовується метод showTicket, який завантажує всі наявні квитки в базі. Після чого порівнює критерії пошуку, які задає користувач, та білети з бази, при збіжності зберігає данні про квиток. Після проходу всіх квитків повертає, всі можливі варіанти (див. рис. 2.10).

Коли виникає необхідність повернути квиток, метод для видалення завантажує білет для видалення та автобус, на який був куплений цей квиток. У автобусі додається вільне місце, а квиток видаляється з бази (див. рис. 2.9).

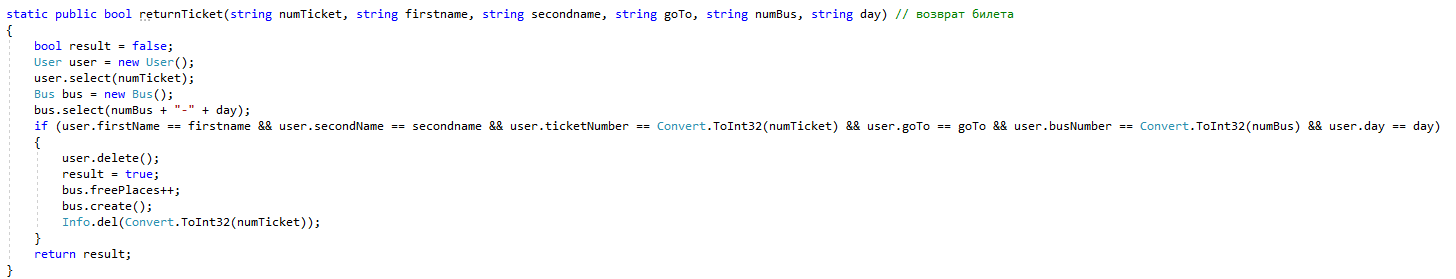


Рисунок 2.9 – Метод повернення квитків

У головній формі відбувається лише передача даних з полів вводу до статичного класу Admin та навпаки передача вихідної інформації користувачу. Прикладом може бути метод реєстрації квитка (див. рис. 2.11).

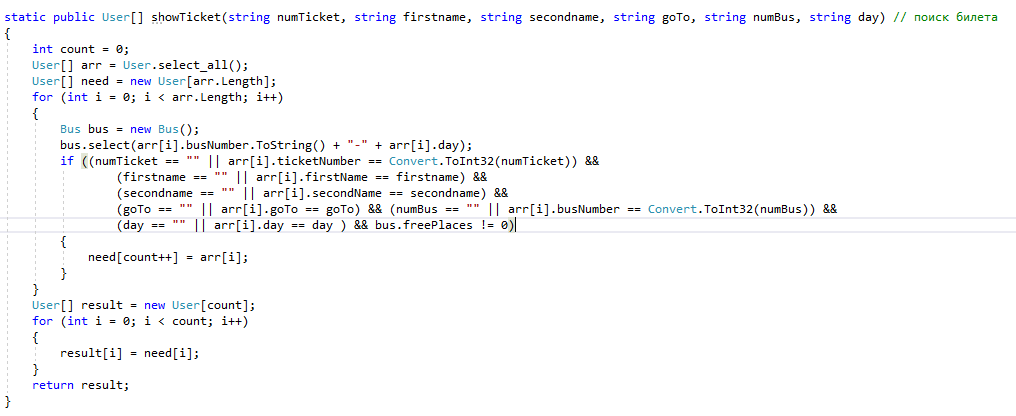


Рисунок 2.10 – Метод пошуку квитків

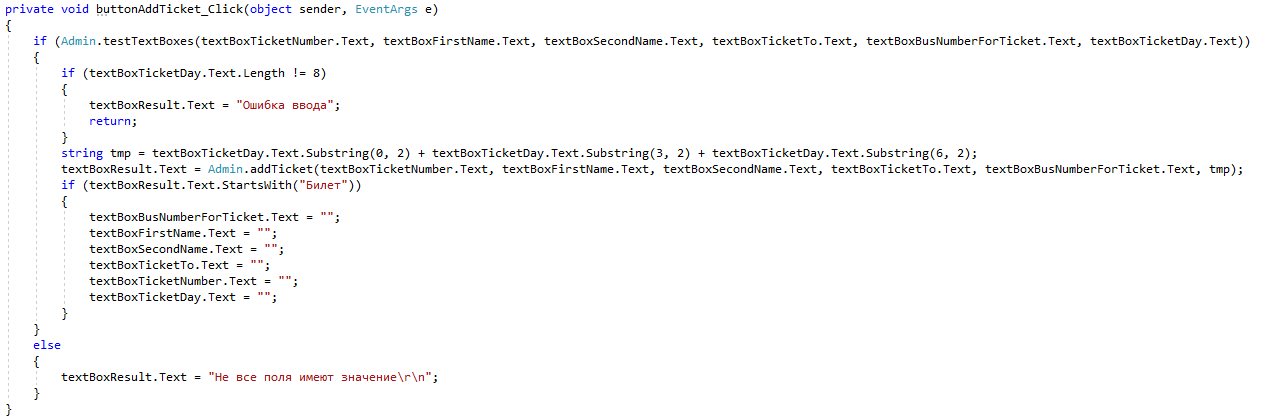


Рисунок 2.11 – Метод реєстрації квитка

При першому завантаженні програми, система перевіряє наявність необхідних для роботи файлів, якщо таких немає, програма створює їх для коректної роботи.

3 ІНСТРУКЦІЯ КОРИСТУВАЧА

Для користування програмою її необхідно завантажити на ваш комп’ютер. Після запуску, вам буде доступне вікно налаштування автобусних рейсів та можливість переходу до налаштування квитків (див. рис. 3.1) [5].

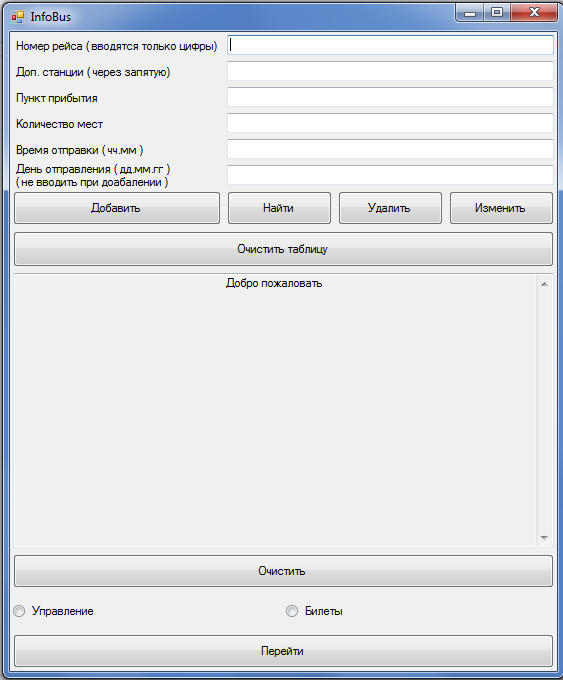


Рисунок 3.1 – Вікно налаштування автобусів

Номер рейсу. Це найголовніше, що має автобус, це те, що відрізняє його від інших. Номер вводиться тільки цілими цифрами.

Додаткові станції. Долаючи простори, ми зможемо відвідувати додаткові станції. Інформація в це поле вводиться тільки літерами через кому, без пробілів, кожне слово з великої літери.

Кінцевий пункт. Місце, де закінчується подорож.

Кількість місць. Кількість можливих посадових місць, які може надати автобус.

Час відправлення. Час в який автобус відправляється з нашої станції. Вводиться чотирма цілими цифрами розділеними крапкою (перші дві цифри позначають години, другі дві – хвилини). В дні не може буди більше ніж двадцять три години, а в годині – п’ятдесят дев’ять хвилин.

День відправлення. День, коли автобус відправляється з нашої станції. Вводиться спочатку дві цифри, які позначають день, потім, через крапку дві цифри, які позначають місяць, через крапку дві цифри, які позначають рік. Приклад: двадцять сьомого травня дві тисячі сімнадцятого року позначається як «27.05.17».

Щоб додати новий автобус необхідно ввести всі поля, окрім дня відправлення. Для пошуку, можна вводити будь-які данні, які б ви хотіли побачити. Для видалення треба заповнити хоча б одне поле, але якщо в «Номер рейса» ввести значення «-1» та натиснути «Удалить», то видаляться всі наявні дані. Щоб змінити деяку інформацію обов’язково треба заповнити всі поля (день відправлення заповнюється по необхідності), або ввести необхідний номер рейсу натиснути кнопку «Изменить» і всі діючі данні будуть внесені в відповідні поля.

При переході до розділу квитків перед нами зявляється відповідне вікно (див. рис. 3.2).

Прізвище та ім’я. Персональні дані, які надає пасажир. Вводяться у відповідні поля з великої літери.

Кінцевий пункт. Місце до якого їде власник квитка. Вводиться з великої літери.

Номер квитка. Персональний код, який надається пасажиру. Для того, щоб програма сама надавала персональний номер, достатньо в полі ввести значення «0», оскільки нульового квитка не існує, то програма розуміє це, як необхідність генерації номера самостійно.

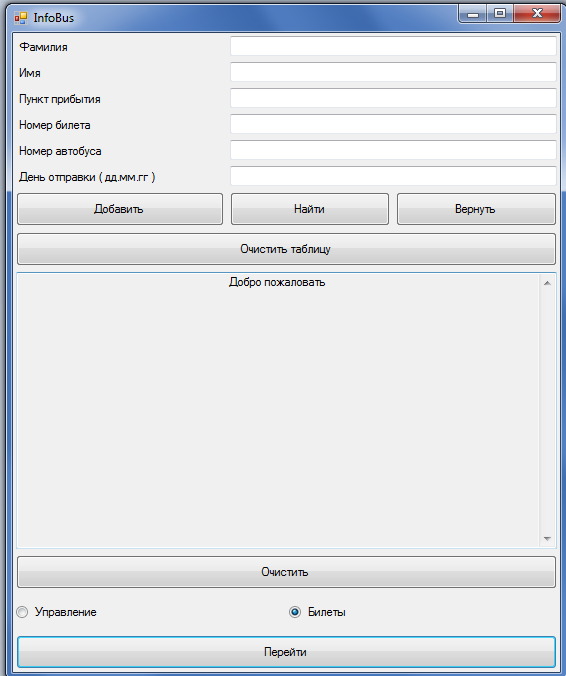


Рисунок 3.2 – Вікно обробки квитків

Номер автобуса. Номер того рейсу, на який придбається квиток, також автобус повинен мати кінцевий пункт.

День відправлення. День, коли відправляється автобус. Вводиться спочатку дві цифри, які позначають день, потім, через крапку дві цифри, які позначають місяць, через крапку дві цифри, які позначають рік. Приклад: двадцять сьомого травня дві тисячі сімнадцятого року позначається як «27.05.17».

При реєстрації квитка необхідно заповнити всі представлені поля для вводу. Під час пошуку можна вносити будь-яку кількість інформації.

Щоб подивитися на останній придбаний квиток, необхідно, не заповнюючи всі поля, натиснути кнопку «Вернуть» і в панелі виводу з’явиться останній придбаний білет. Щоб не вносити всі дані в поля, достатньо ввести номер квитка та натиснути кнопку повернення, у відповідних полях з’явиться вся необхідна інформація, натискаючи повторно кнопку, ви повернете квиток.

ВИСНОВКИ

В ході виконання курсової роботи були освоєні методи створення програмних систем, було використано об’єктно-орієнтоване програмування.

У результаті створено програмну систему «InfoBus», яка реєструє, змінює, видаляє та знаходить необхідні данні, які потрібні для організації руху транспорту та обробки потоку квитків. Ця система має змогу контролювати потік інформації та поділяти на необхідні групи. Обробляти усі запроси користувача та видавати коректні відгуки. Зберігати всю подану інформацію локально, мати можливість змінювати збереженні дані, додавати нові та видаляти непотрібні.

На практиці, виконуючи курсову роботу, ми познайомилися з середою розробки Microsoft Visual Studio 2016, Windows Forms, платформою .NET Framework 4.5, мовою програмування C#.

Під час створення курсової роботи придбалися навички роботи з програмною системою, створення в’язків між об’єктами.

ПЕРЕЛІК ПОСИЛАНЬ

1. Бондарев В. М. Объектно-ориентированное программирование на С# [Текст] : учеб. Пособие / В. М. Бондарев. – Харьков : СМИТ, 2009. –224 с. – Библиогр. : ISBN 978-966-2028-34-8.
2. SoloLearn. Freeprogramingcourses. C# Tutorial[Электронный ресурс] / SoloLearn –бесплатные курсы по программированию. – Режим доступа : www/URL:https://www.sololearn.com/Course/CSharp/ – 22.05.2017 г. – Загл. сэкрана.
3. Троелсен, Э. Язык программирования C# 5.0 и платформа .NET 4.5, 6-е изд. [Текст]: пер. с англ./ Э. Троелсен.- М.: Вильямс, 2013 - 1312 с. – ISBN 978-5-8459-1814-7, 978-1-43-024233-8.
4. Либерти Дж. Программирование на С# [Текст] : пер. с англ. / Дж. Либерти. – 2-е изд. – СПб. ; М. : Символ-Плюс, 2003. – 688 с.
5. Microsoft Developer Network. Библиотека MSDN. Разработка на .NET [Электронный ресурс] / MSDN – сеть разработчиков Microsoft. – Режим доступа : www/URL:https://msdn.microsoft.com/ru-ru/ - 24.05.2017 г. – Загл. с экрана.

Додаток А

Метод пошуку рейсів «showBus»

static public Bus[] showBus(string num, string stops, string go, string free, string timeout, string day) // поиск автобуса

{

try

{

string returngo = "";

bool boolreturngo = false;

int time = -1;

int count = 0;

Bus[] arr = Bus.select\_all();

Bus[] need = new Bus[arr.Length];

for (int i = 0; i < arr.Length; i++)

{

for (int j = 0; j < arr[i].intermediatStops.Length; j++)

{

if (arr[i].intermediatStops[j] == go && arr[i].goTo != go)

{

returngo = arr[i].goTo;

arr[i].goTo = go;

boolreturngo = true;

}

}

if (timeout != "" && arr[i].timeOut - timeToInt(timeout) < 0)

{

arr[i].timeOut += 1440;

time = i;

}

if ((num == "" || arr[i].numberOfTrip == Convert.ToInt32(num)) &&

(stops == "" || string.Join(",", (arr[i].intermediatStops)) == stops) &&

(go == "" || arr[i].goTo == go) &&

(free == "" || arr[i].Places == Convert.ToInt32(free)) &&

(timeout == "" || arr[i].timeOut >= timeToInt(timeout)) &&

arr[i].freePlaces != 0 && arr[i].day == day)

{

if (boolreturngo)

{

boolreturngo = false;

arr[i].goTo = returngo;

}

need[count++] = arr[i];

}

}

Bus[] result = new Bus[count];

for (int i = 0; i < count; i++)

{

result[i] = need[i];

}

Array.Sort(result, (a, b) =>

{

if (a.timeOut == b.timeOut)

{

return a.timeOut;

}

else

{

return a.timeOut - b.timeOut;

}

});

for (int i = 0; i < count; i++)

{

if (result[i].timeOut > 1440)

{

result[i].timeOut -= 1440;

}

}

return result;

}

catch (Exception)

{

return null;

}

}

Додаток Б

Метод оновлення автобусів «updateBus»

static public User[] updateBus(string num, string stops, string go, string free, string timeout, string day)

{

try

{

string datenow = DateTime.Now.TimeOfDay.ToString();

string nowtime = datenow.Substring(0, 2) + datenow.Substring(3, 2);

string date = DateTime.Today.ToString();

string now = date.Substring(0, 2) + date.Substring(3, 2) + date.Substring(8, 2);

Bus[] buses = Bus.select\_all();

User[] users = User.select\_all();

User[] toDel = new User[users.Length];

int usersToDelCount = 0;

bool truenum = false;

bool forall = false;

if (day == "")

{

forall = true;

}

for (int i = 0; i < buses.Length; i++)

{

if (buses[i].numberOfTrip.ToString() == num && (buses[i].day == day || forall))

{

truenum = true;

Bus work = buses[i];

if (work.intermediatStops == stops.Split(',') && work.goTo == go && work.Places.ToString() == free && work.timeOut == timeToInt(timeout) && (work.day == day || day == ""))

{

User tmp = new User();

tmp.ticketNumber = -2;

return new User[] { tmp };

}

else

{

for (int j = 0; j < users.Length; j++)

{

if (users[j].busNumber.ToString() == num && (users[j].day == day || day == ""))

{

toDel[usersToDelCount++] = users[j];

}

}

delBus(work.numberOfTrip.ToString(), string.Join(",", work.intermediatStops), work.goTo, work.Places.ToString(), timeToString(work.timeOut), work.day);

break;

}

}

}

if (!truenum)

{

User tmp = new User();

tmp.ticketNumber = -1;

return new User[] { tmp };

}

if (forall)

{

Bus newBus = new Bus();

newBus.numberOfTrip = Convert.ToInt32(num);

newBus.intermediatStops = stops.Split(',');

newBus.goTo = go;

newBus.Places = Convert.ToInt32(free);

newBus.timeOut = timeToInt(timeout);

newBus.create();

}

else

{

Bus newBus = new Bus();

newBus.numberOfTrip = Convert.ToInt32(num);

newBus.intermediatStops = stops.Split(',');

newBus.goTo = go;

newBus.Places = Convert.ToInt32(free);

newBus.freePlaces = Convert.ToInt32(free);

newBus.timeOut = timeToInt(timeout);

newBus.day = day;

newBus.create();

}

User[] deleted = new User[usersToDelCount];

int count = 0;

for (int i = 0; i < usersToDelCount; i++)

{

string result = addTicket(toDel[i].ticketNumber.ToString(), toDel[i].firstName, toDel[i].secondName, toDel[i].goTo, toDel[i].busNumber.ToString(), toDel[i].day);

if (result.StartsWith("Билет"))

{

continue;

}

else

{

deleted[count++] = toDel[i];

}

}

User[] toReturn = new User[count];

for (int i = 0; i < count; i++)

{

toReturn[i] = deleted[i];

}

return toReturn;

}

catch (Exception)

{

User tmp = new User();

tmp.ticketNumber = -3;

return new User[] { tmp };

}

}

Додаток С

Метод додавання квитка «addTicket»

static public string addTicket(string numTicket, string firstname, string secondname, string goTo, string numBus, string day) // добавление билета

{

try

{

int y = Convert.ToInt32(day.Substring(4, 2));

bool v = false;

if (y % 4 == 0)

{

v = true;

}

int m = Convert.ToInt32(day.Substring(2, 2));

int d = Convert.ToInt32(day.Substring(0, 2));

bool stop = false;

if (d <= 0 || d > 31)

{

return "Неправильно ввели дату!";

}

switch (m)

{

case 2:

if ((v && d > 29) || (!v && d > 28))

{

stop = true;

}

break;

case 4:

if (d > 30)

{

stop = true;

}

break;

case 6:

if (d > 30)

{

stop = true;

}

break;

case 9:

if (d > 30)

{

stop = true;

}

break;

case 11:

if (d > 30)

{

stop = true;

}

break;

}

if (stop)

{

return "Неправильно ввели дату!";

}

string datenow = DateTime.Now.TimeOfDay.ToString();

string nowtime = datenow.Substring(0, 2) + datenow.Substring(3, 2);

string date = DateTime.Today.ToString();

string now = date.Substring(0, 2) + date.Substring(3, 2) + date.Substring(8, 2);

if (Convert.ToInt32(now.Substring(4,2)) > Convert.ToInt32(day.Substring(4,2)))

{

return "Автобус уже ушёл!";

}

else if (Convert.ToInt32(now.Substring(4, 2)) < Convert.ToInt32(day.Substring(4, 2)))

{

}

else if (Convert.ToInt32(now.Substring(2, 2)) > Convert.ToInt32(day.Substring(2, 2)))

{

return "Вы пытаетесь купить старый билет!";

}

else if (Convert.ToInt32(now.Substring(2, 2)) < Convert.ToInt32(day.Substring(2, 2)))

{

}

else if (Convert.ToInt32(now.Substring(0, 2)) > Convert.ToInt32(day.Substring(0, 2)))

{

return "Попробуйте купить более новый билет!";

}

int[] ticket = Info.select\_all();

int tmpTicket = Convert.ToInt32(numTicket);

if (numTicket == "0")

{

tmpTicket++;

}

for (int i = 1; i < ticket.Length; i++)

{

if (ticket[i] == Convert.ToInt32(numTicket) && numTicket != "0")

{

User user = new User();

user.select(ticket[i].ToString());

return "Внимание! Билет с этим номером уже зарегистрирован!\r\nПокупатель: " + user.secondName;

}

else if (ticket[i] == tmpTicket && numTicket == "0")

{

tmpTicket = (ticket[i] + 1);

}

}

if (numTicket != tmpTicket.ToString())

{

numTicket = tmpTicket.ToString();

}

Bus bus = new Bus();

bus.select(numBus + "-" + day);

if (bus.intermediatStops == null)

{

bus.select(numBus + "-");

}

if (timeToInt(nowtime) >= bus.timeOut && day == now)

{

return "Автобус будет завтра!";

}

if (bus.goTo == null)

{

return "Не правильно указан автобуса!!!";

}

bool stops = false;

for (int i = 0; i < bus.intermediatStops.Length; i++)

{

if (bus.intermediatStops[i] == goTo)

{

stops = true;

break;

}

}

if (stops == false && bus.goTo != goTo)

{

return "Автобус не следует в выбраном направлении";

}

Bus[] buses = Bus.select\_all();

bool need = false;

for (int i = 0; i < buses.Length; i++)

{

if (buses[i].numberOfTrip.ToString() + buses[i].day == numBus + day)

{

if (buses[i].freePlaces == 0)

{

return "На выбраный рейс нет свободных мест!";

}

else

{

Bus update = new Bus();

update.select(buses[i].numberOfTrip.ToString() + "-" + day);

update.freePlaces--;

update.create();

need = true;

}

}

}

if (!need)

{

Bus busnew = new Bus();

busnew.numberOfTrip = bus.numberOfTrip;

busnew.intermediatStops = bus.intermediatStops;

busnew.Places = bus.Places;

busnew.freePlaces = bus.Places - 1;

busnew.goTo = bus.goTo;

busnew.timeOut = bus.timeOut;

busnew.day = day;

busnew.create();

}

User tmp = new User();

tmp.ticketNumber = Convert.ToInt32(numTicket);

tmp.firstName = firstname;

tmp.secondName = secondname;

tmp.goTo = goTo;

tmp.busNumber = Convert.ToInt32(numBus);

tmp.day = day;

tmp.create();

Info.add(tmp.ticketNumber);

return "Билет успешно добавлен!\r\n";

}

catch (Exception)

{

return "Вы ввели некорректные данные!!";

}

}