НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМ. ІГОРЯ СІКОРСЬКОГО”

Кафедра технічної кібернетики

**Курсова Робота**

З дисципліни «Об’єктно-орієнтовне програмування -2»

Купівля квитків УкрЗалізниці через інтернет-банкінг

(Виконано на мові Java)

Керівник Виконавець

Хмелюк В.С ст. Малина А.А.

зал. книжка №ІП-5210

«Допущений до захисту» гр. ІТ-52

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(Особистий підпис керівника) (Особистий підпис виконавця)

« » \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2017р. « » \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2017р.

Захищений з оцінкою

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(Оцінка)

Члени комісії:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(Особистий підпис) (Розшифровка підпису)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(Особистий підпис) (Розшифровка підпису)

Київ – 2017

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ «КПІ ім. Ігоря Сікорського»

(назва вищого закладу)

Кафедра ТЕХНІЧНОЇ КІБЕРНЕТИКИ

Дисципліна «Об’єктно-орієнтовне програмування»

Курс ІІ Група ІТ-52 Семестр 4 .

**ЗАВДАННЯ**

**на курсову роботу студента**

Малини Андрія Андрійовича .

(прізвище, ім’я, по батькові)

1. Тема роботи Розроблення системи замовлення та купівлі залізничних квитків, аналогом якої є система в «Приват24 .

.

1. Строк здачі студентом закінченої роботи 30.05.2017 р.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_ \_\_\_
2. Вихідні дані до роботи: Розробка десктопної програми мовою програмування Java у середовищі розробки InteljIDEA, задачею якої є програма купівлі залізничних квитків з алгоритмом створення залізничних шляхів, та використання СУБД H2.
3. Зміст розрахунково – пояснювальної записки (перелік питань, які підлягають розробці)

*1.\_Вступ. 2.\_Постановка задачі. 3.\_Огляд існуючих рішень та опис моделі.*

*4. Режим роботи та опис діалогу. 5. Опис програми. 6.Інструкція програми.*

*7. Керівництво користувача. 8. Висновки. 9. Список використаних джерел.*

***Додатки*:**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

*10. Текст програми. 11. Результати працездатності програми.*

5. Перелік графічного матеріалу (з точним зазначенням обов’язкових креслень): Діаграми класів, Скрін-шоти (роботи програми).

7.Список класів: Train, City, Reservation, Seat, Stop, Ticket, Controller, PersistenceManager, TicketService, TrainTableRow, AbstractTicketService, AbstractTrainService, DBTicketService, DBTrainService, TicketService, TrainService.

8. Дата видачі завдання 28.02.2017 р.

КАЛЕНДАРНИЙ ПЛАН

|  |  |  |  |
| --- | --- | --- | --- |
| №  п/п | Назва етапів виконання курсової роботи | Строк виконання етапів роботи | Підписи або примітки |
| 1. | Отримання та узгодження теми курсової роботи | 28.02.2017 |  |
| 2. | Огляд існуючих рішень з тематики роботи | 15.03.2017 |  |
| 3. | Розробка моделі та структури програми | 28.03.2017 |  |
| 4. | Вибір алгоритмів розв’язання задачі | 10.04.2017 |  |
| 5. | Розробка діалогового інтерфейсу програми | 18.04.2017 |  |
| 6. | Кодування програми | 24.04.2017 |  |
| 7. | Налагодження та перевірка програми | 05.05.2017 |  |
| 8. | Оформлення пояснювальної записки | 12.05.2017 |  |
| 9. | Захист курсової роботи | 25.05.2017 |  |

Студент \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_Малина А.А\_\_\_\_\_

(Особистий підпис) (Прізвище, ім’я, по батькові)

Керівник \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_Хмелюк В.С\_\_\_\_\_

(Особистий підпис) (Прізвище, ім’я, по батькові)

«30» травня 2017 р.

АНОТАЦІЯ

У цьому проекті буде розроблено систему купівлі та бронювання залізничних квитків «Ticket Service». У роботі зображено теоретичний огляд мотодів розв’язання даної задачі, обґрунтується актуальність теми, розкриється найменування і призначення області застосування програми, що розробляється в курсовій роботі. Будуть розглянуті нові рішення й методи розв'язан­ня задачі, обґрунтується вибір конкретного методу та сформується мате­матична модель й загальна структура програми. У якості допоміжних алгоритмів було обрано алгоритм Дейкстри. Також будуть наведені інструкції до програми, вказівки програміста, висновки та продемонстровано дружній інтерфейс програми.

Алгоритми та інтерфейс програми реалізовані на мові програмування Java в середовищі JetBrains Intellij IDEA.

Дана пояснювальна записка містить 54 сторіноки, а 4 таблиці, а також 4 скріншоти роботи програми та її використання і діаграму класів.

ЗМІСТ

[ВСТУП 9](#_Toc484468128)

[1 ПОСТАНОВКА ЗАДАЧІ 10](#_Toc484468129)

[2 РОЗГЛЯД ІСНУЮЧИХ РІШЕНЬ 11](#_Toc484468130)

[3 МАТЕМАТИЧНА МОДЕЛЬ ТА СТРУКТУРА ПРОГРАМИ 12](#_Toc484468131)

[4 ПЕРЕЛІК І ПРИЗНАЧЕННЯ РЕЖИМІВ ТА СТРУКТУРА ДІАЛОГУ 15](#_Toc484468132)

[5 СТРУКТУРА ДАНИХ ТА РЕСУРСІВ ПРОГРАМИ 16](#_Toc484468133)

[6 ОПИС ПРОГРАМИ 21](#_Toc484468134)

[7 ІНСТРУКЦІЯ АДМІНІСТРАТОРА 23](#_Toc484468135)

[8 КЕРІВНИЦТВО КОРИСТУВАЧА 25](#_Toc484468136)

[ВИСНОВКИ 26](#_Toc484468137)

[ВИКОРИСТАНІ ДЖЕРЕЛА 27](#_Toc484468138)

[1.H2–база даних. – Режим доступу: 27](#_Toc484468139)

[2.UMLдіаграми . – Режим доступу: 27](#_Toc484468140)

[3.JavaFx.–Режимдоступу: 27](#_Toc484468141)

[ДОДАТОК А 28](#_Toc484468142)

[ДОДАТОК Б 55](#_Toc484468143)

## ВСТУП

Мільйони людей знаходяться в подорожі щодня, використовуючи різний вид транспорту для поліпшення умов та швидкості переміщення. Одним з таких видів транспорту є потяг. Близько 15 000 потягів та електричок курсують українською залізницею щодня, що робить цей вид транспорту другим по популярності на теренах України. Використання простого програмного забезпечення полегшить роботу працівників та клієнтів даного виду транспорту, підвищить якість обслуговування клієнтів, дозволить в чотири кліки знайти потрібний потяг та оформити купівлю квитка на даний потяг.

Виходячи з всього цього можна сказати – вибрана тема роботи є актуально.

## 1 ПОСТАНОВКА ЗАДАЧІ

Ціллю цієї курсової роботи розробка СКБК «Ticket Service» для пошуку, купівлі і бронювання квитка на потяг, що підходить за заданим маршрутом, датою і обраним з результататів пошуку. Програма повинна забезпечувати виконання таких функцій:

* пошук потягів за вказаним маршрутом і датою;
* вивід списку потягів що підходять з детальною інформацією;
* бронювання квитка за заданим ім’ям і прізвищем;
* мати інтуїтивно зрозумілий інтерфейс;

Виходячи з цілі потрібно вирішити такі проблеми:

* розробити алгоритм додавання потягів, зупинок, вагоні та місць в них;
* розробити алгоритм прокладання маршруту потягів;
* заповнення часових інтервалів подорожі для кожного потягу;
* розробити алгоритм додавання заброньованих;
* виключити можливість бронювання заброньованого квитка;
* налаштувати взаємодію з базою даних [Н2[1];](#_1._MySQL_–)
* створити [UML[2]](#_2.UML_діаграми_.)- діаграми;
* закодувати програму;
* провести тестування;

# 2 РОЗГЛЯД ІСНУЮЧИХ РІШЕНЬ

Динамічними варіантами рішення даної задачі є лише варіанти стартового генерування інформації потягів. А саме:

* час зупинки;
* дистанції;
* маршрут;

В даному випадку можливе використання алгоритму Дейкстри, або алгоритм Флойда-Уоршала. В роботі мною був обраний алгоритм Дейкстри, тому що він є менш ресурсозатратним, і видає саме ті дані які будуть потрібні для подальшого розв’язання поставленої задачі. А алгоритм Фдойда-Уоршала - напроти працює довше, використовує більше пам’яті і видає більше даних ніж нам потрібно в поставленій задачі. Але у висновку жодного з цих алгоритмів користувач чи адміністратор не побачать, так як вони будуть видалені через непотрібність для подальшого вирішення даної задачі.

Для реалізації додавання заброньованих квитків, було прийнято рішення використовувати клас-обгортку в який заповнювались всі дані, а в подальшому об’єкт даного класу було додано через транзакцію в базу-даних.

Релізацію заборони повторного бронювання заданого місця, було виконано за допомогою транзакції і подальшого скачування таблиці заброньованих квитків в масив типу класа-обгортки з усіми необхідними полями для перевірки. Даний варіант є кращим, ніж виконання даного пункту через постійні транзакції, щоб призвело для значного сповільненя роботи програми. Обраний варіант дозволяє кешувати необхідні нам дані для подальшого використання. Далі ми перевіряємо всі дані введених користувачем з даними масива який є копією бази-даних таблиці резервації. За умови позитивного проходу перевірки введені дані будуть додані до бази-даних. Однак, якщо результатом є негативний прохід даної перевірки, користувачеві буде зображено «вікно помилку» з проханням замінити введені дані, та запропонованою можливістю повторного оформлення квитка вже з новими даними.

Вивід результатів пошуку за задиними даними було прийнято оформити в вигляді таблиці. Для цього необхідно використати масив типу класа-обгортки за допомогою транзакції завантажити в цей масив список усіх потягів, далі провести фільтрацію отриманого списку відповідно, до заданих даних. Отримані дані також записати в проміжний масив-обгортку для виведення отрманих результатів в графічний інтерфейс користувача а саме в таблицю. Дане рішеня є ліпшим за використання постійних фільтрацій через транзакції, так як являється швидшим та менш затратним по пам’яті. Дані проміжкові результативні масиви будуть стерті за допомогою Garbage Collector.

Проблема інтуїтивно зрозумілого інтерфусу роз’вязана за допомогою ітернет-банкінгу Приват24, а саме прегляду будови їх ергономічного інтерфейсу покупки білетів на потяги. Далі ми намагаємося відтворити їх інтерфейс з реалізацією виправляння їх ергономічних помилок. Дане рішення є найкращим, через отриманий статистичний збір оцінок ергономічно зручних сервісів, для купівлі та бронювання квитків на необхідні потяги.

## 

## 3 МАТЕМАТИЧНА МОДЕЛЬ ТА СТРУКТУРА ПРОГРАМИ

Для вирішення проблеми програма повинна мати початкове вікно: з двома кнопками, шістьма полями для вводу, одним полем для вводу календарних даних та таблицею для відображення списків. Одна з кнопок є вимкненою до натискання іншої. Кожна з них повинна містити інформацію відповідно до тематики, бути інтуїтивно зрозумілою для користувача, та ергономічно зручною. Поля для введення повинні мати лейбли та фонові написи для чіткого розуміння зазначеного поля, до них входить і календар. Таблиця повинна відображати основну необхідну інформацію для користувача. Для створення цих елементів будуть використовуватися [JavaFx[3].](#_3._Windows_Forms.–) Її використання можна обґрунтувати тим, що форма є зручною для реалізації поставленої задачі, а також основною частиною програми , тому потрібно приділити особливу увагу її зовнішньому вигляду і функціям. JavaFx являють собою чистий аркуш, який розробник оснащує елементами управління, формуючи призначений для користувача інтерфейс, і кодом для роботи з даними.

Існує безліч різних систем управління базами даних (СУБД).В цьому проекті в якості бази даних було вирішено обрати Н2. Причинами цього є доступність на всіх популярних операційних системах, а також простота налаштування і адміністрування, а також малий розмір і висока швидкість робота з нею. Також на сайті розробника є повна [документація[4]](#MySQLdocs) .

Для виведення інформації на форму буде використовуватись елемент TableView. Він надає потужний і гнучкий спосіб відображення даних в табличному форматі. TableView можна використовувати для відображення даних невеликого обсягу не тільки для читання, а й для співпраці з ними та можливістю вибору пунктів таблиці та іншого, а також можна масштабувати її для відображення редагованих дуже великих наборів даних. Для того щоб реалізувати визначену поведінку в додатках, його можна розширити декількома способами. Наприклад, можна програмно задати власні алгоритми сортування, можна створити власні типи комірок. Зовнішній вигляд елемента управління TableView легко налаштовується встановленням значень декількох властивостей. Як джерело даних можуть використовуватися сховища даних різних типів, також він може працювати без прив'язаного до нього джерела даних.Для зв’язку з базою даних буде використано технологія [програмування](https://uk.wikipedia.org/wiki/%D0%9F%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F), яка зв'язує [бази даних](https://uk.wikipedia.org/wiki/%D0%91%D0%B0%D0%B7%D0%B0_%D0%B4%D0%B0%D0%BD%D0%B8%D1%85) з концепціями [об'єктно-орієнтованих](https://uk.wikipedia.org/wiki/%D0%9E%D0%B1%27%D1%94%D0%BA%D1%82%D0%BD%D0%BE-%D0%BE%D1%80%D1%96%D1%94%D0%BD%D1%82%D0%BE%D0%B2%D0%B0%D0%BD%D0%B0_%D0%BC%D0%BE%D0%B2%D0%B0) мов програмування, створюючи «віртуальну об'єктну базу даних» [ORM[5]](#DAO). А саме для мови програмування Java це є Hibernate[6]. Hibernate - засіб відображення між [об'єктами](https://uk.wikipedia.org/wiki/%D0%9E%D0%B1%27%D1%94%D0%BA%D1%82_%28%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F%29) та [реляційними структурами](https://uk.wikipedia.org/wiki/%D0%A0%D0%B5%D0%BB%D1%8F%D1%86%D1%96%D0%B9%D0%BD%D0%B0_%D0%B1%D0%B0%D0%B7%D0%B0_%D0%B4%D0%B0%D0%BD%D0%B8%D1%85) (object-relational mapping, [ORM](https://uk.wikipedia.org/wiki/ORM)) для платформи [Java](https://uk.wikipedia.org/wiki/Java). Hibernate надає легкий для використання каркас ([фреймворк](https://uk.wikipedia.org/wiki/%D0%A4%D1%80%D0%B5%D0%B9%D0%BC%D0%B2%D0%BE%D1%80%D0%BA)) для відображення між об'єктно-орієнтованою моделлю даних і традиційною [реляційною базою даних](https://uk.wikipedia.org/wiki/%D0%A0%D0%B5%D0%BB%D1%8F%D1%86%D1%96%D0%B9%D0%BD%D0%B0_%D0%B1%D0%B0%D0%B7%D0%B0_%D0%B4%D0%B0%D0%BD%D0%B8%D1%85).

3.1 Діаграма класів

Діаграма класів визначає набір статичних, декларативних елементів моделі, інтерфейс і відносини. Інформація, що міститься в діаграмах класів, відображається в вихідний код програми у вигляді функцій членів і атрибутів класу, зазначених у діаграмі.

# 

# 4 ПЕРЕЛІК І ПРИЗНАЧЕННЯ РЕЖИМІВ ТА СТРУКТУРА ДІАЛОГУ

Програма має тільки один режим роботи, а все управління здійснюється за допомогою елементів керування.

Проектування інтерфейсу дуже важливе, він повинен бути простим і зрозумілим для користувача, забезпечувати швидкий доступ до всіх інструментів і містити інформацію про поточний стан програми.

Головне вікно програми буде містити напис – назву програми, а також дві кнопки «Знайти», «Оформити» при натисканні на які можна буде здійснювати дії з базою даних. Також ця форма має поля вводу «Звідки», «Куди», «Дата», «Ім’я», «Прізвище», «Вагон», «Місце», таблицю для виведення даних, і малюнок зображення вагону. Структура діалогу - для початку користувачу потрібно ввеси початкові дані, а саме:

* місто початку маршруту;
* місто кінця маршруту;
* дата подорожі;

Далі після заповнення натиснути конопку пошуку. Після цього ми переходимо до другого етапу діалогу – «Оформлення квитка». Ми повинні обрати підходящий для нас потяг заповнити поля:

* ім’я;
* прізвище;
* вагон;
* місце;

Після всіх цих маніпуляцій, квиток додано до бази даних і робота завершена.

Форматом виведених даних ми маємо лише список потягів що відповідають запиту пошуку в формі таблиці з якою ми в подальшому співпрацюємо.

## 5 СТРУКТУРА ДАНИХ ТА РЕСУРСІВ ПРОГРАМИ

В програмі використовуються такі пакети: javax, javafx, org, java

|  |
| --- |
|  |
|  |

|  |
| --- |
|  |

Пакети org і javax використовується для роботи з базою даних H2. Розглянемо використані класи з цьих пакетів[[6]:](#MySQLnameSpace)

Таблиця 5.1– Використані класи з пакету org і javax

|  |  |
| --- | --- |
| Клас | Використання |
| persistence.EntityManager | Надає відкрите підключення до бази даних сервера MySQL. Цей клас не може успадковуватися. Використовується при підключенні до сервера бази даних . При створені приймає строку підключення в яку входить : назва сервера, логін користувача , пароль , назва бази . |
| hibernate.annotations | Анотації для розмітки ORM для бази даних. Таких як:   * назва таблиці; * назва колонки; * генерація значень для поля id; * показу відношень таблиць; |

Розглянемо детальніше класи які містить програма :

Таблиця 5.2 – Перелік і опис класів

|  |  |
| --- | --- |
| Назва | Опис |
| Train | Клас який має п’ять private полів, має дефолтний конструктор. Поля: int id, String name, String type, List<Seat> seats, List<Stop> stops. Клас використовується для заповнення таблиць бази даних, та даних обробки. |

Продовження таблиці 5.2

|  |  |  |
| --- | --- | --- |
| Ticket | | Клас аналогічний класу Train (має ті самі функції). Має вісім полів: Integer id, Seat seat,int start, int end, double price, String name, String surname, int distance.  Для запису квитків. |
| Stop | | Клас аналогічний класу Train (має ті самі функції). Має чотири полів: int id, int city, Date date, Train train.  Для запису зупинок потягів. |
| Seat | | Клас аналогічний класу Train (має ті самі функції). Має п’ять полів: int id, Train train, int carriage, int number,  List<Reservation> reservations .  Для запису місць потягу. |
| Reservation | | Клас аналогічний класу Train (має ті самі функції). Має п’ять полів: int id, Seat seat, int start, int end.  Для резервування квитків. |
| City | Клас аналогічний класу Train (має ті самі функції). Має два поля: int id, int city  Для перевірки міст. | |
| **Controller** | Клас який описує весь графічний інтерфейс і роботу з ним. В ньому знаходяться методи які працюють через події на клавішах які надалі запускають процес пошуку поїздів, та процес оформлення покупця. Також завдяки цьому класу ми зчитуємо данні з GUI за допомогою його прив’язки до xml самого проекту. Також в цьому класі знаходиться метод перебудови стрічки дати для коректного співвідношення з стрічкою з бази даних . | |

Продовження таблиці 5.2

|  |  |
| --- | --- |
| Ticket\_Service | Клас який символізує початок роботи програми в ньому знаходяться методи main i start що реалізують старт GUI та показу всієї необхідної інформації для роботи з інтерфейсом. |
| PersistenceManager | Клас використовується для підключення, зчитування і записів полів в БД, за допомогою EntityManagerFactory. |
| TrainTableRow | Клас має вісім приватних полів Integer id; String name;  String type; String route; String departureTime; String arrivalTime; Integer numberOfCars; List<Stop> stopList;. Також має два конструктори. Клас використовується для відображення необхідних потягів в таблиці а також виділення необхідного потягу зчитування його необхідних параметрів та передачу цих параметрів в оформлення даного квитка. |
| AbstractTrainService | Реалізує інтерфейс TrainService. А сам являється абстрактним класом для роботи з отриманими даними головною задачею класу є пошук необхідних потягів та пошук і зрівняння міст даних потягів. |
| DBTrainService | Наслідує AbstractTrainService. Реалізує запити в базу даних та витяг всіх потягів що є в наявності для перевірки на відповідність до запиту користувача. |
| TrainService | Інтерфейс для класу AbstractTrainService, з інтерфейсами його методів. |
| TicketService | Інтерфейс для класу AbstractTicketService, з інтерфейсами його методів. |

Продовження таблиці 5.2

|  |  |
| --- | --- |
| AbstractTicketService | Абстрактний клас який реалізує інтерфейс TicketService.  А також обрахунку ціни квитка та дистанції подорожі. |
| DBTicketService | Клас для роботи з базою даних який унаслідується від AbstractTicketService. Його функція обробка отриманих результатів та запису даних до БД в представленні квитка. |

5.3 Структура бази даних

Для виконання роботи була створена база даних TService. Вона містить 6 таблиць:City, Reservation, Ticket, Train, Stop, Seat.Розглянемо деякі з них :

Таблиця 5.3 – Структура бази даних

|  |  |
| --- | --- |
| Назва таблиці | Опис |
| City | Зберігає інформацію про всі міста. Містить такі стовпці :id INT PRIMARY KEY AUTO\_INCREMENT, city\_name VARCHAR(60) |
| Ticket | Використовується для зберіганняпідготованих квитків. Складається з:  id INT NOT NULL, start INT NOT NULL, end ,price, seat, name, surname, distance.  Ключі : PRIMARY KEY (id) |
| Train | Зберігає в собі всі генеровані завчасно потяги. Стовпці: id INT AUTO\_INCREMENT, name VARCHAR(30),  type VARCHAR(200),stop, seats. |
| Stop | Зберігає в собі генеровані зупинки кожного потягу. Стовпці id INT AUTO\_INCREMENT, int city, Date date, Train train |

Продовження таблиці 5.3

|  |  |
| --- | --- |
| Seat | Зберігає кількість вагонів і місць в кожному вагоні для кожного потягу. Стовпці id INT AUTO\_INCREMENT, Train train, int carriage, int number, seats |
| Reservation | Зберігає зайняті місця та звідки і до куди вони зайняті. id INT AUTO\_INCREMENT, Seat seat, int start, int end |

## 6 ОПИС ПРОГРАМИ

Опис роботи програми: після запуску з’являється головне вікно програми на якому є дві кнопки «Знайти» та «Оформити». «Оформити» - є недоступною доти, доки не натиснуть кнопку «Знайти». Кожна з них реалізує відповідні задачі. На цій формі можна знайти маршрут, та оформити квиток на нього.

Перед натисканням кнопки «Знайти» повинні бути заповнені поля:

* місто початку маршруту;
* місто кінця маршруту;
* дата подорожі;

Для коректного пошуку та виконання алгоритмів відображення.

Подія buyEvent викликає оформлення квитка та передачу необхідних даних до бази даних. Також в ньому реалізуються перевірки та виклик функцій для перевірки зайнятості.

Подія searchEvent викликає пошук у базі даних. А також заміну ергономіки клавіш для поліпшеного розуміння подальших дій. Відображення отриманого результату та переводить користувача на другий етап діалогу з програмою.

Подія dateEvent викликає заміну даних дати на ту, яка підходить для бази даних, так як в базі даних дата відрізняється від введеної.

Метод showTable() відображає генеровану таблицю натиском клавіші «Знайти». А саме зображає відповідні потяги та всю необхідну інформацію для користувача.

Метод buyTicket() використовується для закриття з’єднання з базою даних, оформлення квитка та перевірки зайнятості обраного місця.

Далі ці методи передають данні до необхідних класів для продовження роботи та відображення необхідних результатів

Метод reservation(start, end, carriageText, placeText, id) використовується ддля перевірки можливості бронювання місця, а саме чи не зайняте це місце іншою людиною на цьому проміжку дистанції потягу. Цей метод також аналізує проміжки зайнятості та може дати можливість бронювання місця, якщо до чи після вас його хтось також забронював, що відтворює реальний алгоритм бронювання кивтків на потяг.

Робота програми відображається на формі з якою працює користувач в даний момент.

Запуск програми відбувається при вводі в термінал команди «java –jar TicketService.jar» в папці де знаходиться даний файл.

## 7 ІНСТРУКЦІЯ АДМІНІСТРАТОРА

Назва програми «Ticket Service». ЇЇ призначення спрощення купівлі та бронювання квитків на потяг як користувачеві покупцю, так і користувачеві співробітнику УкрЗалізниці. Вона написана на мові програмування Java в середовищі розробки IntelliJ IDEA 2017.2.2(64) . Периферійні пристрої використані під час проектування: миша , тачпад, клавіатура, монітор.

Перед початком кодування програми потрібно завантажити середовище розробки з сайту <https://www.jetbrains.com/>. Встановлення IntelliJ IDEA повинне виконуватися від імені облікового запису адміністратора . Однак , після встановлення IntelliJ IDEA ці облікові дані не потрібні [[7]](#setupVS).Якщо на комп'ютері встановлена ​​ попередня версія IntelliJ IDEA, видальте її. Вимоги середовища розробки до обладнання можна переглянути на сторінці по посиланню[[8]](#requirementsVS) .

Після встановлення потрібно налаштувати IntelliJ IDEA. Можна налаштувати кольори , шрифти , меню , панелі інструментів , розташування вікон і поєднання клавіш , а також створювати шаблони , використовувати зовнішні засоби і управляти розширеннями. Будь-які налаштування IntelliJ IDEA можна зробити доступними для інших користувачів або скинути до значень за замовчуванням .

Якщо закрити і повторно відкрити IntelliJ IDEA , задані параметри збережуться , оскільки вони автоматично зберігаються в окремому файлі[[9]](#Properties) .

Далі створюємо програму JavaApllipation і розміщуємо на формі всі потрібні елементи . Після додавання кнопок , потрібно призначити їм відповідну картинку з ресурсів, які підібрані заздалегідь:

Перейшовши до коду, потрібно підключити додаткові пакети.

Рекомендовані вимоги для стабільної роботи програми :

* Процесор з частотою 1,8 ГГц або більш потужний
* 8гігабайт ( ГБ ) оперативної пам'яті
* 24 Мб доступного простору на жорсткому диску
* Жорсткий диск з частотою обертання 5400 об . / Хв
* Відеоадаптер з підтримкою DirectX 9 і розширенням екрану не менше 1024 x 768

Для запуску програми потрібно скопіювати собі на комп’ютер папку «Ticket Service» і запустити файл jar file.

## 8 КЕРІВНИЦТВО КОРИСТУВАЧА

Назва програми «Ticket Service». Вона слугує для купівлі і бронювання квитків на потяг.

Для роботи програми потрібна будь-яка операційна система. Вона не потребує встановлення додаткових драйверів чи бібліотек. Програма не потребує великих ресурсів від ПК : кількість оперативної пам’яті 8 ГБ і вище (Windows 7) і місце на жорсткому диску 25 мб.

Для запуску програми потрібно скопіювати собі на комп’ютер папку «Ticket Service» і запустити файл jar file.

Основні можливості програми та їх використання:

* заповнюємо поля звідки, куди, дата;
* натискаємо на кнопку пошуку;
* отримуємо таблицю з відповідними потягами які підходять до заданих параметрів;
* вибираємо натиском ЛКМ необхідного нам потягу;
* заповнюємо поля: ім’я , прізвище, вагон і номер місця;
* натискаємо кнопку оформити;

Можливі виключення:

* обирання зайнятого місця. В даному випадку натиснути на новому вікні кнопку ок, та потім обрати нове місце чи вагон, і знову натиснути кнопку оформлення;
* вибір місця чи вагону яких не існує. Вирішення відповідне як і в минулому пункті;
* неправильне введення міст. Вирішення: повторити введення міста українською мовою та з великої літри.
* Після виконня всіх пунктів для пошуку, таблиця пуста. Вирішення: змінити дату пошуку на іншу.

## ВИСНОВКИ

В результаті виконання роботи було повторено багато можливостей мови програмування Java. Розробка повністю ООП програми яка поліпшить знання і навички використання даної парадигми програмування, а саме наслідування інкапсуляція і поліморфізм. Покаращення навичок роботи з базою даних Н2. Отримання навичок при використання framework Hibernate та Maven. Отримання навичок роботи з пакетами JavaFx та опанування середи розробки Scene builder для JavaFx. Створено функціональну систему яку можна використовувати на будь-якому комп’тері незалежно від операційної системи, так як програми написані на мові програмування Java являються кросплатформеними. Інтерфейс програми є зручним та був сконструйований ергономічно вірно для інтуїтивного розуміння користувачем програми.Також він надає швидкий доступ до всіх інструментів. Проведене тестування шляхом вводу тестових данних показало , що програма коректно виконує свою роботу. Також було проведено код-ревю, що дозволило поліпшити код.Ці всі пункти дозволяють зробити висновок, що поставлена ціль зроблена якісно та може бути застосована в комерційних цілях.

## ВИКОРИСТАНІ ДЖЕРЕЛА

# 1.H2–база даних. – Режим доступу:

[https://www.h2.com/](https://www.h2.com/ )

# 2.UMLдіаграми . – Режим доступу:

<https://ru.wikipedia.org/wiki/UML>

# 3.JavaFx.–Режимдоступу:

<https://ru.wikipedia.org/wiki/JavaFx>

4.Документація H2. – Режим доступу:

<https://www.h2.com/doc/>

5.ORM. – Режим доступу:

<https://ru.wikipedia.org/wiki/ORM>

6.Package– простір імен.– Режим доступу:

<https://oracle.com/java/package/>

7.Встановлення Intellj IDEA. – Режим доступу:

Jetbrains.com/Intellj IDEA/download

8.Системні вимоги для Intellj IDEA. – Режим доступу:

Jetbrains.com/Intellj IDEA

9.Налаштування параметрів разработки в VisualStudio. – Режим доступу:

Jetbrains.com/Intellj IDEA/lib/

10. Герберт Шилд. «Java Руководство для профисионалов». Спільна частина: -ученик Издательский дом «Вильямс»2015

11.Брюс Еккель. «Философия Java» Спільна частина: -ученик Издательский дом «Норма»2012

12. Герберт Шилд. «Патерны проектирования на Java». Спільна частина: -ученик Издательский дом «Вильямс»2013

## ДОДАТОК А

Код програми

|  |
| --- |
| package models; |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| import javax.persistence.\*; |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| @Entity |
|  |

|  |
| --- |
| @Table(name = "city") |
|  |

|  |
| --- |
| public class City { |
|  |

|  |
| --- |
| @Id |
|  |

|  |
| --- |
| @GeneratedValue |
|  |

|  |
| --- |
| @Column(name = "id\_city") |
|  |

|  |
| --- |
| private Integer id; |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| @Column(name = "name\_city") |
|  |

|  |
| --- |
| private String cityName; |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| public Integer getId() { |
|  |

|  |
| --- |
| return id; |
|  |

|  |
| --- |
| } |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| public void setId(Integer id) { |
|  |

|  |
| --- |
| this.id = id; |
|  |

|  |
| --- |
| } |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| public String getCityName() { |
|  |

|  |
| --- |
| return cityName; |
|  |

|  |
| --- |
| } |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| public void setCityName(String cityName) { |
|  |

|  |
| --- |
| this.cityName = cityName; |
|  |

|  |
| --- |
| } |
|  |

}

|  |
| --- |
| package models; |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| import javax.persistence.\*; |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| @Entity |
|  |

|  |
| --- |
| @Table(name = "reservations") |
|  |

|  |
| --- |
| public class Reservation { |
|  |

|  |
| --- |
| @Id |
|  |

|  |
| --- |
| @GeneratedValue |
|  |

|  |
| --- |
| @Column(name = "id\_reservation") |
|  |

|  |
| --- |
| private Integer id; |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| @ManyToOne |
|  |

|  |
| --- |
| @JoinColumn(name = "id\_seat") |
|  |

|  |
| --- |
| private Seat seat; |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| private int start; |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| private int end; |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| public Integer getId() { |
|  |

|  |
| --- |
| return id; |
|  |

|  |
| --- |
| } |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| public void setId(Integer id) { |
|  |

|  |
| --- |
| this.id = id; |
|  |

|  |
| --- |
| } |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| public int getStart() { |
|  |

|  |
| --- |
| return start; |
|  |

|  |
| --- |
| } |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| public void setStart(int start) { |
|  |

|  |
| --- |
| this.start = start; |
|  |

|  |
| --- |
| } |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| public int getEnd() { |
|  |

|  |
| --- |
| return end; |
|  |

|  |
| --- |
| } |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| public void setEnd(int end) { |
|  |

|  |
| --- |
| this.end = end; |
|  |

|  |
| --- |
| } |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| public Seat getSeat() { |
|  |

|  |
| --- |
| return seat; |
|  |

|  |
| --- |
| } |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| public void setSeat(Seat seat) { |
|  |

|  |
| --- |
| this.seat = seat; |
|  |

|  |
| --- |
| } |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| @Override |
|  |

|  |
| --- |
| public String toString() { |
|  |

|  |
| --- |
| return "Reservation{" + |
|  |

|  |
| --- |
| "id=" + id + |
|  |

|  |
| --- |
| ", seat=" + seat.getTrain().getName() + "/" + seat.getCarriage() + "/" + seat.getNumber() + |
|  |

|  |
| --- |
| ", start=" + start + |
|  |

|  |
| --- |
| ", end=" + end + |
|  |

|  |
| --- |
| '}'; |
|  |

|  |
| --- |
| } |
|  |

}

|  |
| --- |
| package models; |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| import org.hibernate.annotations.\*; |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| import javax.persistence.\*; |
|  |

|  |
| --- |
| import javax.persistence.CascadeType; |
|  |

|  |
| --- |
| import javax.persistence.Entity; |
|  |

|  |
| --- |
| import javax.persistence.Table; |
|  |

|  |
| --- |
| import java.util.ArrayList; |
|  |

|  |
| --- |
| import java.util.List; |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| @Entity |
|  |

|  |
| --- |
| @Table(name = "seats") |
|  |

|  |
| --- |
| public class Seat { |
|  |

|  |
| --- |
| @Id |
|  |

|  |
| --- |
| @GeneratedValue |
|  |

|  |
| --- |
| @Column(name = "id\_seat") |
|  |

|  |
| --- |
| private Integer id; |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| @ManyToOne |
|  |

|  |
| --- |
| @JoinColumn(name = "id\_train") |
|  |

|  |
| --- |
| private Train train; |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| private int carriage; |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| private int number; |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| @OneToMany(mappedBy = "seat", cascade = CascadeType.ALL, orphanRemoval = true) |
|  |

|  |
| --- |
| private List<Reservation> reservations = new ArrayList<>(); |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| public void addReservation(Reservation r) { |
|  |

|  |
| --- |
| reservations.add(r); |
|  |

|  |
| --- |
| r.setSeat(this); |
|  |

|  |
| --- |
| } |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| public Integer getId() { |
|  |

|  |
| --- |
| return id; |
|  |

|  |
| --- |
| } |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| public void setId(Integer id) { |
|  |

|  |
| --- |
| this.id = id; |
|  |

|  |
| --- |
| } |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| public Train getTrain() { |
|  |

|  |
| --- |
| return train; |
|  |

|  |
| --- |
| } |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| public void setTrain(Train train) { |
|  |

|  |
| --- |
| this.train = train; |
|  |

|  |
| --- |
| } |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| public int getCarriage() { |
|  |

|  |
| --- |
| return carriage; |
|  |

|  |
| --- |
| } |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| public void setCarriage(int carriage) { |
|  |

|  |
| --- |
| this.carriage = carriage; |
|  |

|  |
| --- |
| } |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| public int getNumber() { |
|  |

|  |
| --- |
| return number; |
|  |

|  |
| --- |
| } |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| public void setNumber(int number) { |
|  |

|  |
| --- |
| this.number = number; |
|  |

|  |
| --- |
| } |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| public List<Reservation> getReservations() { |
|  |

|  |
| --- |
| return reservations; |
|  |

|  |
| --- |
| } |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| public void setReservations(List<Reservation> reservations) { |
|  |

|  |
| --- |
| this.reservations = reservations; |
|  |

|  |
| --- |
| } |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| @Override |
|  |

|  |
| --- |
| public String toString() { |
|  |

|  |
| --- |
| return "Seat{" + |
|  |

|  |
| --- |
| "id=" + id + |
|  |

|  |
| --- |
| ", train=" + train.getName() + |
|  |

|  |
| --- |
| ", carriage=" + carriage + |
|  |

|  |
| --- |
| ", number=" + number + |
|  |

|  |
| --- |
| ", reservations=" + reservations + |
|  |

|  |
| --- |
| '}'; |
|  |

|  |
| --- |
| } |
|  |

}

|  |
| --- |
| package models; |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| import java.util.Date; |
|  |

|  |
| --- |
| import javax.persistence.\*; |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| @Entity |
|  |

|  |
| --- |
| @Table(name = "stops") |
|  |

|  |
| --- |
| public class Stop { |
|  |

|  |
| --- |
| @Id |
|  |

|  |
| --- |
| @GeneratedValue |
|  |

|  |
| --- |
| @Column(name = "id\_stop") |
|  |

|  |
| --- |
| private Integer id; |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| @Column(name = "city") |
|  |

|  |
| --- |
| private int city; |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| @Column(name = "date") |
|  |

|  |
| --- |
| private Date date; |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| @ManyToOne |
|  |

|  |
| --- |
| @JoinColumn(name = "id\_train") |
|  |

|  |
| --- |
| private Train train; |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| public int getCity() { |
|  |

|  |
| --- |
| return city; |
|  |

|  |
| --- |
| } |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| public void setCity(int city) { |
|  |

|  |
| --- |
| this.city = city; |
|  |

|  |
| --- |
| } |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| public Date getDate() { |
|  |

|  |
| --- |
| return date; |
|  |

|  |
| --- |
| } |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| public void setDate(Date date) { |
|  |

|  |
| --- |
| this.date = date; |
|  |

|  |
| --- |
| } |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| public Integer getId() { |
|  |

|  |
| --- |
| return id; |
|  |

|  |
| --- |
| } |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| public void setId(Integer id) { |
|  |

|  |
| --- |
| this.id = id; |
|  |

|  |
| --- |
| } |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| public Train getTrain() { |
|  |

|  |
| --- |
| return train; |
|  |

|  |
| --- |
| } |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| public void setTrain(Train train) { |
|  |

|  |
| --- |
| this.train = train; |
|  |

|  |
| --- |
| } |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| @Override |
|  |

|  |
| --- |
| public String toString() { |
|  |

|  |
| --- |
| return "Stop{" + |
|  |

|  |
| --- |
| "id=" + id + |
|  |

|  |
| --- |
| ", city=" + city + |
|  |

|  |
| --- |
| ", date=" + date + |
|  |

|  |
| --- |
| ", train=" + train.getName() + |
|  |

|  |
| --- |
| '}'; |
|  |

|  |
| --- |
| } |
|  |

}

|  |
| --- |
| package models; |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| import javax.persistence.\*; |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| @Entity |
|  |

|  |
| --- |
| @Table(name = "tickets") |
|  |

|  |
| --- |
| public class Ticket { |
|  |

|  |
| --- |
| @Id |
|  |

|  |
| --- |
| @GeneratedValue |
|  |

|  |
| --- |
| @Column(name = "id\_ticket") |
|  |

|  |
| --- |
| private Integer id; |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| @ManyToOne |
|  |

|  |
| --- |
| @JoinColumn(name = "id\_train") |
|  |

|  |
| --- |
| private Seat seat; |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| private int start; |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| private int end; |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| private double price; |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| private String name; |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| private String surname; |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| private int distance; |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| public Seat getSeat() { |
|  |

|  |
| --- |
| return seat; |
|  |

|  |
| --- |
| } |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| public void setSeat(Seat seat) { |
|  |

|  |
| --- |
| this.seat = seat; |
|  |

|  |
| --- |
| } |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| public int getStart() { |
|  |

|  |
| --- |
| return start; |
|  |

|  |
| --- |
| } |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| public void setStart(int start) { |
|  |

|  |
| --- |
| this.start = start; |
|  |

|  |
| --- |
| } |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| public int getEnd() { |
|  |

|  |
| --- |
| return end; |
|  |

|  |
| --- |
| } |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| public void setEnd(int end) { |
|  |

|  |
| --- |
| this.end = end; |
|  |

|  |
| --- |
| } |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| public double getPrice() { |
|  |

|  |
| --- |
| return price; |
|  |

|  |
| --- |
| } |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| public void setPrice(double price) { |
|  |

|  |
| --- |
| this.price = price; |
|  |

|  |
| --- |
| } |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| public String getName() { |
|  |

|  |
| --- |
| return name; |
|  |

|  |
| --- |
| } |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| public void setName(String name) { |
|  |

|  |
| --- |
| this.name = name; |
|  |

|  |
| --- |
| } |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| public String getSurname() { |
|  |

|  |
| --- |
| return surname; |
|  |

|  |
| --- |
| } |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| public void setSurname(String surname) { |
|  |

|  |
| --- |
| this.surname = surname; |
|  |

|  |
| --- |
| } |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| public int getDistance() { |
|  |

|  |
| --- |
| return distance; |
|  |

|  |
| --- |
| } |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| public void setDistance(int distance) { |
|  |

|  |
| --- |
| this.distance = distance; |
|  |

|  |
| --- |
| } |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| public Integer getId() { |
|  |

|  |
| --- |
| return id; |
|  |

|  |
| --- |
| } |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| public void setId(Integer id) { |
|  |

|  |
| --- |
| this.id = id; |
|  |

|  |
| --- |
| } |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| @Override |
|  |

|  |
| --- |
| public String toString() { |
|  |

|  |
| --- |
| return "Ticket{" + |
|  |

|  |
| --- |
| "id=" + id + |
|  |

|  |
| --- |
| ", seat=" + seat + |
|  |

|  |
| --- |
| ", start=" + start + |
|  |

|  |
| --- |
| ", end=" + end + |
|  |

|  |
| --- |
| ", price=" + price + |
|  |

|  |
| --- |
| ", name='" + name + '\'' + |
|  |

|  |
| --- |
| ", surname='" + surname + '\'' + |
|  |

|  |
| --- |
| ", distance=" + distance + |
|  |

|  |
| --- |
| '}'; |
|  |

|  |
| --- |
| } |
|  |

}

|  |
| --- |
| package models; |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| import org.hibernate.annotations.Cascade; |
|  |

|  |
| --- |
| import org.hibernate.annotations.Fetch; |
|  |

|  |
| --- |
| import org.hibernate.annotations.FetchMode; |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| import javax.persistence.\*; |
|  |

|  |
| --- |
| import javax.persistence.criteria.CriteriaBuilder; |
|  |

|  |
| --- |
| import java.util.ArrayList; |
|  |

|  |
| --- |
| import java.util.List; |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| @Entity |
|  |

|  |
| --- |
| @Table(name = "trains") |
|  |

|  |
| --- |
| public class Train { |
|  |

|  |
| --- |
| @Id |
|  |

|  |
| --- |
| @GeneratedValue |
|  |

|  |
| --- |
| @Column(name = "id\_train") |
|  |

|  |
| --- |
| private Integer id; |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| private String name; |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| private String type; |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| @OneToMany(mappedBy = "train", cascade = CascadeType.ALL, orphanRemoval = true, fetch = FetchType.EAGER) |
|  |

|  |
| --- |
| @Fetch(FetchMode.SELECT) |
|  |

|  |
| --- |
| private List<Stop> stops = new ArrayList<>(); |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| @OneToMany(mappedBy = "train", cascade = CascadeType.ALL, orphanRemoval = true, fetch = FetchType.EAGER) |
|  |

|  |
| --- |
| @Fetch(FetchMode.SELECT) |
|  |

|  |
| --- |
| private List<Seat> seats = new ArrayList<>(); |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| public void addSeat(Seat s) { |
|  |

|  |
| --- |
| seats.add(s); |
|  |

|  |
| --- |
| s.setTrain(this); |
|  |

|  |
| --- |
| } |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| public void addStop(Stop s) { |
|  |

|  |
| --- |
| stops.add(s); |
|  |

|  |
| --- |
| s.setTrain(this); |
|  |

|  |
| --- |
| } |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| public Integer getId() { |
|  |

|  |
| --- |
| return id; |
|  |

|  |
| --- |
| } |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| public void setId(Integer id) { |
|  |

|  |
| --- |
| this.id = id; |
|  |

|  |
| --- |
| } |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| public String getName() { |
|  |

|  |
| --- |
| return name; |
|  |

|  |
| --- |
| } |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| public void setName(String name) { |
|  |

|  |
| --- |
| this.name = name; |
|  |

|  |
| --- |
| } |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| public String getType() { |
|  |

|  |
| --- |
| return type; |
|  |

|  |
| --- |
| } |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| public void setType(String type) { |
|  |

|  |
| --- |
| this.type = type; |
|  |

|  |
| --- |
| } |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| public List<Stop> getStops() { |
|  |

|  |
| --- |
| return stops; |
|  |

|  |
| --- |
| } |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| public void setStops(List<Stop> stops) { |
|  |

|  |
| --- |
| this.stops = stops; |
|  |

|  |
| --- |
| } |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| public List<Seat> getSeats() { |
|  |

|  |
| --- |
| return seats; |
|  |

|  |
| --- |
| } |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| public void setSeats(List<Seat> seats) { |
|  |

|  |
| --- |
| this.seats = seats; |
|  |

|  |
| --- |
| } |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| @Override |
|  |

|  |
| --- |
| public String toString() { |
|  |

|  |
| --- |
| return "Train{" + |
|  |

|  |
| --- |
| "id=" + id + |
|  |

|  |
| --- |
| ", name='" + name + '\'' + |
|  |

|  |
| --- |
| ", type='" + type + '\'' + |
|  |

|  |
| --- |
| ", stops=" + stops + |
|  |

|  |
| --- |
| ", seats=" + seats + |
|  |

|  |
| --- |
| '}'; |
|  |

|  |
| --- |
| } |
|  |

}

|  |
| --- |
| package sample; |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| import javafx.collections.FXCollections; |
|  |

|  |
| --- |
| import javafx.collections.ObservableList; |
|  |

|  |
| --- |
| import javafx.event.ActionEvent; |
|  |

|  |
| --- |
| import javafx.fxml.FXML; |
|  |

|  |
| --- |
| import javafx.fxml.Initializable; |
|  |

|  |
| --- |
| import javafx.scene.control.Button; |
|  |

|  |
| --- |
| import javafx.scene.control.DatePicker; |
|  |

|  |
| --- |
| import javafx.scene.control.TableView; |
|  |

|  |
| --- |
| import javafx.scene.control.TextField; |
|  |

|  |
| --- |
| import javafx.scene.paint.Paint; |
|  |

|  |
| --- |
| import javafx.util.StringConverter; |
|  |

|  |
| --- |
| import models.City; |
|  |

|  |
| --- |
| import models.Stop; |
|  |

|  |
| --- |
| import models.Train; |
|  |

|  |
| --- |
| import service.DBTicketService; |
|  |

|  |
| --- |
| import service.DBTrainService; |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| import java.net.URL; |
|  |

|  |
| --- |
| import java.time.LocalDate; |
|  |

|  |
| --- |
| import java.time.format.DateTimeFormatter; |
|  |

|  |
| --- |
| import java.util.List; |
|  |

|  |
| --- |
| import java.util.ResourceBundle; |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| public class Controller implements Initializable { |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| private final String pattern = "yyyy-MM-dd"; |
|  |

|  |
| --- |
| @FXML |
|  |

|  |
| --- |
| private Button buyButton, searchButton; |
|  |

|  |
| --- |
| @FXML |
|  |

|  |
| --- |
| private TextField fieldFrom, fieldTo, fieldCarriage, fieldPlace,fieldName,fieldSurname; |
|  |

|  |
| --- |
| @FXML |
|  |

|  |
| --- |
| private TableView<TrainTableRow> table; |
|  |

|  |
| --- |
| @FXML |
|  |

|  |
| --- |
| private DatePicker fieldDate; |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| @Override |
|  |

|  |
| --- |
| public void initialize(URL location, ResourceBundle resources) { |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| } |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| public void searchEvent(ActionEvent event) { |
|  |

|  |
| --- |
| buyButton.setStyle("-fx-background-color: #60a024; -fx-border-color: #5b9b21; -fx-border-radius: 4;"); |
|  |

|  |
| --- |
| buyButton.setTextFill(Paint.valueOf("WHITE")); |
|  |

|  |
| --- |
| buyButton.setDisable(false); |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| searchButton.setStyle(null); |
|  |

|  |
| --- |
| searchButton.setTextFill(Paint.valueOf("BLACK")); |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| showTable(); |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| } |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| public void buyEvent(ActionEvent event) { |
|  |

|  |
| --- |
| buyButton.setStyle(null); |
|  |

|  |
| --- |
| buyButton.setTextFill(Paint.valueOf("BLACK")); |
|  |

|  |
| --- |
| buyButton.setDisable(true); |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| searchButton.setStyle("-fx-background-color: #60a024; -fx-border-color: #5b9b21; -fx-border-radius: 4;"); |
|  |

|  |
| --- |
| searchButton.setTextFill(Paint.valueOf("WHITE")); |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| buyTicket(); |
|  |

|  |
| --- |
| } |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| public void dateEvent(ActionEvent event) { |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| StringConverter converter = new StringConverter<LocalDate>() { |
|  |

|  |
| --- |
| DateTimeFormatter dateFormatter = |
|  |

|  |
| --- |
| DateTimeFormatter.ofPattern(pattern); |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| @Override |
|  |

|  |
| --- |
| public String toString(LocalDate date) { |
|  |

|  |
| --- |
| if (date != null) { |
|  |

|  |
| --- |
| return dateFormatter.format(date); |
|  |

|  |
| --- |
| } else { |
|  |

|  |
| --- |
| return ""; |
|  |

|  |
| --- |
| } |
|  |

|  |
| --- |
| } |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| @Override |
|  |

|  |
| --- |
| public LocalDate fromString(String string) { |
|  |

|  |
| --- |
| if (string != null && !string.isEmpty()) { |
|  |

|  |
| --- |
| return LocalDate.parse(string, dateFormatter); |
|  |

|  |
| --- |
| } else { |
|  |

|  |
| --- |
| return null; |
|  |

|  |
| --- |
| } |
|  |

|  |
| --- |
| } |
|  |

|  |
| --- |
| }; |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| fieldDate.setConverter(converter); |
|  |

|  |
| --- |
| fieldDate.setPromptText(pattern.toLowerCase()); |
|  |

|  |
| --- |
| fieldDate.requestFocus(); |
|  |

|  |
| --- |
| } |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| private void showTable() { |
|  |

|  |
| --- |
| table.setItems(getTrainTableRow()); |
|  |

|  |
| --- |
| } |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| private ObservableList<TrainTableRow> getTrainTableRow() { |
|  |

|  |
| --- |
| ObservableList<TrainTableRow> trainTableRows = FXCollections.observableArrayList(); |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| List<Train> needTrains = DBTrainService.getInstance().search(fieldFrom.getText(), fieldTo.getText(), fieldDate.getValue()); |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| List<Stop> stopTrains; |
|  |

|  |
| --- |
| DBTrainService city = new DBTrainService(); |
|  |

|  |
| --- |
| List<City> cities = city.getAllCity(); |
|  |

|  |
| --- |
| String start = ""; |
|  |

|  |
| --- |
| String end = ""; |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| for (Train train : needTrains) { |
|  |

|  |
| --- |
| stopTrains = train.getStops(); |
|  |

|  |
| --- |
| for (Stop stop : stopTrains) { |
|  |

|  |
| --- |
| if (fieldFrom.getText().equals(cities.get(stop.getCity() - 1).getCityName())) { |
|  |

|  |
| --- |
| start = stop.getDate().toString(); |
|  |

|  |
| --- |
| } |
|  |

|  |
| --- |
| if (fieldTo.getText().equals(cities.get(stop.getCity() - 1).getCityName())) { |
|  |

|  |
| --- |
| end = stop.getDate().toString(); |
|  |

|  |
| --- |
| } |
|  |

|  |
| --- |
| } |
|  |

|  |
| --- |
| trainTableRows.add(new TrainTableRow(train.getId(), train.getName(), train.getType(), fieldFrom.getText() + " - " + fieldTo.getText(), start, end, 2, |
|  |

|  |
| --- |
| train.getStops())); |
|  |

|  |
| --- |
| } |
|  |

|  |
| --- |
| return trainTableRows; |
|  |

|  |
| --- |
| } |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| private void buyTicket() { |
|  |

|  |
| --- |
| TrainTableRow selectedTrain; |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| selectedTrain = table.getSelectionModel().getSelectedItem(); |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| DBTicketService dbTicketService = new DBTicketService(); |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| int start, end; |
|  |

|  |
| --- |
| start = end = 0; |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| DBTrainService city = new DBTrainService(); |
|  |

|  |
| --- |
| List<City> cities = city.getAllCity(); |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| for (City cityObject : cities) { |
|  |

|  |
| --- |
| if (cityObject.getCityName().equalsIgnoreCase(fieldFrom.getText())) { |
|  |

|  |
| --- |
| start = cityObject.getId(); |
|  |

|  |
| --- |
| } |
|  |

|  |
| --- |
| if (cityObject.getCityName().equalsIgnoreCase(fieldTo.getText())) { |
|  |

|  |
| --- |
| end = cityObject.getId(); |
|  |

|  |
| --- |
| } |
|  |

|  |
| --- |
| } |
|  |

|  |
| --- |
| dbTicketService.buy(selectedTrain.getStopList(),start,end,fieldName.getText(),fieldSurname.getText(),fieldCarriage.getText(),fieldPlace.getText(),selectedTrain.getType(), |
|  |

|  |
| --- |
| selectedTrain.getId()); |
|  |

|  |
| --- |
| } |
|  |

}

|  |
| --- |
| package sample; |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| import javax.persistence.EntityManager; |
|  |

|  |
| --- |
| import javax.persistence.EntityManagerFactory; |
|  |

|  |
| --- |
| import javax.persistence.Persistence; |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| public enum PersistenceManager { |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| INSTANCE; |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| private EntityManagerFactory emFactory; |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| private PersistenceManager() { |
|  |

|  |
| --- |
| emFactory = Persistence.createEntityManagerFactory("TSPU"); |
|  |

|  |
| --- |
| } |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| public EntityManager getEntityManager() { |
|  |

|  |
| --- |
| return emFactory.createEntityManager(); |
|  |

|  |
| --- |
| } |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| public void close() { |
|  |

|  |
| --- |
| emFactory.close(); |
|  |

|  |
| --- |
| } |
|  |

|  |
| --- |
|  |
|  |

}

|  |
| --- |
| package sample; |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| import javafx.application.Application; |
|  |

|  |
| --- |
| import javafx.collections.FXCollections; |
|  |

|  |
| --- |
| import javafx.collections.ObservableList; |
|  |

|  |
| --- |
| import javafx.fxml.FXMLLoader; |
|  |

|  |
| --- |
| import javafx.scene.Group; |
|  |

|  |
| --- |
| import javafx.scene.Parent; |
|  |

|  |
| --- |
| import javafx.scene.Scene; |
|  |

|  |
| --- |
| import javafx.scene.control.TableColumn; |
|  |

|  |
| --- |
| import javafx.scene.control.TableView; |
|  |

|  |
| --- |
| import javafx.scene.control.cell.PropertyValueFactory; |
|  |

|  |
| --- |
| import javafx.stage.Stage; |
|  |

|  |
| --- |
| import java.io.IOException; |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| public class Ticket\_Service extends Application { |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| public static void main(String[] args) { |
|  |

|  |
| --- |
| launch(args); |
|  |

|  |
| --- |
| } |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| @Override |
|  |

|  |
| --- |
| public void start(Stage primaryStage) throws IOException { |
|  |

|  |
| --- |
| Parent root = FXMLLoader.load(getClass().getClassLoader().getResource("sample.fxml")); |
|  |

|  |
| --- |
| Scene scene = new Scene(root, 1160, 650); |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| primaryStage.setResizable(false); |
|  |

|  |
| --- |
| primaryStage.setScene(scene); |
|  |

|  |
| --- |
| primaryStage.setTitle("Ticket Service"); |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| primaryStage.show(); |
|  |

|  |
| --- |
| } |
|  |

}

|  |
| --- |
| package sample; |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| import models.Stop; |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| import java.util.List; |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| public class TrainTableRow { |
|  |

|  |
| --- |
| private Integer id; |
|  |

|  |
| --- |
| private String name; |
|  |

|  |
| --- |
| private String type; |
|  |

|  |
| --- |
| private String route; |
|  |

|  |
| --- |
| private String departureTime; |
|  |

|  |
| --- |
| private String arrivalTime; |
|  |

|  |
| --- |
| private Integer numberOfCars; |
|  |

|  |
| --- |
| private List<Stop> stopList; |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| public TrainTableRow(Integer id, String name, String type, String route, String departureTime, String arrivalTime, Integer numberOfCars,List<Stop> stops) { |
|  |

|  |
| --- |
| this.id = id; |
|  |

|  |
| --- |
| this.name = name; |
|  |

|  |
| --- |
| this.type = type; |
|  |

|  |
| --- |
| this.route = route; |
|  |

|  |
| --- |
| this.departureTime = departureTime; |
|  |

|  |
| --- |
| this.arrivalTime = arrivalTime; |
|  |

|  |
| --- |
| this.numberOfCars = numberOfCars; |
|  |

|  |
| --- |
| this.stopList = stops; |
|  |

|  |
| --- |
| } |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| public TrainTableRow() { |
|  |

|  |
| --- |
| this.id = 0; |
|  |

|  |
| --- |
| this.name = "ERROR"; |
|  |

|  |
| --- |
| this.type = "ERROR"; |
|  |

|  |
| --- |
| this.route = "ERROR"; |
|  |

|  |
| --- |
| this.departureTime = "ERROR"; |
|  |

|  |
| --- |
| this.arrivalTime = "ERROR"; |
|  |

|  |
| --- |
| this.numberOfCars = 0; |
|  |

|  |
| --- |
| } |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| public Integer getId() { |
|  |

|  |
| --- |
| return id; |
|  |

|  |
| --- |
| } |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| public void setId(Integer id) { |
|  |

|  |
| --- |
| this.id = id; |
|  |

|  |
| --- |
| } |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| public String getName() { |
|  |

|  |
| --- |
| return name; |
|  |

|  |
| --- |
| } |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| public void setName(String name) { |
|  |

|  |
| --- |
| this.name = name; |
|  |

|  |
| --- |
| } |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| public String getType() { |
|  |

|  |
| --- |
| return type; |
|  |

|  |
| --- |
| } |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| public void setType(String type) { |
|  |

|  |
| --- |
| this.type = type; |
|  |

|  |
| --- |
| } |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| public String getRoute() { |
|  |

|  |
| --- |
| return route; |
|  |

|  |
| --- |
| } |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| public void setRoute(String route) { |
|  |

|  |
| --- |
| this.route = route; |
|  |

|  |
| --- |
| } |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| public String getDepartureTime() { |
|  |

|  |
| --- |
| return departureTime; |
|  |

|  |
| --- |
| } |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| public void setDepartureTime(String departureTime) { |
|  |

|  |
| --- |
| this.departureTime = departureTime; |
|  |

|  |
| --- |
| } |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| public String getArrivalTime() { |
|  |

|  |
| --- |
| return arrivalTime; |
|  |

|  |
| --- |
| } |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| public void setArrivalTime(String arrivalTime) { |
|  |

|  |
| --- |
| this.arrivalTime = arrivalTime; |
|  |

|  |
| --- |
| } |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| public Integer getNumberOfCars() { |
|  |

|  |
| --- |
| return numberOfCars; |
|  |

|  |
| --- |
| } |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| public void setNumberOfCars(Integer numberOfCars) { |
|  |

|  |
| --- |
| this.numberOfCars = numberOfCars; |
|  |

|  |
| --- |
| } |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| public List<Stop> getStopList() { |
|  |

|  |
| --- |
| return stopList; |
|  |

|  |
| --- |
| } |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| public void setStopList(List<Stop> stopList) { |
|  |

|  |
| --- |
| this.stopList = stopList; |
|  |

|  |
| --- |
| } |
|  |

|  |
| --- |
|  |
|  |

}

|  |
| --- |
| package service; |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| import models.Stop; |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| import java.util.ArrayList; |
|  |

|  |
| --- |
| import java.util.List; |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| public abstract class AbstractTicketService implements TicketService { |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| public int getDistance(List<Stop> stop, int start, int end) { |
|  |

|  |
| --- |
| List<Integer> stops = new ArrayList<>(); |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| boolean inStopList = false; |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| for (Stop stoping : stop) { |
|  |

|  |
| --- |
| if (stoping.getCity() == start) { |
|  |

|  |
| --- |
| inStopList = true; |
|  |

|  |
| --- |
| } |
|  |

|  |
| --- |
| if (inStopList) { |
|  |

|  |
| --- |
| stops.add(stoping.getCity()); |
|  |

|  |
| --- |
| } |
|  |

|  |
| --- |
| if (stoping.getCity() == end) { |
|  |

|  |
| --- |
| break; |
|  |

|  |
| --- |
| } |
|  |

|  |
| --- |
| } |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| int distance = 0; |
|  |

|  |
| --- |
| for (int i = 0; i < stops.size() - 1; i++) { |
|  |

|  |
| --- |
| distance += arrStop(stops.get(i)-1, stops.get(i + 1)-1); |
|  |

|  |
| --- |
| } |
|  |

|  |
| --- |
| return distance; |
|  |

|  |
| --- |
| } |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| @Override |
|  |

|  |
| --- |
| public double getPrice(int distance, String type) { |
|  |

|  |
| --- |
| double price = 0; |
|  |

|  |
| --- |
| switch (type) { |
|  |

|  |
| --- |
| case "Night": { |
|  |

|  |
| --- |
| price = distance / 5; |
|  |

|  |
| --- |
| } |
|  |

|  |
| --- |
| case "Day": { |
|  |

|  |
| --- |
| price = distance / 10; |
|  |

|  |
| --- |
| } |
|  |

|  |
| --- |
| } |
|  |

|  |
| --- |
| return price; |
|  |

|  |
| --- |
| } |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| private int arrStop(int from, int to) { |
|  |

|  |
| --- |
| final int array[][] = { |
|  |

|  |
| --- |
| {0, 299, 250, 401, 464, 378, 437, 489, 653, 608, 1042, 793, 949, 928, 1110, 1180, 1448, 935, 1232, 1136, 1172, 1258, 1280, 1438, 1604}, |
|  |

|  |
| --- |
| {299, 0, 134, 285, 289, 134, 138, 245, 429, 364, 798, 569, 705, 684, 866, 936, 1204, 711, 1008, 912, 928, 1014, 1056, 1194, 1360}, |
|  |

|  |
| --- |
| {299, 134, 0, 151, 214, 128, 272, 239, 403, 358, 792, 543, 699, 678, 860, 930, 1198, 685, 982, 886, 922, 1008, 1030, 1188, 1354}, |
|  |

|  |
| --- |
| {450, 285, 151, 0, 73, 228, 402, 267, 262, 386, 820, 402, 570, 700, 882, 952, 1220, 544, 841, 745, 915, 1001, 889, 1181, 1219}, |
|  |

|  |
| --- |
| {513, 289, 214, 73, 0, 155, 329, 194, 189, 313, 747, 329, 497, 627, 809, 879, 1147, 471, 768, 672, 842, 928, 816, 1108, 1146}, |
|  |

|  |
| --- |
| {427, 134, 128, 228, 155, 0, 174, 111, 295, 230, 664, 435, 571, 550, 732, 802, 1070, 577, 874, 778, 794, 880, 922, 1060, 1226}, |
|  |

|  |
| --- |
| {437, 138, 272, 402, 329, 174, 0, 190, 374, 273, 707, 514, 614, 593, 775, 845, 1113, 656, 953, 839, 837, 923, 983, 1103, 1269}, |
|  |

|  |
| --- |
| {538, 245, 239, 267, 194, 111, 190, 0, 184, 119, 553, 324, 460, 439, 621, 691, 959, 466, 763, 667, 683, 769, 811, 949, 1115}, |
|  |

|  |
| --- |
| {702, 429, 403, 262, 189, 295, 374, 184, 0, 150, 584, 140, 308, 438, 620, 690, 958, 282, 579, 483, 653, 739, 627, 919, 957}, |
|  |

|  |
| --- |
| {657, 364, 358, 386, 313, 230, 273, 119, 150, 0, 434, 268, 341, 320, 502, 572, 840, 410, 707, 566, 564, 650, 710, 830, 996}, |
|  |

|  |
| --- |
| {1091, 798, 792, 820, 747, 664, 707, 553, 584, 434, 0, 613, 445, 315, 133, 203, 471, 755, 736, 561, 525, 490, 705, 720, 886}, |
|  |

|  |
| --- |
| {842, 569, 543, 402, 329, 435, 514, 324, 140, 268, 613, 0, 168, 298, 480, 550, 818, 142, 439, 343, 513, 599, 487, 779, 817}, |
|  |

|  |
| --- |
| {998, 705, 699, 570, 497, 571, 614, 460, 308, 341, 445, 168, 0, 130, 312, 382, 650, 310, 409, 234, 374, 460, 378, 640, 708}, |
|  |

|  |
| --- |
| {977, 684, 678, 700, 627, 550, 593, 439, 438, 320, 315, 298, 130, 0, 182, 252, 520, 440, 421, 246, 244, 330, 390, 510, 676}, |
|  |

|  |
| --- |
| {1159, 866, 860, 882, 809, 732, 775, 621, 620, 502, 133, 480, 312, 182, 0, 70, 338, 622, 603, 428, 392, 357, 572, 587, 753}, |
|  |

|  |
| --- |
| {1229, 936, 930, 952, 879, 802, 845, 691, 690, 572, 203, 550, 382, 252, 70, 0, 268, 692, 673, 498, 330, 287, 553, 517, 683}, |
|  |

|  |
| --- |
| {1497, 1204, 1198, 1220, 1147, 1070, 1113, 959, 958, 840, 471, 818, 650, 520, 338, 268, 0, 960, 941, 766, 598, 555, 821, 785, 951}, |
|  |

|  |
| --- |
| {984, 711, 685, 544, 471, 577, 656, 466, 282, 410, 755, 142, 310, 440, 622, 692, 960, 0, 297, 405, 575, 661, 482, 778, 812}, |
|  |

|  |
| --- |
| {1281, 1008, 982, 841, 768, 874, 953, 763, 579, 707, 736, 439, 409, 421, 603, 673, 941, 297, 0, 175, 345, 431, 185, 481, 515}, |
|  |

|  |
| --- |
| {1185, 912, 886, 745, 672, 778, 839, 667, 483, 566, 561, 343, 234, 246, 428, 498, 766, 405, 175, 0, 170, 256, 144, 436, 474}, |
|  |

|  |
| --- |
| {1221, 928, 922, 915, 842, 794, 837, 683, 653, 564, 525, 513, 374, 244, 392, 330, 598, 575, 345, 170, 0, 86, 223, 266, 432}, |
|  |

|  |
| --- |
| {1307, 1014, 1008, 1001, 928, 880, 923, 769, 739, 650, 490, 599, 460, 330, 357, 287, 555, 661, 431, 256, 86, 0, 309, 230, 396}, |
|  |

|  |
| --- |
| {1329, 1056, 1030, 889, 816, 922, 983, 811, 627, 710, 705, 487, 378, 390, 572, 553, 821, 482, 185, 144, 223, 309, 0, 296, 330}, |
|  |

|  |
| --- |
| {1487, 1194, 1188, 1181, 1108, 1060, 1103, 949, 919, 830, 720, 779, 640, 510, 587, 517, 785, 778, 481, 436, 266, 230, 296, 0, 166}, |
|  |

|  |
| --- |
| {1653, 1360, 1354, 1219, 1146, 1226, 1269, 1115, 957, 996, 886, 817, 708, 676, 753, 683, 951, 812, 515, 474, 432, 396, 330, 166, 0} |
|  |

|  |
| --- |
| }; |
|  |

|  |
| --- |
| return array[from][to]; |
|  |

|  |
| --- |
| } |
|  |

}

|  |
| --- |
| package service; |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| import models.City; |
|  |

|  |
| --- |
| import models.Train; |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| import java.time.LocalDate; |
|  |

|  |
| --- |
| import java.util.ArrayList; |
|  |

|  |
| --- |
| import java.util.Date; |
|  |

|  |
| --- |
| import java.util.List; |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| public abstract class AbstractTrainService implements TrainService { |
|  |

|  |
| --- |
| @Override |
|  |

|  |
| --- |
| public List<Integer> searchCity(String start, String end){ |
|  |

|  |
| --- |
| List<City> allCity = getAllCity(); |
|  |

|  |
| --- |
| List<Integer> startEnd = new ArrayList<>(); |
|  |

|  |
| --- |
| startEnd.add(null); |
|  |

|  |
| --- |
| startEnd.add(null); |
|  |

|  |
| --- |
| for (City count : allCity) { |
|  |

|  |
| --- |
| if (start.equalsIgnoreCase(count.getCityName())) { |
|  |

|  |
| --- |
| startEnd.set(0,count.getId()); |
|  |

|  |
| --- |
| } |
|  |

|  |
| --- |
| if (end.equalsIgnoreCase(count.getCityName())) { |
|  |

|  |
| --- |
| startEnd.set(1,count.getId()); |
|  |

|  |
| --- |
| } |
|  |

|  |
| --- |
| } |
|  |

|  |
| --- |
| return startEnd; |
|  |

|  |
| --- |
| } |
|  |

|  |
| --- |
| @Override |
|  |

|  |
| --- |
| public List<Train> search(String start, String end, LocalDate date) { |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| List<Train> allTrains = getAllTrains(); |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| Integer startInt = searchCity(start,end).get(0); |
|  |

|  |
| --- |
| Integer endInt = searchCity(start,end).get(1); |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| boolean startBool,dateBool; |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| String needDate = (date.toString()); |
|  |

|  |
| --- |
| List<Train> resultTrains = new ArrayList<>(); |
|  |

|  |
| --- |
| for (Train count : allTrains) { |
|  |

|  |
| --- |
| startBool =dateBool = false; |
|  |

|  |
| --- |
| for (int i = 0; i < count.getStops().size(); i++) { |
|  |

|  |
| --- |
| if (count.getStops().get(i).getCity() == startInt) { |
|  |

|  |
| --- |
| startBool = true; |
|  |

|  |
| --- |
| String getDate = (count.getStops().get(i).getDate().toString()); |
|  |

|  |
| --- |
| getDate = getDate.substring(0,10); |
|  |

|  |
| --- |
| if(getDate.equalsIgnoreCase(needDate)){ |
|  |

|  |
| --- |
| dateBool = true; |
|  |

|  |
| --- |
| } |
|  |

|  |
| --- |
| } |
|  |

|  |
| --- |
| if (count.getStops().get(i).getCity() == endInt && startBool &&dateBool) { |
|  |

|  |
| --- |
| resultTrains.add(count); |
|  |

|  |
| --- |
| } |
|  |

|  |
| --- |
| } |
|  |

|  |
| --- |
| } |
|  |

|  |
| --- |
| return resultTrains; |
|  |

|  |
| --- |
| } |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| abstract List<Train> getAllTrains(); |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| abstract List<City> getAllCity(); |
|  |

}

|  |
| --- |
| package service; |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| import models.Seat; |
|  |

|  |
| --- |
| import models.Stop; |
|  |

|  |
| --- |
| import models.Ticket; |
|  |

|  |
| --- |
| import sample.PersistenceManager; |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| import javax.persistence.EntityManager; |
|  |

|  |
| --- |
| import javax.persistence.Query; |
|  |

|  |
| --- |
| import java.util.List; |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| public class DBTicketService extends AbstractTicketService { |
|  |

|  |
| --- |
| @Override |
|  |

|  |
| --- |
| public void buy(List<Stop> stop, int start, int end, String name, String surname, String carriage, String number, String type, int id) { |
|  |

|  |
| --- |
| Ticket ticket = new Ticket(); |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| EntityManager em = PersistenceManager.INSTANCE.getEntityManager(); |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| ticket.setName(name); |
|  |

|  |
| --- |
| ticket.setSurname(surname); |
|  |

|  |
| --- |
| ticket.setStart(start); |
|  |

|  |
| --- |
| ticket.setEnd(end); |
|  |

|  |
| --- |
| ticket.setSeat(seat(Integer.parseInt(carriage), Integer.parseInt(number), id)); |
|  |

|  |
| --- |
| ticket.setDistance(getDistance(stop, start, end)); |
|  |

|  |
| --- |
| ticket.setPrice(getPrice(ticket.getDistance(), type)); |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| em.getTransaction().begin(); |
|  |

|  |
| --- |
| em.persist(ticket); |
|  |

|  |
| --- |
| em.getTransaction().commit(); |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| em.close(); |
|  |

|  |
| --- |
| } |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| public Seat seat(int carriage, int number, int id) { |
|  |

|  |
| --- |
| EntityManager em = PersistenceManager.INSTANCE.getEntityManager(); |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| Query query = em.createQuery("from Seat"); |
|  |

|  |
| --- |
| List resultList = query.getResultList(); |
|  |

|  |
| --- |
| Seat seat = new Seat(); |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| for (Object seats : resultList) { |
|  |

|  |
| --- |
| if (carriage == ((Seat) seats).getCarriage() && number == ((Seat) seats).getNumber() && id == ((Seat) seats).getTrain().getId()) { |
|  |

|  |
| --- |
| seat = (Seat) seats; |
|  |

|  |
| --- |
| } |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| } |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| em.close(); |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| return seat; |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| } |
|  |

}

|  |
| --- |
| package service; |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| import models.City; |
|  |

|  |
| --- |
| import models.Train; |
|  |

|  |
| --- |
| import sample.PersistenceManager; |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| import javax.persistence.EntityManager; |
|  |

|  |
| --- |
| import javax.persistence.Query; |
|  |

|  |
| --- |
| import java.util.ArrayList; |
|  |

|  |
| --- |
| import java.util.List; |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| public class DBTrainService extends AbstractTrainService { |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| private static TrainService instance; |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| public static TrainService getInstance() { |
|  |

|  |
| --- |
| if (instance == null) { |
|  |

|  |
| --- |
| instance = new DBTrainService(); |
|  |

|  |
| --- |
| } |
|  |

|  |
| --- |
| return instance; |
|  |

|  |
| --- |
| } |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| public DBTrainService() { } |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| public List<City> getAllCity() { |
|  |

|  |
| --- |
| EntityManager em = PersistenceManager.INSTANCE.getEntityManager(); |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| Query query = em.createQuery("from City"); |
|  |

|  |
| --- |
| List resultList = query.getResultList(); |
|  |

|  |
| --- |
| List<City> cities = new ArrayList<>(); |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| for (Object city : resultList) { |
|  |

|  |
| --- |
| cities.add((City) city); |
|  |

|  |
| --- |
| } |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| em.close(); |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| return cities; |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| } |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| List<Train> getAllTrains() { |
|  |

|  |
| --- |
| EntityManager em = PersistenceManager.INSTANCE.getEntityManager(); |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| Query query = em.createQuery("from Train"); |
|  |

|  |
| --- |
| List resultList = query.getResultList(); |
|  |

|  |
| --- |
| List<Train> trains = new ArrayList<>(); |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| for (Object train : resultList) { |
|  |

|  |
| --- |
| trains.add((Train) train); |
|  |

|  |
| --- |
| } |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| em.close(); |
|  |

|  |
| --- |
| return trains; |
|  |

|  |
| --- |
| } |
|  |

}

|  |
| --- |
| package service; |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| import models.Stop; |
|  |

|  |
| --- |
| import models.Ticket; |
|  |

|  |
| --- |
| import models.Train; |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| import java.util.List; |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| public interface TicketService { |
|  |

|  |
| --- |
| void buy(List<Stop> stop, int start, int end, String name, String surname, String carriage, String number, String type,int id); |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| double getPrice(int distance, String type); |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| int getDistance(List<Stop> stop, int start, int end); |
|  |

}

|  |
| --- |
| package service; |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| import models.Train; |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| import java.time.LocalDate; |
|  |

|  |
| --- |
| import java.util.Date; |
|  |

|  |
| --- |
| import java.util.List; |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| public interface TrainService { |
|  |

|  |
| --- |
| List<Integer> searchCity(String start, String end); |
|  |

|  |
| --- |
| List<Train> search(String start,String end,LocalDate date); |
|  |

}

## ДОДАТОК Б

Інтерфейс програми

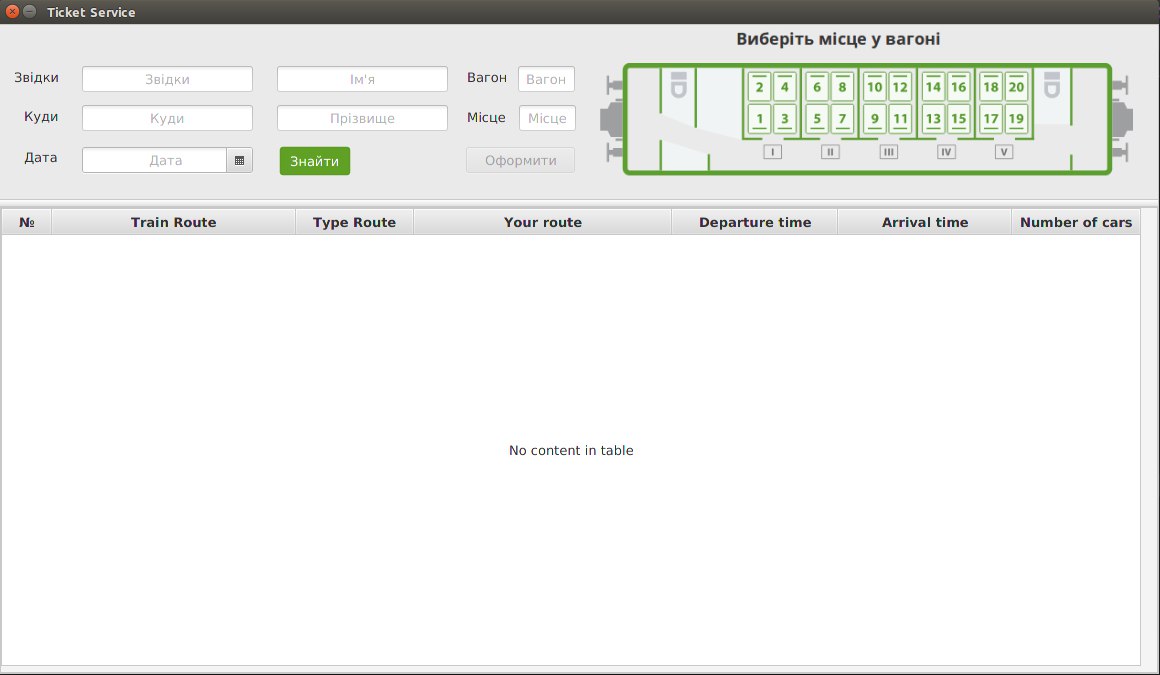


Рисунок Б.1- Інтерфейс програми

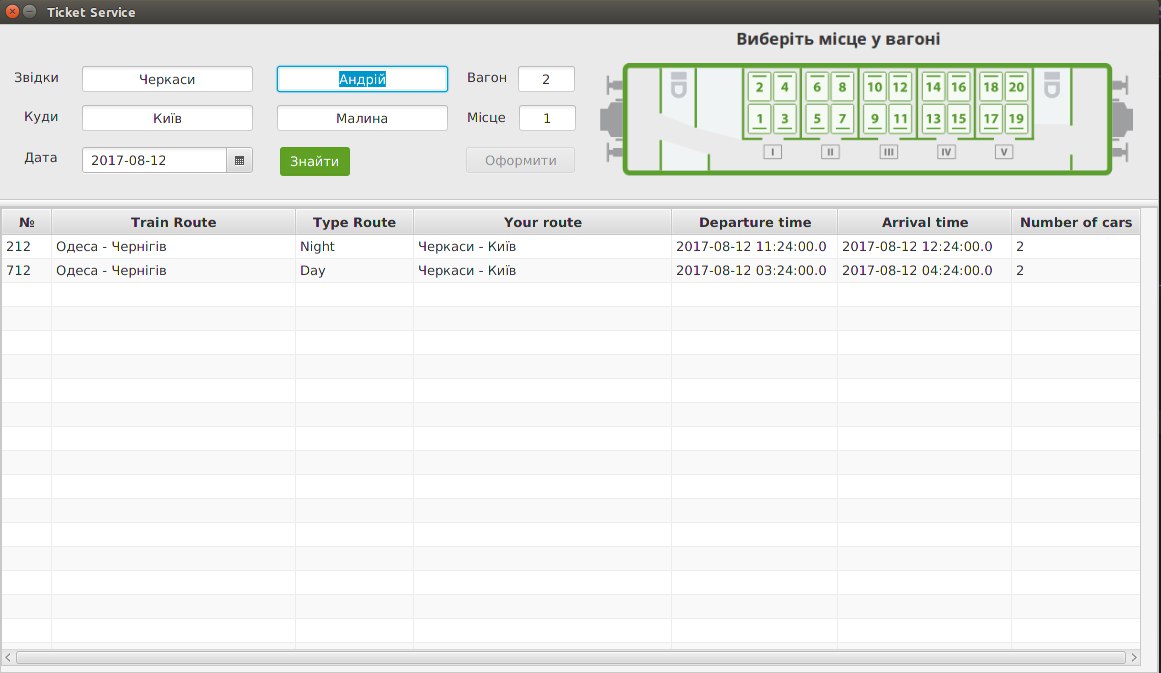


Рисунок Б.2-Пошук

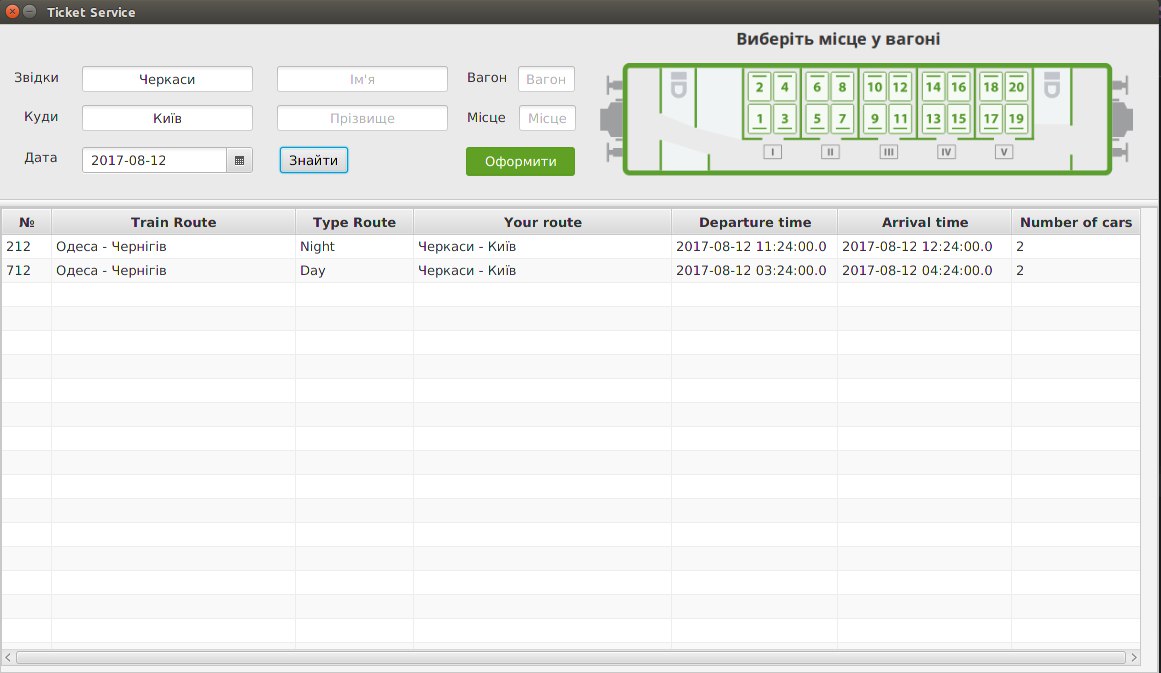


Рисунок Б.2-Оформлення

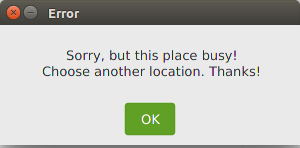


Рисунок Б.2-Помилка