**НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНИВЕРСИТЕТ УКРАЇНИ**

**«Київський політехнічний інститут»**

Факультет інформатики та обчислювальної техніки

Кафедра автоматизованої обробки інформації та управління

**КУРСОВА РОБОТА**

по дисципліні

Об'єктно-орієнтоване програмування – 1

**Варіант No 8**

Виконала: студентка групи ІП-81

Качан О. В.

Прийняла: доцент Іванова Л.М.

Київ – 2020

**ЗМІСТ**

Contents

[**ЗМІСТ** 2](#_Toc42218294)

[**ЗАВДАННЯ** 3](#_Toc42218295)

[**ОПИС ЗАСТОСУВАННЯ** 5](#_Toc42218296)

[Архітектура застосування 5](#_Toc42218297)

[Діаграма варіантів використання 7](#_Toc42218298)

[Діаграма класів 8](#_Toc42218299)

[Таблиця 1. Специфікація методів 10](#_Toc42218300)

[**РЕКОМЕНДАЦІЇ ЩОДО ВИКОРИСТАННЯ** 21](#_Toc42218301)

[Технічні характеристики 21](#_Toc42218302)

[Інкструкція використання 22](#_Toc42218303)

[РЕКОМЕНДАЦІЇ КОРИСТУВАЧЕВІ 23](#_Toc42218304)

[**СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ** 25](#_Toc42218305)

**ЗАВДАННЯ**

Написати консольне застосування за архітектурним шаблоном MVC з розподіленням відповідальності між компонентами, яке виконує обробку набору даних згідно з варіантом завдання (додаток А) та забезпечує наступне:

- Збереження даних у файлі (формат файлу буль-який);

- Читання даних у пам'ять при запуску застосунку;

- Збереження даних у той же файл при завершені роботи застосунку, якщо дані були змінені;

- Збереження проміжних даних у будь-який файл (користувач вводить ім’я файлу);

- Інтеракивність з користувачем (мова інтерфейсу має обиратися при запуску

застосунку на виконання);

- Логування подій та помилок в роботі застосунку.

Варіант 8. **Поїзд**:

Пункт призначення, Номер поїзда, Час відправлення, Кількість місць (загальних, купе, плацкарт, СВ). Конструктор, Методи доступу, Метод *toString()*

1. Отримати список поїздів, які мають загальні місця.

2. Отримати список поїздів, які слідують до заданого пункту призначення та відправляються після заданого часу.

**ОПИС ЗАСТОСУВАННЯ**

Архітектура застосування

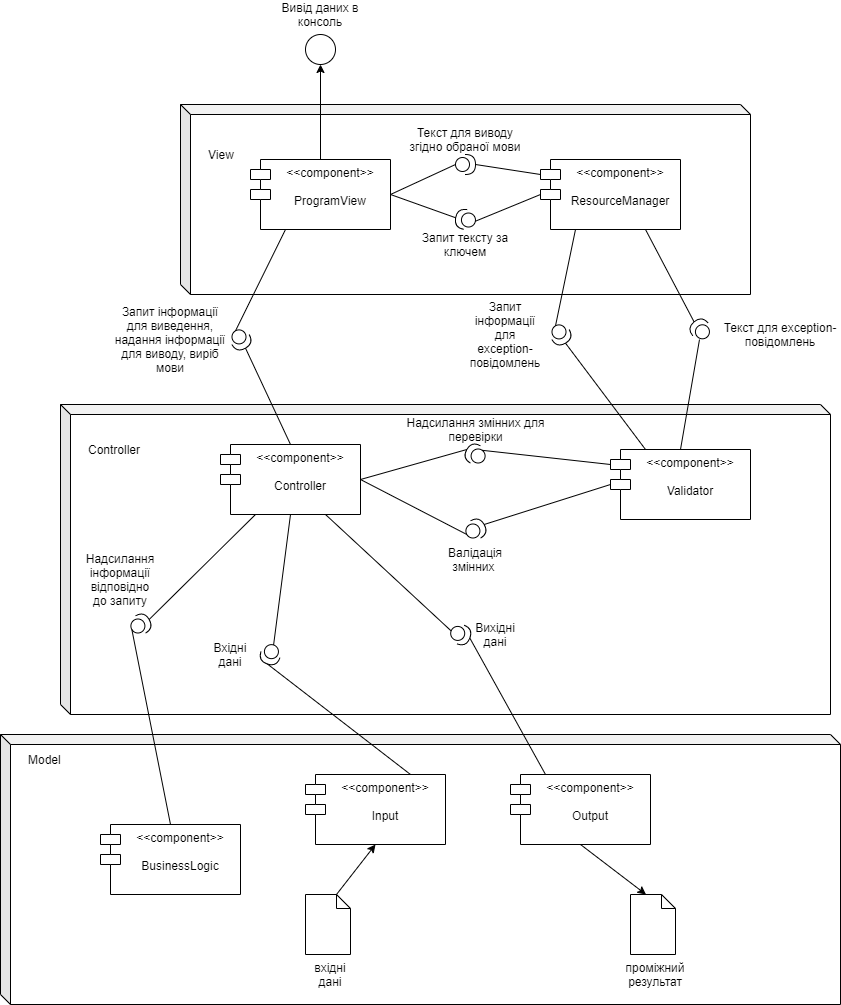


Рисунок 1 – Архітектура застосування

На рисунку 1 зображено трьохрівневу архітектуру застосування програмного забезпечення.

До складу першого рівня входять компоненти ProgramView та ResourceManager. Компонент ProgramView призначений для відображення інформації в консолі, виводу вхідних даних, а також фіксації мови інтерфейсу. ResourceManager, в свою чергу, отримує від ProgramView запити щодо рядків, які мають бути виведені в консоль, в тому числі і запити щодо мови інтерфейсу.

До складу другого рівня – Controller – входять компоненти Controller та Validator. Controller здійснює керування програмою, а саме: передачу вхідних даних та запиту користувача на BusinessLogic, надсилання результату, отриманого з BusinessLogic на Output, надсилає запити щодо відображення певної інормації на ProgramView, а також здійснює валідацію даних за допомогою Validator. Validator перевіряє корректність даних, отриманих в резльтаті читання файлу, і також даних, введених користувачем.

До складу третього рівня – Model – входять компоненти BusinessLogic, Input та Output. BusinessLogic здійснює обробку та відбір даних згідно запиту користувача; Input – клас, який здійснює зчитування даних з файлу, а також отримує дані, введені користувачем; Output – клас, який здійснює запис отриманих даних в файл.

Діаграма варіантів використання

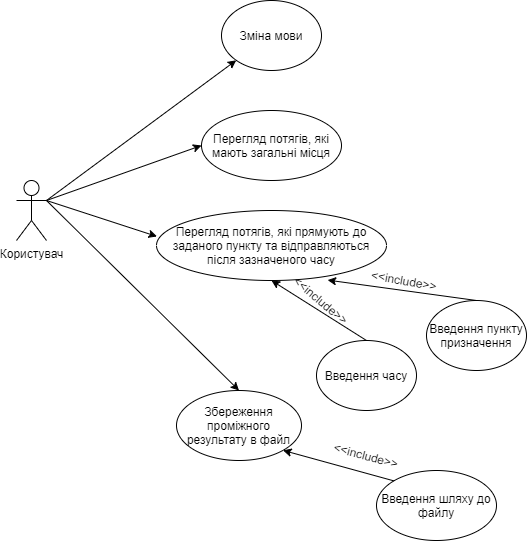


Рисунок 2 - Діаграма способів використання

На рисунку 2 зображено способи використання програми, серед яких є:

- зміна мови;

- перегляд потягів, які мають загальні місця;

- перегляд потягів, які прямують до заданого пункту та відправляються після зазначеного часу, що включає в себе:

* введення часу;
* введення пункту призначення;

- збереження проміжного результату в файл, що включає в себе:

* введення шляху до файлу.

Діаграма класів

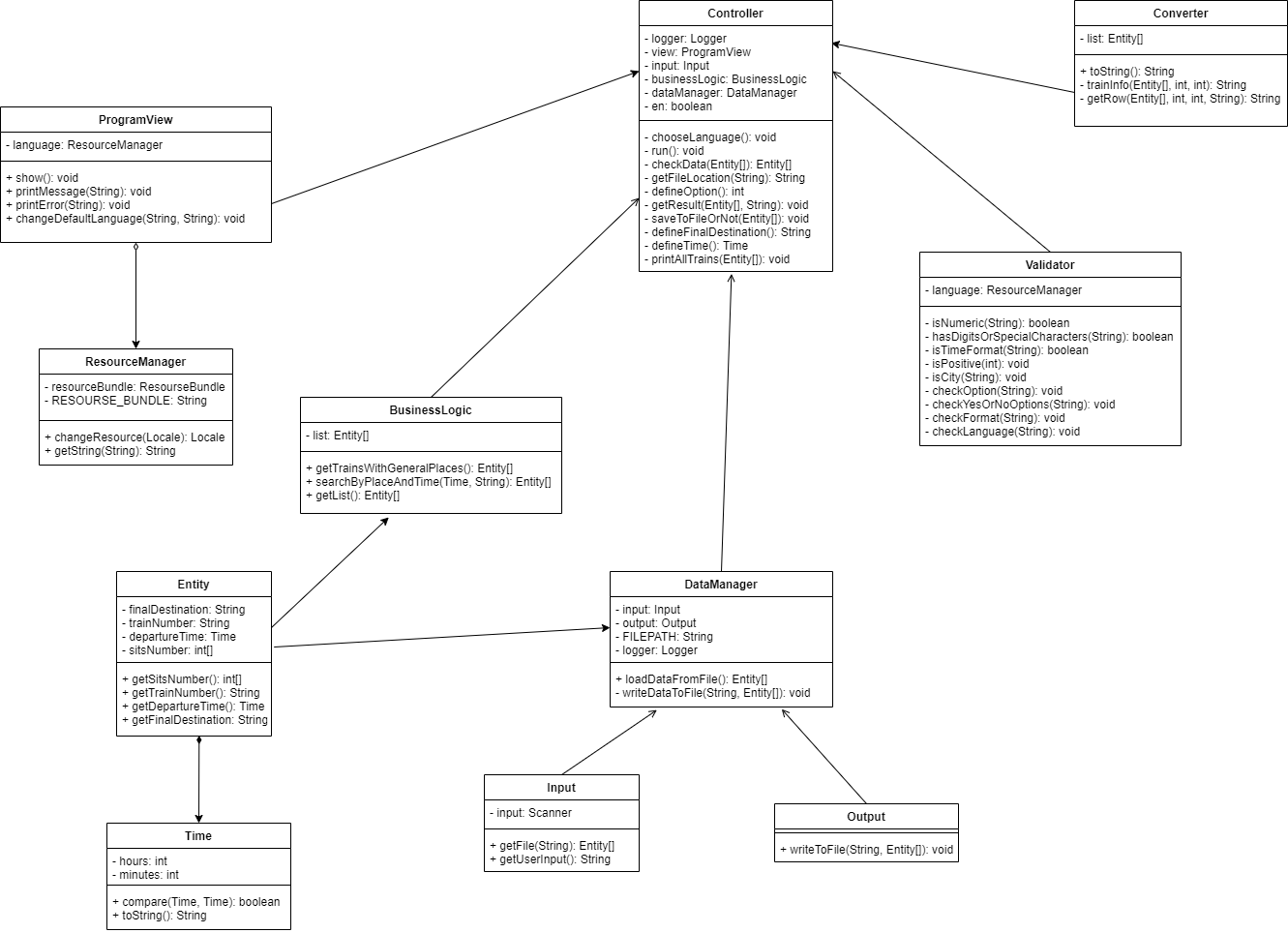


Рисунок 3 – діаграма класів

Програма поділена на 3 пакети: view, controller та model.

Пакет view містить класи ProgramView та ResourceManager. Клас ProgramView здійснює вивід в консоль повідомлень, помилок, списку всіх потягів, проміжних результатів, отриманих згідно запиту користувача, а також визначає мову інтерфейсу. Клас ResourseManager містить в собі мову інтерфейсу та постачає ProgramView рядками згідно зазначеної мови.

Пакет controller містить наступні класи: Controller, Converter, DataManager та Validator. Клас Controller контролює процес виконання програми: дістає вхідні дані та відправляє вихідні дані в DataManager, керує запитами користувача, відправляє на виведення повідомлення, помилки та результати запиту, здійснює валідацію вхідних даних. Клас Validator отримує дані з класу Controller, здійснює їх перевірку та надсилає до Controller. Converter в свою чергу отримує результат запиту з класу Controller, який в свою чергу отримав ці дані з BusinessLogic, та здійснює їх конвертацію в дані типу String, аби Controller зміг надіслати ці дані на ProgramView. Клас DataManager керує вхідними та вихідними даними, тобто здійснює читання та запис файлу за допомогою класів Input та Output.

Пакет model містить в собі класи BusinessLogic, Input та Output, а також пакети entity, до якого входять Entity та Time, і exceptions, який містить custom-виключення, які виникають під час невірного вводу користувачем. Клас BusinessLogic здійснює пошук та повертає список поїздів згідно запиту; клас Input здійснює читання файлу і також приймає ввід користувача; клас Output здійснює вивід проміжного результату в файл; клас Entity містить інформацію про конкретний потяг; Time є допоміжним класом, який використовується для визначення поля departureTime в Entity. Наявні виключення – InvalidFileTypeException – недопустиме розширення файлу; InvalidFinalDestinationException – недопустиме значення пункту призначення; InvalidOptionException – недопустима опція; InvalidTimeException – недопустиме значення часу; NegativeNumberOfSits – число місць потягу є від’ємним.

Таблиця 1. Специфікація методів

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| № п/п | Назва класу | Назва методу | Призначення методу | Опис вхідних параметрів | Опис вихідних параметрів |
| 1 | Controller | run() | Виконання програми згідно обраної опції | - | - |
| 2 | Controller | Controller() | Конструктор класу | - | - |
| 3 | Controller | chooseLanguage() | Зміна основної мови за бажанням користувача | - | - |
| 4 | Controller | checkData() | Здійнює валідацію даних, отриманих з таблиці | data – вхідний масив даних, який первіряється | data – той самий масив даних, який повертається, якщо дані пройшли валідацію |
| 5 | Controller | getFileLocation() | Метод, який приймає на вхід шлях до файлу, перевіряє його та повертає в форматі, в якому його можуть обробляти інші методи | userFile – шлях до файлу | userFile.replace("\\","/") – видозмінений шлях до файлу |
| 6 | Controller | defineOption() | Вибір опції, згідно якої буде здійснено подальше виконання програми | - | Integer.parseInt(userOption) – опція, обрана користувачем, конвертована в змінну типу int |
| 7 | Controller | getResult() | Метод , який виводить результат запиту користувачеві | result – масив даних, який є результатом виконання запиту  option – рядок, згідно якого в консоль виводиться відповідна до запиту інформація | - |
| 8 | Controller | saveToFileOrNot() | Проміжний метод, в якому користувач вирішує, зберігати проміжні дані в файл чи ні | result – масив даних, який є результатом виконання запиту | - |
| 9 | Controller | defineFinalDestination() | Метод, в якому користувач здійснює ввід пункту призначення | - | destination – введений користувачем пункт призначення |
| 10 | Controller | defineTime() | Метод, в якому користувач здійснює ввід часу | - | new Time(time) – час, введений користувачем в форматі Time |
| 11 | Controller | printAllTrains() | Метод, який друкує усі наявні поїзди | list – список поїздів, отриманих з файлу | - |
| 12 | Converter | Converter() | Конструктор класу | list – вхідний набір даних для конвертації в String | - |
| 13 | Converter | toString() | Конветація даних в дані типу String для виводу в консоль | - | result1.toString() – отримані дані типу StringBuilder, конвертовані в String |
| 14 | Converter | trainInfo() | Дістає рядок потягів | trains – масив потягів, який потребує конвертації  size – розмір одного рядка з потягами  start – індекс елемента, з якого починається рядок потягів | getRow(trains, size, start, "TrainNumber") + "\n" + getRow(trains, size, start, "FinalDestination") + "\n" + getRow(trains, size, start, "DepartureTime") + "\n" + getRow(trains, size, start, "SitsNumber") – рядок потягів, конвертований в String |
| 15 | Converter | getRow() | Відображення необхідних компонентів | trains – масив потягів, який потребує конвертації  size – розмір одного рядка з потягами  start – індекс елемента, з якого починається рядок потягів  kindOfInfo – рядок, який відповідає назві атрибуту потягу | row – рядок, який містить інформацію про кілька потягів згідно kindOfInfo |
| 16 | DataManager | DataManager() | Конструктор класу | - | - |
| 17 | DataManager | loadDataFromFile() | Здійнює читання файла та поертає отримані дані | - | input.getFile(FILEPATH) – масив даних, отриманих з вказаного файлу |
| 18 | DataManager | writeDataToFile() | Здійснює запис проміжних даних, отриманих згідно запиту | location – шлях, де користувач бажає зберегти новий файл  result – дані, отримані згідно запиту | - |
| 19 | Validator | isNumeric() | Перевіряє, чи є вхідна змінна числом | strNum – змінна, яка перевіряється | Pattern.compile("-?\\d+([\\.\\d+)?").matcher(strNum).matches](file:///\\.\\d+)%3f%22).matcher(strNum).matches)() – булева змінна, яка повертає true, якщо змінна є числом, або false, якщо змінна не відповідає патерном і відповідно не є числом |
| 20 | Validator | hasDigitsOrSpecialCharacters() | Перевіряє, чи містить даний рядок цифри або спеціальні символи | str – рядок, який перевіряється | Pattern.compile("[0-9]+", Pattern.CASE\_INSENSITIVE).matcher(str).matches() – повертає true, якщо рядок не містить цифр, тобто відповідає патерну, або false, якщо рядок не відповідає патерну |
| 21 | Validator | isTimeFormat() | Перевіряє, чи відповідає рядок формату Time | time – рядок, який перевіряється | - |
| 22 | Validator | isPositive() | Перевіряє, чи є дане значення невід’ємним цілим числом | number – число, яке перевіряється | - |
| 23 | Validator | isCity() | Перевіряє, чи містить пункт призначення спеціальні символи/цифри | city – рядок, який перевіряється | - |
| 24 | Validator | checkOption() | Перевіряє, чи правильно було обрано опцію | option – опція, яка перевіряється | - |
| 25 | Validator | checkYesOrNoOptions() | Перевіряє, чи правильно було введено опцію «так» або «ні» | filename – ім’я файлу, яке перевіряється | - |
| 26 | Validator | checkLanguage() | Перевіряє формат файлу | language – значення мови, яке перевіряється | - |
| 27 | Entity | Entity() | Конструктор, за допомогою якого утворюється об’єкт на основі даних, отриманих з масиву | info - масив, який містить значення відповідних полів об’єкта класу | - |
| 28 | Entity | getSitsNumber() | Геттер для поля sitsNumber | - | sitsNumber – масив чисел, який містить к-сть загальних, плацкартних, купе- та СВ-місць. |
| 29 | Entity | getDepartureTime() | Геттер для поля departureTime | - | departureTime – час відправлення потяга |
| 30 | Entity | getFinalDestination() | Геттер для поля finalDestination | - | finalDestination – пункт призначення потяга |
| 31 | Entity | getTrainNumber() | Геттер для поля trainNumber | - | trainNumber – номер потягу |
| 32 | Time | Time() | Конструктор класу | time – рядок, який відповідає часу | - |
| 33 | Time | compareTo() | Порівнює два зазначені часи | time1 – перша позначка часу  time2 – друга позначка часу | Повертає true, якщо друга позначка часу більша за першу, і false в іншому випадку |
| 34 | Time | toString() | Повертає значення часу у вигляді рядка | - | Повертає значення часу типу String виду hours + ":" + minutes |
| 35 | InvalidFileTypeException | InvalidFileTypeException() | Конструктор ексепшену, який є дочірнім конструктором RuntimeException | message – повідомлення, яке отримується при throw exception | - |
| 36 | InvalidFinalDestinationException | InvalidFinalDestinationException() | Конструктор ексепшену, який є дочірнім конструктором RuntimeException | message – повідомлення, яке отримується при throw exception | - |
| 37 | InvalidOptionException | InvalidOptionException() | Конструктор ексепшену, який є дочірнім конструктором RuntimeException | message – повідомлення, яке отримується при throw exception | - |
| 38 | InvalidTimeException | InvalidTimeException() | Конструктор ексепшену, який є дочірнім конструктором RuntimeException | message – повідомлення, яке отримується при throw exception | - |
| 39 | NegativeNumberOfSitsException | NegativeNumberOfSitsException() | Конструктор ексепшену, який є дочірнім конструктором RuntimeException | message – повідомлення, яке отримується при throw exception | - |
| 40 | BusinessLogic | BusinessLogic() | Конструктор класу | - | - |
| 41 | BusinessLogic | getTrainsWithGeneralPlaces() | Метод, який знаходить в масиві потягів ті, які мають загальні місця | - | result – результат даного запиту |
| 42 | BusinessLogic | searchByPlaceAndTime() | Метод, який знаходить в масиві потягів ті, які прямують до заданого пункту призначення та відправляються після заданого часу | time – позначка часу, за якою шукаються потяги  finalDestination – пункт призначення, за яким здійснюється пошук | result – результат даного запиту |
| 43 | BusinessLogic | getList() | Геттер для поля list | - | list – масив, який містить всі наявні потяги |
| 44 | Input | Input() | Конструктор классу | - | - |
| 45 | Input | getFile() | Зчитує рядки з текстового файлу та повертає масив потягів | filepath – шлях, який використовується для читання файлу | Trains – масив Entity[], який є списком усіх наявних потягів |
| 46 | Input | getUserInput() | Зчитує інформацію, яку ввів користувач | - | input.nextLine() – рядок, введений користувачем |
| 47 | Output | writeToFile() | Здійснює запис проміжних результатів в файл | location – шлях, де має бути записано файл  result – масив потягів, отриманий згідно запиту | - |
| 48 | ProgramView | ProgramView() | Конструктор класу | - | - |
| 49 | ProgramView | show() | Метод, який показує меню | - | - |
| 50 | ProgramView | printError() | Метод, який друкує помилку | message – повідомлення, яке має бути надруковане | - |
| 51 | ProgramView | printMessage() | Метод, який друкує повідомлення | message – повідомлення, яке має бути надруковане | - |
| 52 | ProgramView | changeDefaultLanguage() | Метод, який змінює мову інтерфейсу | language – відмітка мови  country – відмітка країни | - |
| 53 | ProgramView | result() | Метод, який друкує результат згідно запиту | message – повідомлення, яке містить інформацію про запит | - |
| 54 | ResourceManager | ResourceManager() | Конструктор класу enum | - | - |
| 55 | ResourceManager | changeResource() | Змінює параметр properties, де містяться рядки згідно поточного locale | locale – змінна локалізації, яка визначає, які properties мають бути застосовані для інтерфейсу | - |

**РЕКОМЕНДАЦІЇ ЩОДО ВИКОРИСТАННЯ**

Технічні характеристики

Дане програмне забезпечення працює за наступних технічних умов:

* Windows 7, 8 або 10;
* JDK версії 8 або вище;
* Підтримка UTF-8 кодування(command prompt).

Інкструкція використання

1. Переконайтеся, що у вас на комп’ютері встановлено JDK версії 8 або вище. Для цього виконайте Пуск ->Пошук, введіть cmd та оберіть Command Prompt. В консолі пропишыть команду “java -version”.

2. Якщо було виведено повідомлення *’java’ is not recognized as internal command* або версія Java менша за 8, перейдіть за посиланням та завантажте файл: [jdk-14.0.1\_windows-x64\_bin.exe](https://www.oracle.com/java/technologies/javase-jdk14-downloads.html#license-lightbox) . Виконайте установку.

Якщо на комп’ютері встановлено Java версії 8+, переходьте до наступного пункту.

3. Завантажте архів з програмою за посиланням та розпакуйте в будь-якій зручній для вас директорії: <https://github.com/lamecode/JAVALabs_OOP/tree/new-branch/KURSACH>

4. Відкрийте директорію в Провіднику Файлів та перейдіть за шляхом out->artifacts->KURSACH\_jar.

5. Натисніть на поле, яке містить Ваше поточне положення в директорії, введіть cmd та натисніть Enter.

6. В консолі введіть команду *java -cp KURSACH.jar Main.*

7. Слідуйте вказівкам, які виводяться в консоль.

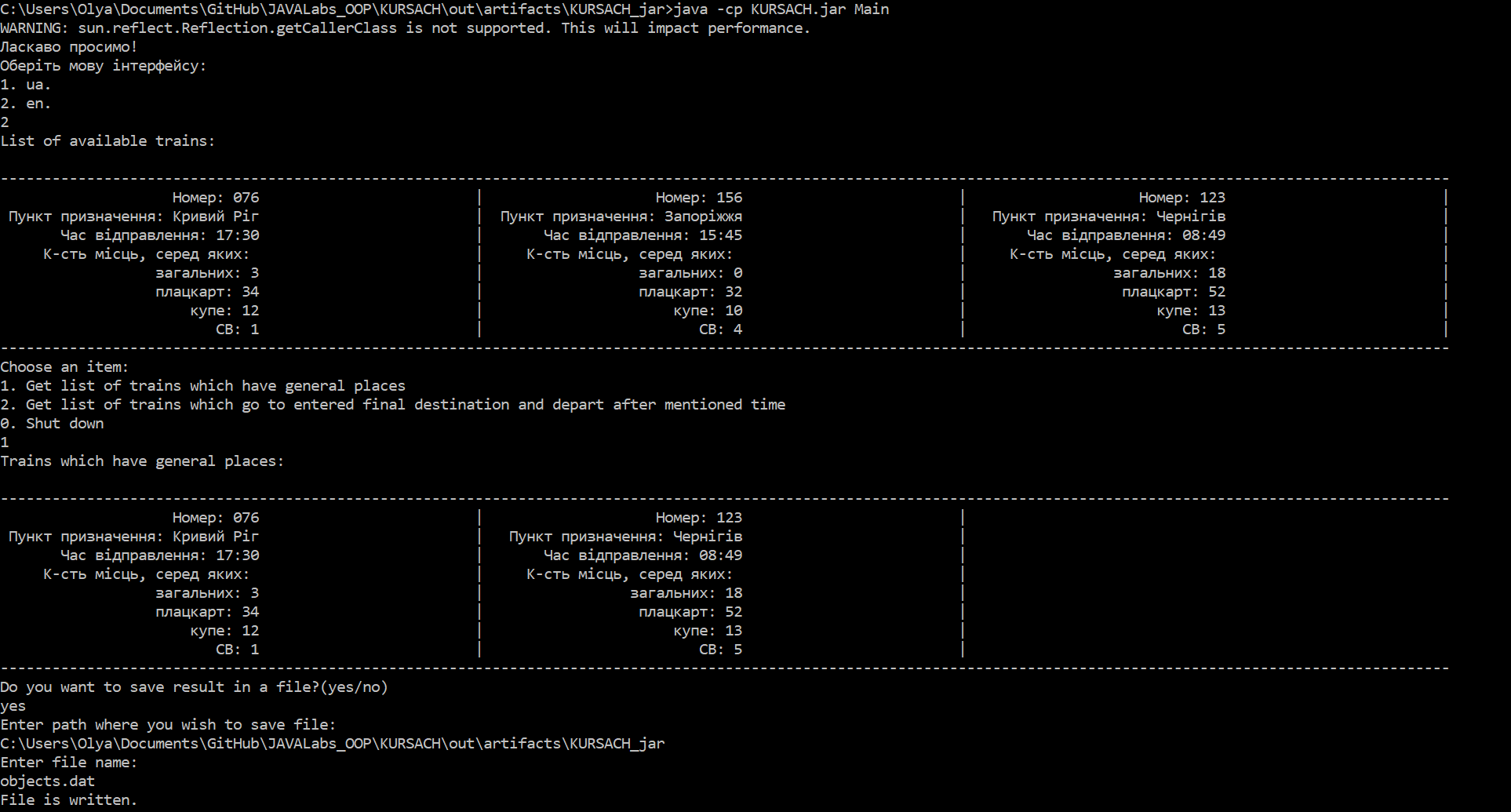


Рисунок 4. Приклад виконання програми

РЕКОМЕНДАЦІЇ КОРИСТУВАЧЕВІ

* Пункт призначення не має містити цифр або спеціальних символів.
* Час має бути введено в форматі “hh:mm”, де hh та mm – невід’ємні числа. К-сть годин має бути не більша за 24, а к-сть хвилин – не більша за 60.
* Дані записуються як у вигляді об’єктів, так і у вигляді тексту, а саме у форматах .dat та .csv відповідно, але читання здійснюється лише у випадку текстових файлів, тобто .csv.
* При введенні шляху до файлу, за яким бажано зберегти проміжні результати, потрібно спочатку ввести сам шлях(без назви файл), а потім назву файлу та його формат окремо.
* Додаток за замовчуванням має україномовну локалізацію, але за бажанням можна змінити її на англомовну.
* Зміна мови здійснюється лише один раз – під час запуску додатку.
* Під час запуску в консолі Windows бажано змінити кодування на chcp 855, щоб було відображено всі україномовні символи.

**ВИХІДНИЙ КОД ВИКОРИСТАННЯ**

<https://github.com/lamecode/JAVALabs_OOP/tree/new-branch/KURSACH>

**СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ**

1. Level (Apache Log4j 1.2.17 API). <https://logging.apache.org/log4j/1.2/apidocs/org/apache/log4j/Level.html>

2. Java RegEx meta character (.) and ordinary dot? - Stack Overflow. <https://stackoverflow.com/questions/3674930/java-regex-meta-character-and-ordinary-dot>

3. Guide to Character Encoding. <https://www.baeldung.com/java-char-encoding>

4. java - How to get the default ResourceBundle regardless of current default Locale - Stack Overflow. <https://stackoverflow.com/questions/24305512/how-to-get-the-default-resourcebundle-regardless-of-current-default-locale/24325833>

5. java - log4j debug messages not showing in console despite isDebugEnabled being true - Stack Overflow. <https://stackoverflow.com/questions/15086818/log4j-debug-messages-not-showing-in-console-despite-isdebugenabled-being-true>

6. Log4j – Log4j 2 Appenders. <https://logging.apache.org/log4j/2.x/manual/appenders.html#RollingFileAppender>

7. How to configure log4j as logging mechanism in Java. <https://www.codejava.net/coding/how-to-configure-log4j-as-logging-mechanism-in-java#BasicConfiguration>

8. <https://docs.oracle.com/javase/7/docs/api/java/util/Locale.html>

9. <https://docs.oracle.com/javase/7/docs/api/java/util/logging/Logger.html#:~:text=A%20Logger%20object%20is%20used,as%20java.net%20or%20javax.>