# НАЦІОНАЛЬНЫЙ 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

# Варіант № 2

Виконав: студент групи ІП-82

Бебех О. О.

Прийняла: доцент Іванова Л.М.

Київ – 2020

**Зміст**

1.Завдання на курсову роботу………………………………………………………………3

2. Опис застосування………………………………………………………………………..4

2.1 Архітектура застосування………………………………………………………5

2.2 Діаграма варіантів використання………………………………………………8

2.3 Діаграма класів та таблиці……………………………………………………...9

3. Рекомендації щодо використання застосування………………………………………13

3.1 Технічна характеристика…………………………………………………………

3.2 Рекомендації по встановленню та налаштуванню……………………………

3.3 Рекомендації користувачеві……………………………………………………

4. Вихідний код застосування ……………………………………………………………

5. Список літератури, що використовувалася…………………………………………………

**1. Завдання на курсову роботу**

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.

Згідно 2-го варіанту застосування повинне мати наступні характеристики:

1. **Покупець**:

- Прізвище, Ім’я, По батькові, Адреса, Номер кредитної картки, Номер банківського рахунку.

- Конструктор, Методи доступу, Метод *toString()*

2. Отримати список покупців у алфавітному порядку.

3. Отримати список покупців, у яких номер кредитної картки знаходиться в заданому діапазоні.

**2. Опис застосування**

Консольна програма - це комп'ютерна програма, призначена для використання через текстовий комп'ютерний інтерфейс, наприклад, текстовий термінал, інтерфейс командного рядка деяких операційних систем (Unix, DOS тощо) або текстовий інтерфейс, що входить до більшості Операційні системи графічного інтерфейсу користувача (GUI), такі як консоль Win32 в Microsoft Windows, термінал у macOS та xterm в Unix. Користувач, як правило, взаємодіє з консольним додатком, використовуючи лише клавіатуру та екран дисплея, на відміну від додатків GUI, які зазвичай вимагають використання миші або іншого вказівного пристрою. Багато консольних додатків, таких як інтерпретатори командного рядка, є інструментами командного рядка, але існують також численні програми на основі текстового інтерфейсу (TUI).

Оскільки швидкість та простота використання додатків графічного інтерфейсу покращилися з часом, використання консольних додатків значно зменшилось, але не зникло. Деякі користувачі просто віддають перевагу консольним програмам, тоді як деякі організації все ще покладаються на існуючі консольні програми для обробки ключових завдань з обробки даних.

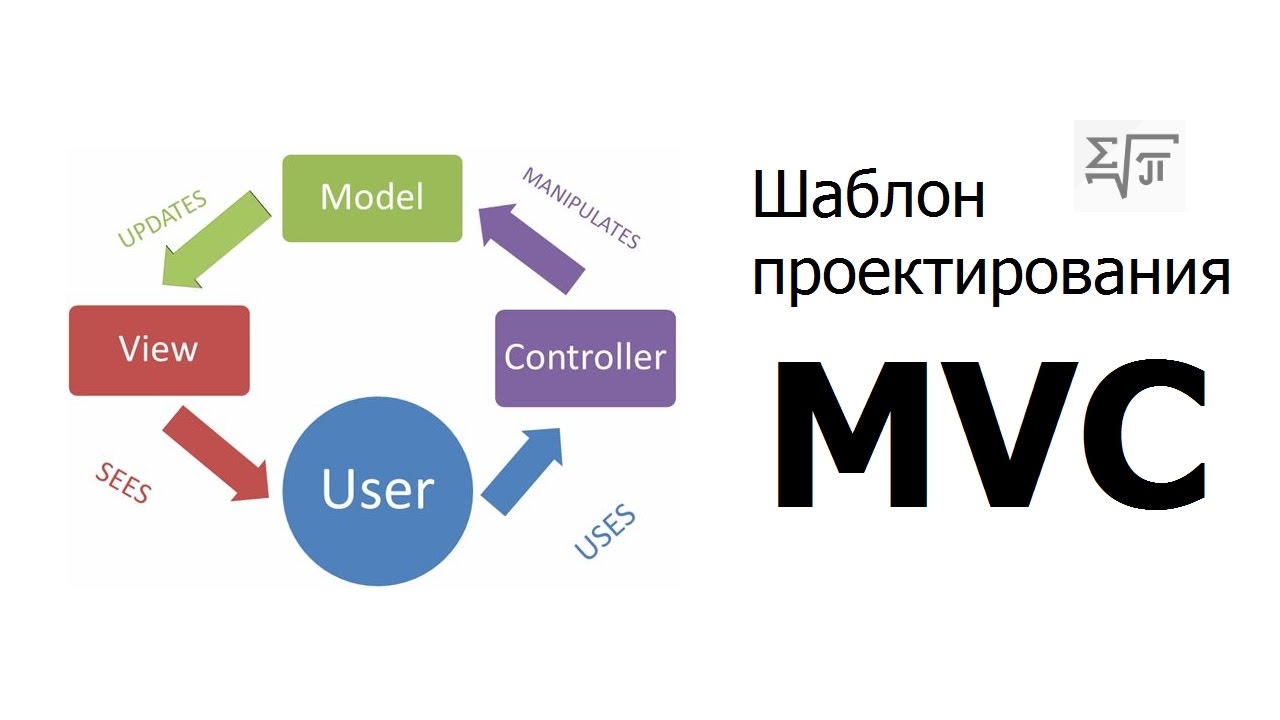
Можливість створення консольних додатків зберігається як особливість сучасних середовищ програмування, таких як Visual Studio та .NET Framework в Microsoft Windows, оскільки це значно спрощує процес навчання нової мови програмування, усуваючи складність графічного інтерфейсу користувача.**2.1 Архітектура застосування**

Як основу для архітектури даного застосунку була взята модель **MVC** або Модель–представлення–контролер. Цей шаблон передбачає поділ програми на три компоненти, що взаємодіють між собою, а саме: модель даних, інтерфейс та модуль керування. Мета шаблону — гнучкий дизайн програмного забезпечення, який повинен полегшувати подальші зміни чи розширення програм, а також надавати можливість повторного використання окремих компонентів програми. Крім того використання цього шаблону у великих системах сприяє впорядкованості їхньої структури і робить їх більш зрозумілими за рахунок зменшення складності.

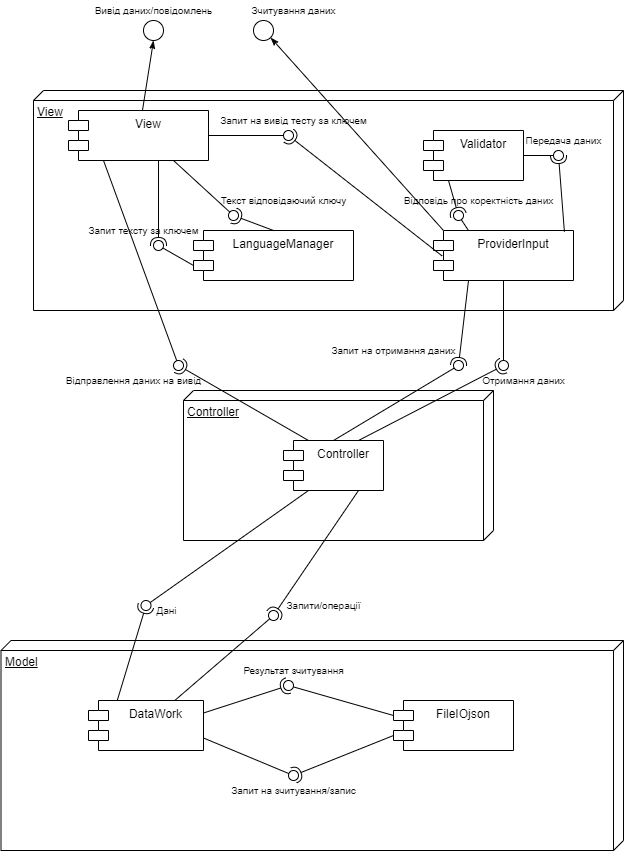
Кожен компонент виконує свою роль:

* Модель є центральним компонентом шаблону MVC і відображає поведінку застосунку, незалежну від інтерфейсу користувача. Модель стосується прямого керування даними, логікою та правилами застосунку.
* Вигляд може являти собою будь-яке представлення інформації, одержуване на виході, наприклад графік чи діаграму. Одночасно можуть співіснувати кілька виглядів (представлень) однієї і тієї ж інформації, наприклад гістограма для керівництва компанії й таблиці для бухгалтерії.
* Контролер одержує вхідні дані й перетворює їх на команди для моделі чи вигляду.

У мові програмування Java концепція MVC підтримується на рівні стандартних класів-бібліотек. В результаті використання парадигми MVC програміст отримує в своє розпорядження могутню структуру об'єктів-компонентів, функції яких чітко розмежовані, що гарантує надійність і розширюваність системи, що розробляється.



**Рис 1. – Вигляд та взаємодія моделі MVC**

****

**Рис 2. – Діаграма компонентів**

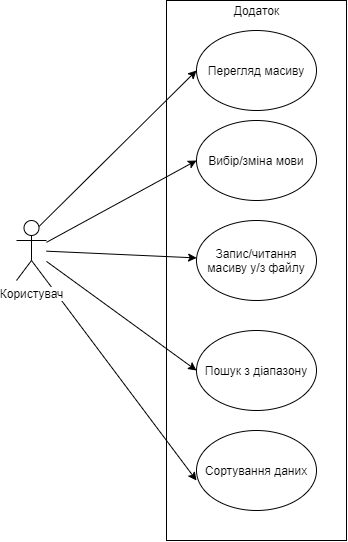
Ця діаграма показує нам поділ на модель даних(Model), інтерфейс(View) та модуль керування(Controller). Також вона показує взаємодію частин як між собою так і на внутрішньому рівні.

Модель даних (Model) виконує всі операції пов’язані із масивом покупців, записом у файли та зміни розмірності. Ця частина складається з двох основних компонент: DataWork та FileIOjson. DataWork зберігає у собі масив покупців, виконує дії про які повідомляє та виконує операції над масивом, серед яких є операції виклику наступного класу, а саме FileIOjson. FileIOjson у свою чергу зчитує дані із файлу, формату json та переводить дані в масив покупців. Також він переводить масив покупців у формат json та зберігає до файлу.

Інтерфейс (View) виводить користувачеві меню та список покупців у табличому вигляді, також приймає та валідує введені дані. Складається із 4-х основних компонентів: View, Validator, ProviderInput та LanguageManager. View виводить список покупців у табличому, повідомлення та меню. Передає ключі до LanguageManager. Щоб вивести текст відповідно до обраної мови. Validator перевіряє коректність введених даних користувачем. ProviderInput приймає дані від користувача та відправляє їх на перевірку вValidator. LanguageManager отримує ключі від View та надає відповідні їм значення згідно обраної локалізації.

Модуль керування (Controller) приймає відповідь користувача та надсилає моделі необхідні дії для виконання. Також вказує що потрібно вивести у відповідності із станом програми.

**2.2 Діаграма варіантів використання**

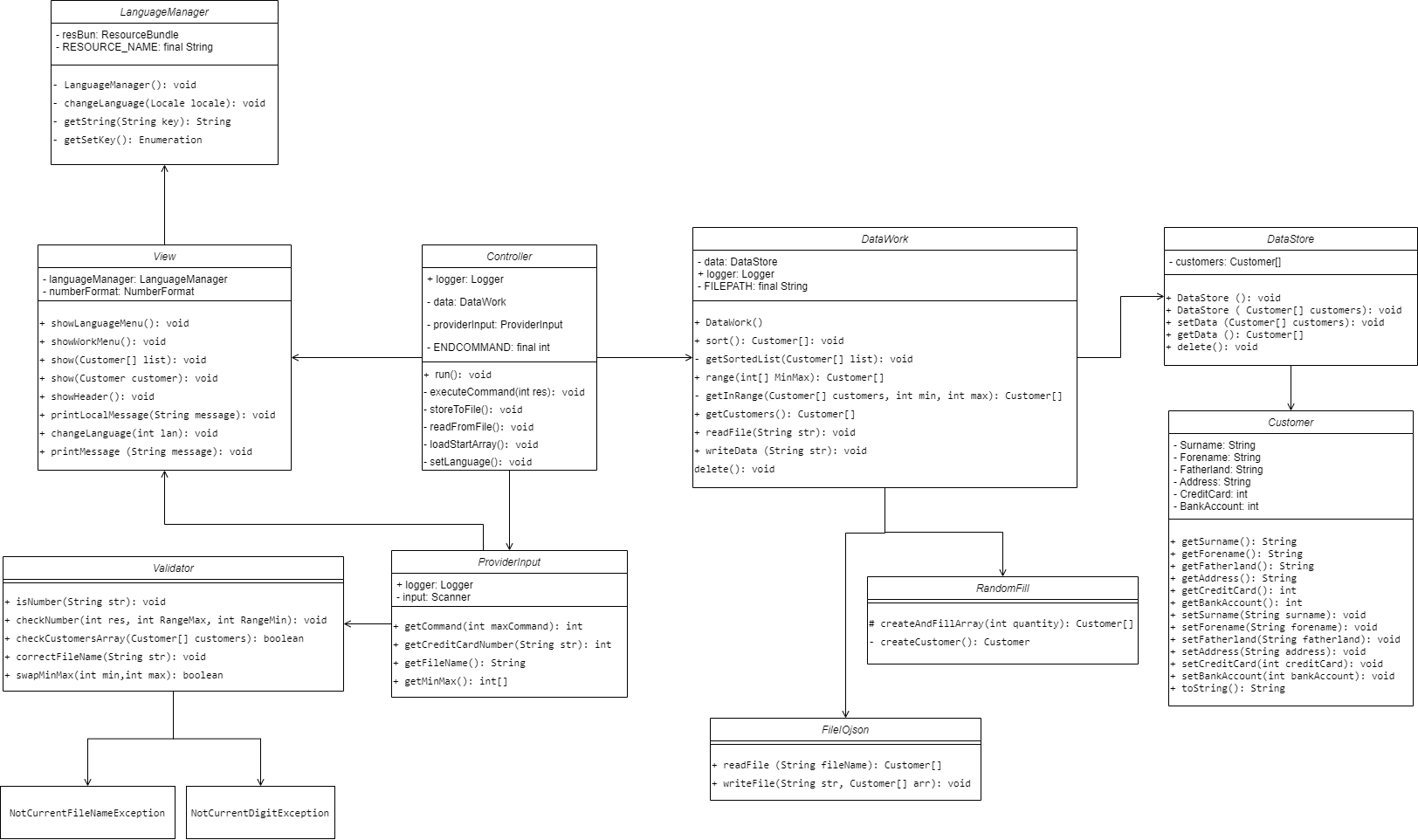
****

**Рис 3. – діаграма варіантів використання**

На цій діаграмі показано дії, які користувач може виконувати у системі, такі як:

* Перегляд вмісту масиву;
* Вибір мови на початку програми;
* Зміна мови;
* Запис у файл масиву покупців;
* Читання з файлу масиву покупців;
* Пошук користувачів банківські карти яких входять у діапазон;
* Сортування масиву покупців.

**2.3 Діаграма класів та таблиці**

****

**Рис 4 – діаграма класів**

Тут показані усі класи та їх взаємодія, також їх поля та методи.

**Таблиця 1 – функції**

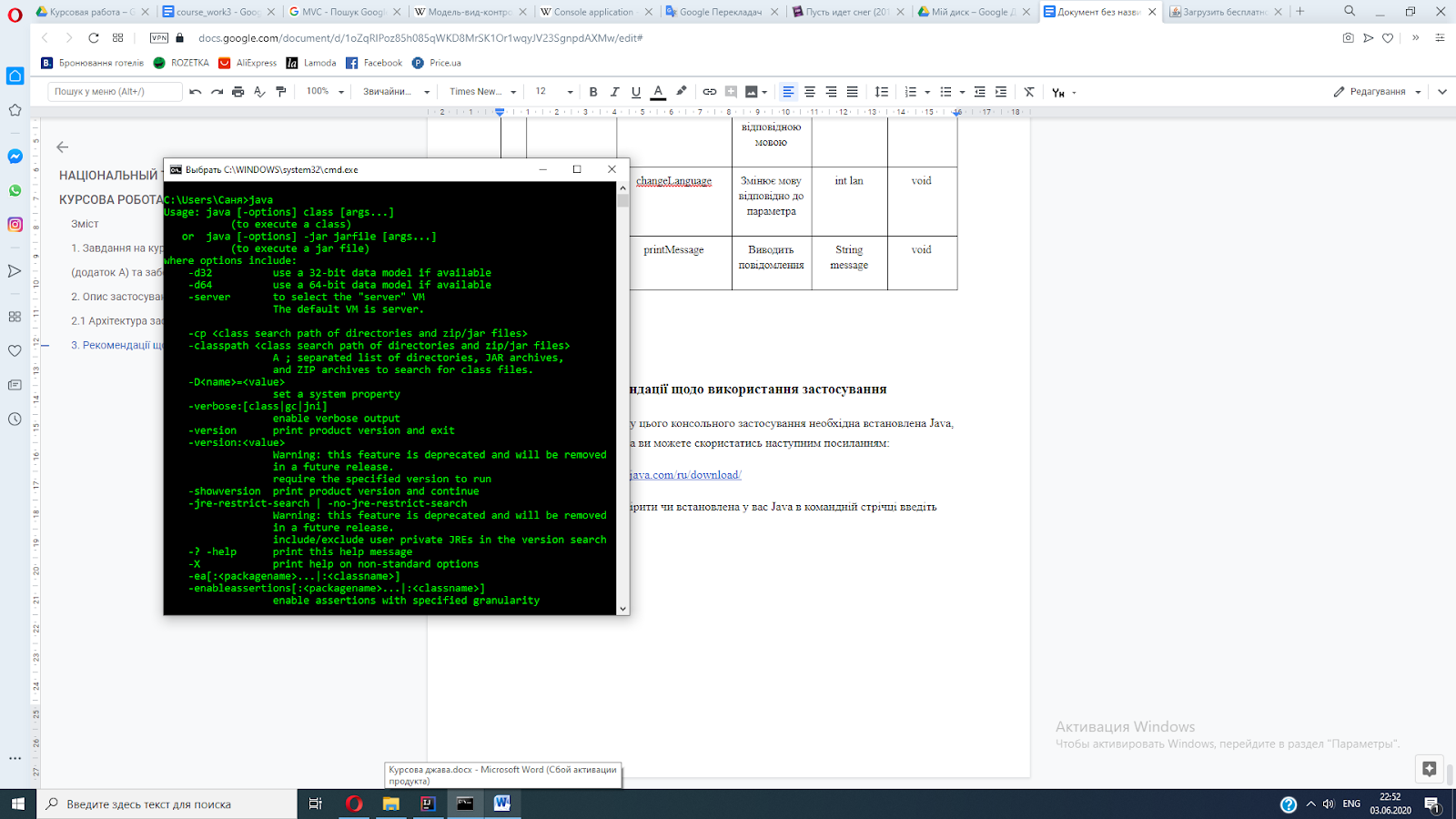
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| № п/п | Назва класу | Назва функції | Призначення функції | Опис вхідних параметрів | Опис вихідних параметрів |
| 1 | Main | main | Стартова точка запуску програми | String[] | void |
| 2 | Controller | run | Надає вибір мови, пунктів меню та отримує від користувача команди | - | void |
| 3 | Controller | executeWorkCommand | Обробляє команду та виконує відповідні дії | int res | void |
| 4 | Controller | storeToFile | Отримує назву файлу та викликає команду на запис у файл з відповідним іменем | - | void |
| 5 | Controller | readFromFile | Отримує назву файлу та зчитує масив з даного файлу | - | void |
| 6 | Controller | loadStartArray | Зчитує масив з початкового файлу | - | void |
| 7 | Controller | setLanguage | Викликає команди для виводу меню мов та вибору мови | - | void |
| 8 | Customer | Customer | Створення покупця з заданими параметрами | String surname, String forename, String fatherland, String address,  int creditCard,  int bankAccount | void |
| 9 | Customer | getSurname | Повертає фамілію покупця | - | String surname |
| 10 | Customer | getForename | Повертає ім’я покупця | - | String forename |
| 11 | Customer | getFatherland | Повертає по-батькові покупця | - | String fatherland |
| 12 | Customer | getAddress | Повертає адрес покупця | - | String address |
| 13 | Customer | getCreditCard | Повертає кредитну картку покупця | - | int creditCard |
| 14 | Customer | getBankAccount | Повертає банківський рахунок покупця | - | int bankAccount |
| 15 | Customer | setSurname | Задає фамілію покупця | String surname | void |
| 16 | Customer | setForename | Задає ім’я покупця | String forename | void |
| 17 | Customer | setFatherland | Задає по-батькові покупця | String fatherland | void |
| 18 | Customer | setAddress | Задає адрес покупця | String address | void |
| 19 | Customer | setCreditCard | Задає кредитну картку покупця | int creditCard | void |
| 20 | Customer | getBankAccount | Задає банківський рахунок покупця | int bankAccount | void |
| 21 | Customer | toString | Створює та повертає строку із параметрів користувача | - | String |
| 22 | DataStore | DataStore | Створює об’єкт класу та задає масиву покупців значення null | - | void |
| 23 | DataStore | DataStore | Створює об’єкт класу та задає масиву покупців значення переданого массиву | Customer[] customers | void |
| 24 | DataStore | setData | Задає масив покупців згідно переданого | Customer[] customers | void |
| 25 | DataStore | getData | Повертає масив покупців | - | Customer[] |
| 26 | DataStore | delete | Задає масиву покупців значення null | - | void |
| 27 | DataWork | DataWork | Створює об’єкт класу та зчитує з файлу початковий масив покупців та присвоює це значення полю data | - | void |
| 28 | DataWork | sort | Повертає відсортований масив | - | Customer[] |
| 29 | DataWork | getSortedList | Отримує масив та сортує його | Customer[] | Customer[] |
| 30 | DataWork | range | Перевіряє чи пустий масив і в протилежному випадку повертає масив покупців у заданому діапазоні | int[] MinMax | Customer[] |
| 31 | DataWork | getInRange | Приймає діапазон та масив, формує масив із покупців у заданому діапазоні, записує цей масив у поле data | Customer[] customers,  int min, int max | Customer[] |
| 32 | DataWork | getCustomers | Повертає масив покупців з поля data | - | Customer[] |
| 33 | DataWork | readFile | Записує масив у поле data із зчитаного файлу | String str | void |
| 34 | DataWork | writeData | Записує масив з поля data у файл зі заданим шляхом | String str | void |
| 35 | DataWork | delete | Видаляє данні з поля data | - | void |
| 36 | RandomFill | createAndFillArray | Створює масив покупців заданої довжини | int quantity | Customer[] |
| 37 | RandomFill | createCustomer | Створює та повертає випадково згенерованого покупця із параметрами із пулу | - | Customer |
| 38 | FileIOjson | readFile | Зчитує файл записаний у форматі json та повертає масив покупців | String fileName | Customer[] |
| 39 | FileIOjson | writeFile | Переводить масив покупців у формат json та записує його у файл із заданим ім’ям | String str, Customer[] arr | void |
| 40 | LanguageManager | LanguageManager | Ініціалізує початкові налаштування мови | - | void |
| 41 | LanguageManager | changeLanguage | Змінює локалізацію об’єкта | Locale locale | void |
| 42 | LanguageManager | getString | Повертає строку відповідно до локалізації | String key | String |
| 43 | LanguageManager | getSetKey | Повертає перелік ключів | - | Enumeration |
| 44 | ProviderInput | getCommand | Приймає нову відповідь від користувача до тих пір поки не отримає натуральне число не більше заданого | int maxCommand | int |
| 45 | ProviderInput | getCreditCardNumber | Показує отримане повідомлення  Приймає нову відповідь від користувача до тих пір поки не отримає натуральне число у діапазоні від 0 до 1000000 | String str | int |
| 46 | ProviderInput | getFileName | Отримує від користувача нову відповідь поки він не введе коректну назву файлу | - | String |
| 47 | ProviderInput | getMinMax | Отримує відповідь та повертає масив з двох чисел | - | int[] |
| 48 | Validator | isNumber | Перевіряє чи строка є числом | String str | void |
| 49 | Validator | checkNumber | Перевіряє чи число входить в діапазон | int res, int RangeMax, int RangeMin | void |
| 50 | Validator | checkCustomersArray | Перевіряє чи пустий масив | Customer[] customers | bool |
| 51 | Validator | correctFileName | Перевіряє розширення файлу | String str | - |
| 52 | Validator | swapMinMax | Перевіряє чи у правильному порядку розташовані мінімальне та максимальне значення | int min,int max | bool |
| 53 | View | showLanguageMenu | Виводить меню з варіантами зміни мови | - | void |
| 54 | View | showWorkMenu | Показує основне меню | - | void |
| 55 | View | show | Виводить масив Покупців у табличному вигляді | Customer[] customer | void |
| 56 | View | show | Виводить покупця | Customer customer | void |
| 57 | View | showHeader | Виводить назви стовпців таблиці |  | void |
| 58 | View | printLocalMessage | Виводить повідомлення відповідною мовою | String message | void |
| 59 | View | changeLanguage | Змінює мову відповідно до параметра | int lan | void |
| 60 | View | printMessage | Виводить повідомлення | String message | void |

**3. Рекомендації щодо використання застосування**

Для запуску цього консольного застосування необхідна встановлена Java, якщо вона не встановлена ви можете скористатись наступним посиланням:

* <https://www.java.com/ru/download/>

Щоб перевірити чи встановлена у вас Java в командній стрічці введіть java або java –version.



**Рис 5. - вивід при виконанні команди java**