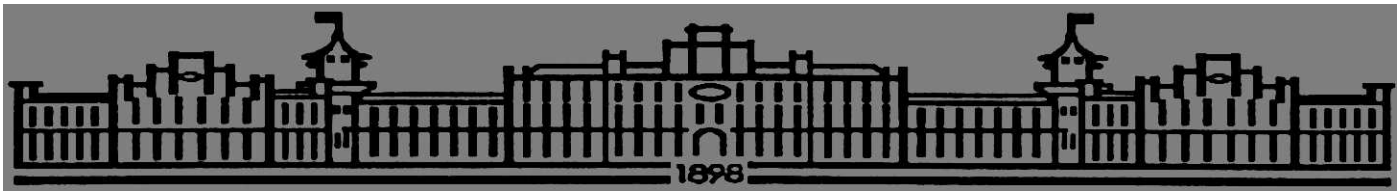

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
ІМЕНІ ІГОРЯ СІКОРСЬКОГО”

Факультет прикладної математики
Кафедра програмного забезпечення комп’ютерних систем

**Лабораторна робота № __3___**

з дисципліни “Основи програмування 2”
тема “Модульне програмування і контракти”

Виконав
студент І курсу
групи КП-03

Сидоренко Олександр Олександрович
(*прізвище, ім’я, по батькові*)

Перевірив
“____” “_____” 20___ р.
викладач

Гадиняк Руслан Анатолійович
(*прізвище, ім’я, по батькові*)

варіант № 16

Київ 2021

## Мета роботи

Виконати розділення коду програми на модулі.

Використати контракти (інтерфейси) для розділення коду клієнта та реалізації та забезпечення змінності реалізації.

## Постановка завдання

Програма дозволяє користувачу виконувати через консоль операції над двома множинами цілих чисел A і B. На початку роботи програми обидві множини порожні.

***Модуль командного інтерфейсу***

Реалізувати **модуль командного інтерфейсу користувача**, через який користувач задає текстову команду у консолі і отримує результат її виконання або опис помилки:

- Команда `**{set} add {value}**` додає значення {value} у множину {set} і виводить результат додавання (true/false). Приклад: `**a add 13**`, `**b add -200**`
- Команда `**{set} contains {value}**` перевіряє чи значення {value} є у множині {set} і виводить результат (true/false). Приклад: `**a contains 13**`, `**b contains -200**`
- Команда `**{set} remove {value}**` видаляє значення {value} з множини {set} і виводить результат видалення (true/false). Приклад: `**a remove 13**`, `**b remove -200**`
- Команда `**{set} clear**` очищує множину {set} (робить її порожньою). Приклад: `**a clear**`, `**b clear**`
- Команда `**{set} log**` виводить числа з множини {set}.

Приклад: `**a log**`, `**b log**`

- Команда `**{set} count**` виводить кількість елементів у множині {set}. Приклад: `**a count**`, `**b count**`
- Команда `**{set} read {file}**` читає з файлу {file} унікальні числа у множину {set} (спосіб запису чисел у файлі за варіантом з **додатку В**). Файли можна попередньо генерувати випадковим чином або записати вручну. Приклади: `**a read ./file1.txt**`, `**b read ./b.txt**`
- Команда `**{set} write {file}**` записує у файл {file} числа з множини {set} спосіб запису чисел у файлі за варіантом з **додатку В**). Приклади: `**a write ./out.txt**`, `**b write ./x.txt**`
- Команда `**{operation}**` виконує над множинами A і B одну із операцій за варіантом (див. **додаток А**) і виводить результат. Множина A - перший операнд, множина B - другий операнд. Приклад: `**IntersectWith**`, `**Overlaps**`

Користувач вводить команди у циклі. Можна перервати цикл спеціальною командою або порожньою командою.

Виникнення будь-якої помилки перехоплювати і виводити, користувач продовжує працювати з програмою.

***Модуль логування***

Весь вивід результатів роботи програми та повідомлень про помилки виконувати через обраний **модуль логування**.

Реалізувати два модулі логування для різних способів логування повідомлень: перший - у консоль, другий за варіантом (див. **додаток С**).

Модуль логування обирається через перший аргумент командного рядка. Приклади запуску програми: `**dotnet run console**` або `**dotnet run csv ./file.csv**`

За замовчуванням (якщо не задано аргументу командного рядка) використовувати модуль логування у консоль.

# Приклади результатів

```
Command list:
{set} add {value}
{set} contains {value}
{set} remove {value}
{set} clear
{set} log
{set} count
{set} read {file}
{set} write {file}
SetEquals
UnionWith
exit

Enter command:
```

| |
|---|
| a add 1 |
| Number `1` was successfully added to the set `a`! |
| a contains 1 |
| Set `a` contains number `1`! |
| a remove 1 |
| Number `1` was successfully removed from the set `a`! |
| a read ./setToRead |
| All values from the file `./setToRead` were successfully added into set `a`! |
| a log |
| Set `a`: 1 3 4 5 6 7 8 9 10 11 12 13 15 |
| a count |
| Number of values in set `a`: 13 |
| b add 100 |
| Number `100` was successfully added to the set `b`! |
| b add 2 |
| Number `2` was successfully added to the set `b`! |
| b add 200 |
| Number `200` was successfully added to the set `b`! |
| b log |
| Set `b`: 2 100 200 |
| UnionWith |
| Sets were successfully united! |

| a log |
| --- |
| Set `a`: 1 2 3 4 5 6 7 8 9 10 11 12 13 15 100 200 |
| SetEquals |
| Sets aren't equal! |
| a read ./setToRead |
| All values from the file `./setToRead` were successfully added into set `a`! |
| b read ./setToRead |
| All values from the file `./setToRead` were successfully added into set `b`! |
| SetEquals |
| Sets are equal! |
| a write ./filePath |
| Set `a` was written to file `./filePath`! |

*Помилки:*

| hello |
| --- |
| ERROR! Incorrect command: `hello`! |
| a add 1s |
| ERROR! 3rd argument shuld be an integer! |
| c add 5 |
| ERROR! It is non-existing set: `c`! Enter `a` or `b`! |
| a adds 5 |
| ERROR! 2nd arg should be `add`, but not `adds`! |
| a add 2 1 |
| ERROR! Incorrect number of command args! |
| a read ./file |
| ERROR! Incorrect file path or there are mistakes in the file! |
| alex@alex-Mi-Gaming-Laptop-15-6:~/projects/progbase2/lab3$ dotnet run file /home/alex/projects/progbase2/lab3/fileLogger/ -10 |
| Unhandled exception. System.Exception: ERROR! Max number of lines in file is positive integer! |
| alex@alex-Mi-Gaming-Laptop-15-6:~/projects/progbase2/lab3$ dotnet run file /home/alex/projects/progbase2/lab3/fileLoer/ 10 |
| Unhandled exception. System.Exception: ERROR! It is non-existing directory: `/home/alex/projects/progbase2/lab3/fileLoer/`! |
| alex@alex-Mi-Gaming-Laptop-15-6:~/projects/progbase2/lab3$ dotnet run ffile |

| |
|---|
| /home/alex/projects/progbase2/lab3/fileLogger/ 10 |
| `Unhandled exception. System.Exception: ERROR! 2nd argument should be `file`, but have `ffile`` |
| alex@alex-Mi-Gaming-Laptop-15-6:~/projects/progbase2/lab3$ dotnet run file /home/alex/projects/progbase2/lab3/fileLogger/ 10 10 |
| `Unhandled exception. System.Exception: ERROR! Incorrect number of command line arguments! Should be `1` or `3`, but have `4`` |

## Висновки

Під час виконання лабораторної роботи я навчився розділювати код програми на модулі. Також використав інтерфейси для розділення коду клієнта та реалізації та забезпечення змінності реалізації.