Міністерство освіти і науки України

Національний університет «Львівська політехніка»

Кафедра «Електронних обчислювальних машин»



**Звіт**

з лабораторної роботи № 6

**з дисципліни:** «Кросплатформенні засоби програмування»

**на тему:** «параметризоване програмування»

**Виконав:**

студент групи *КІ-306*

*Ярмола Юрій*

**Прийняв:**

доцент кафедри ЕОМ

Іванов Ю. С.

**Львів – 2023**

**Мета роботи:** оволодіти навиками параметризованого програмування мовою Java.

**Завдання ( Варіант 30 «Множина»):**

1. Створити параметризований клас, що реалізує предметну область задану варіантом.

Клас має містити мінімум 4 методи опрацювання даних включаючи розміщення та виймання елементів. Парні варіанти реалізують пошук мінімального елементу, непарні –максимального. Написати на мові Java та налагодити програму-драйвер для розробленого класу, яка мстить мінімум 2 різні класи екземпляри яких розмішуються у екземплярі розробленого класу-контейнеру. Програма має розміщуватися в пакеті група.Прізвище.Lab6 та володіти коментарями, які дозволять автоматично згенерувати документацію до розробленого пакету.

2. Автоматично згенерувати документацію до розробленого пакету.

3. Скласти звіт про виконану роботу з приведенням тексту програми, результату її виконання та фрагменту згенерованої документації.

4. Дати відповідь на контрольні запитання.

**Вихідний код програми**

SetContainer.java

package KI306.Yarmola.Lab6;

import java.util.ArrayList;

import java.util.Collections;

import java.util.List;

/\*\*

\* Клас для зберігання та опрацювання множини елементів, які реалізують інтерфейс Comparable.

\*

\* @param <T> Тип елементів множини, які реалізують інтерфейс Comparable.

\*/

public class SetContainer<T extends Comparable<T>> {

private List<T> elements = new ArrayList<>();

/\*\*

\* Додає елементи до множини, якщо вони ще не містяться в ній.

\*

\* @param elements\_list Масив елементів для додавання до множини.

\*/

public void union(T[] elements\_list) {

for (T element : elements\_list) {

if (!elements.contains(element)) {

elements.add(element);

}

}

}

/\*\*

\* Додає елемент до множини, якщо він ще не міститься в ній.

\*

\* @param element Елемент для додавання до множини.

\*/

public void addElement(T element) {

if (!elements.contains(element)) {

elements.add(element);

}

}

/\*\*

\* Видаляє елемент із множини.

\*

\* @param element Елемент для видалення із множини.

\*/

public void removeElement(T element) {

elements.remove(element);

}

/\*\*

\* Знаходить мінімальний елемент у множині.

\*

\* @return Мінімальний елемент у множині або null, якщо множина порожня.

\*/

public T findMin() {

if (elements.isEmpty()) {

return null;

}

return Collections.min(elements);

}

/\*\*

\* Отримує всі елементи множини.

\*

\* @return Список всіх елементів множини.

\*/

public List<T> getElements() {

return elements;

}

/\*\*

\* Очищає множину, видаляючи всі елементи.

\*/

public void clear() {

elements.clear();

}

}

SetDriver.java

import KI306.Yarmola.Lab6.SetContainer;

/\*\*

\* Клас для демонстрації використання класу SetContainer.

\* @author Yura Yarmola

\* @version 1.0

\*

\*/

public class SetDriver {

/\*\*

\* Головний метод для запуску програми.

\*

\* @param args Аргументи командного рядка (не використовуються в цій програмі).

\*/

public static void main(String[] args) {

// Створюємо множину для цілих чисел і виконуємо декілька операцій з нею

SetContainer<Integer> integerSet = new SetContainer<>();

integerSet.addElement(5);

integerSet.addElement(10);

integerSet.addElement(3);

integerSet.removeElement(5);

Integer[] list = new Integer[] { 1, 3, 23, 4, 3 };

integerSet.union(list);

System.out.println("Перелік елементів: " + integerSet.getElements());

System.out.println("Мінімальний елемент: " + integerSet.findMin());

// Створюємо множину для рядків і виконуємо декілька операцій з нею

SetContainer<String> stringSet = new SetContainer<>();

stringSet.addElement("apple");

stringSet.addElement("banana");

stringSet.addElement("cherry");

System.out.println("Мінімальний елемент: " + stringSet.findMin());

stringSet.clear();

}

}

**Результат виконання програми**

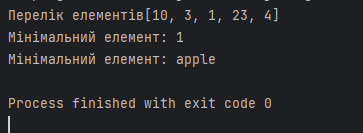
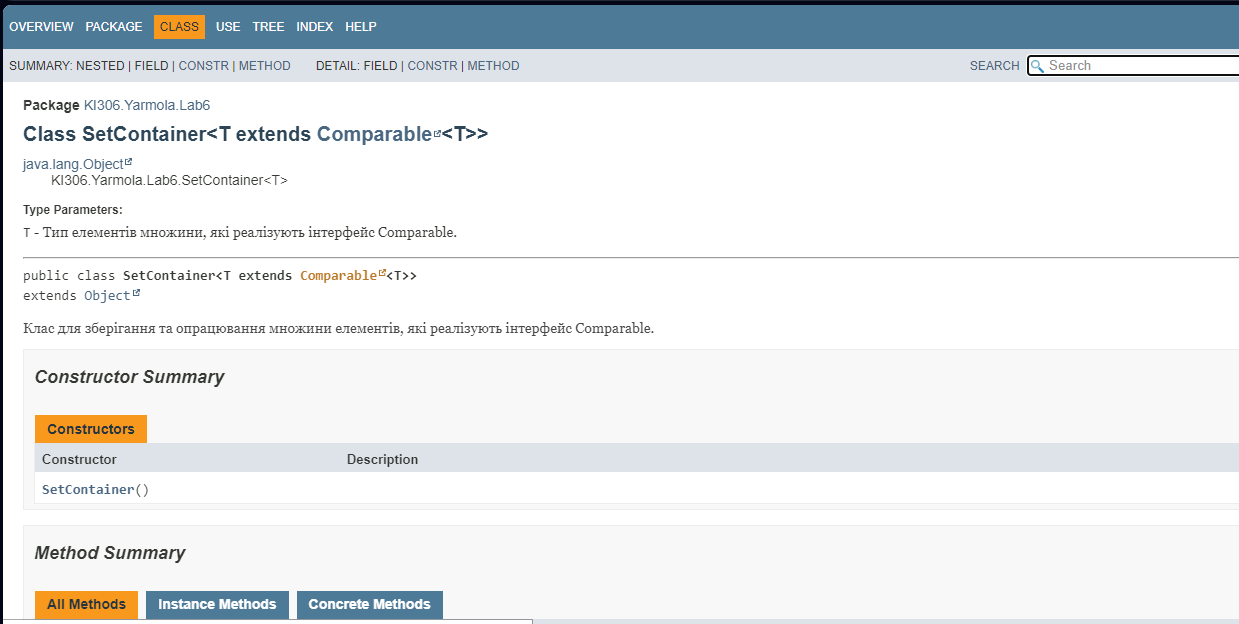
****

Рис.1 – Скріншот виконання програми

**Фрагмент згенерованої документації**



**Відповіді на контрольні запитання**

1. Дайте визначення терміну «параметризоване програмування».

Параметризоване програмування - це підхід до програмування, де класи та методи можуть приймати типи даних як параметри, що дозволяє створювати більш загальні та змінні структури даних та методи.

1. Розкрийте синтаксис визначення простого параметризованого класу.

class MyGenericClass<T> {

// Код класу з використанням параметра T

}

1. Розкрийте синтаксис створення об’єкту параметризованого класу.

MyGenericClass<Integer> obj = new MyGenericClass<>();

1. Розкрийте синтаксис визначення параметризованого методу.

<T> void myGenericMethod(T value) {

// Код методу з використанням параметра T

}

1. Розкрийте синтаксис виклику параметризованого методу.

MyClass.<Integer>myGenericMethod(42); // Виклик методу з параметром Integer

1. Яку роль відіграє встановлення обмежень для змінних типів?

Встановлення обмежень для змінних типів дозволяє обмежити допустимий тип даних, який можна використовувати як параметр типу.

1. Як встановити обмеження для змінних типів?

<T extends Number> void myMethod(T value) { /\* ... \*/ }

1. Розкрийте правила спадкування параметризованих типів.

Правила спадкування параметризованих типів включають здатність успадковувати параметризовані класи та методи з інтерфейсів або суперкласів та можливість використовувати абстрактні типи даних.

1. Яке призначення підстановочних типів?

Підстановочні типи (wildcards) використовуються для створення більш загальних методів та класів, які можуть приймати об'єкти різних типів.

1. Застосування підстановочних типів.

Застосування підстановочних типів включає використання ? для прийняття об'єктів різних типів, таких як <?> для довільного типу або <? extends T> для обмеження типу до підкласу T або <? super T> для обмеження типу до суперкласу T.

**Висновок**

Ознайомився з використанням параметризованого програмування. Створив клас який реалізує предметну область «множина». Та розробив клас драйвер який показує роботу параметризованого класу контейнера.