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*

*Яцків Адріян*

**Прийняв:**

доцент кафедри ЕОМ

Іванов Ю. С.

**Львів – 2023**

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

**Завдання: Варіант 1 «Масив»:**

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**

**M\_Array**

package KI306.Yatskiv.Lab6;  
  
  
import java.util.Arrays;  
import java.util.HashMap;  
import java.util.Map;  
*/\*\*  
 \* Реалізація класу M\_Array  
 \** ***@author*** *Adriyan Yatskiv KI-306  
 \** ***@version*** *1.0  
 \** ***@since*** *version 1.0  
 \*/*public class M\_Array<T extends Comparable<Object>> {  
 private T[] array;  
 private Class<?>[] types;  
 private int[] typeCounts;  
 public M\_Array() {  
 this.array = (T[]) new Comparable[0];  
 this.types = new Class<?>[0];  
 this.typeCounts = new int[0];  
 }  
  
  
 // Пошук найбільшого значення у масиві  
 public void findMaxForEachType() {  
 for (int i = 0; i < types.length; i++) {  
 T max = null;  
  
 for (int j = 0; j < array.length; j++) {  
 if (types[i].isInstance(array[j])) {  
 T element = (T) array[j];  
 if (max == null || element.compareTo(max) > 0) {  
 max = element;  
 }  
 }  
 }  
 System.out.println("Максимальне значення для типу " + types[i].getName() + ": " + max);  
 }  
 }  
  
 public void insert(T element) {  
 array = Arrays.copyOf(array, array.length + 1);  
 array[array.length - 1] = element;  
  
 Class<?> elementType = element.getClass();  
 int index = getTypeIndex(elementType);  
  
 if (index == -1) {  
 types = Arrays.copyOf(types, types.length + 1);  
 types[types.length - 1] = elementType;  
  
 typeCounts = Arrays.copyOf(typeCounts, typeCounts.length + 1);  
 typeCounts[typeCounts.length - 1] = 1;  
 } else {  
 typeCounts[index]++;  
 }  
 }  
 // Видалення елементу з масиву за значенням  
 public T remove(T element) {  
 for (int i = 0; i < array.length; i++) {  
 if (array[i].equals(element)) {  
 T removedElement = array[i];  
 T[] newArray = Arrays.copyOf(array, array.length - 1);  
 System.arraycopy(array, 0, newArray, 0, i);  
 System.arraycopy(array, i + 1, newArray, i, array.length - i - 1);  
 array = newArray;  
 return removedElement;  
 }  
 }  
 return null;  
 }  
 private int getTypeIndex(Class<?> elementType) {  
 for (int i = 0; i < types.length; i++) {  
 if (types[i] == elementType) {  
 return i;  
 }  
 }  
 return -1;  
 }  
  
 public int countStringsT() {  
 int count = 0;  
  
 for (T element : array) {  
 String str = element.toString();  
 if (str.startsWith("T")) {  
 count++;  
 }  
 }  
  
 System.out.println("Number of strings starting with 'T': " + count);  
 return count;  
 }  
  
  
 public void display() {  
 System.out.println(Arrays.toString(array));  
 }  
  
 public void printTypeCounts() {  
 for (int i = 0; i < types.length; i++) {  
 System.out.println(types[i].getName() + ": " + typeCounts[i]);  
 }  
 }  
}

**Main**

package KI306.Yatskiv.Lab6;  
  
import java.util.Map;  
  
*/\*\*  
 \* Дана програма є класом драйвером який тестує роботу класу M\_Array  
 \** ***@author*** *Adriyan Yatskiv KI-306  
 \** ***@version*** *1.0  
 \** ***@since*** *version 1.0  
 \*/*public class Main {  
 public static void main(String[] args) {  
 M\_Array container = new M\_Array<>();  
 // Додавання елементів  
 container.insert(42);  
 container.insert(8);  
 container.insert(3.94);  
 container.insert(9);  
 container.insert("Sample");  
 container.insert(-2);  
 container.insert(5.5);  
 container.insert(2);  
 container.insert("Text");  
 container.insert("-19.5");  
 container.insert("Something");  
 container.insert("Tnfo");  
 container.insert("Trembita");  
 container.insert("Trembita");  
  
 container.remove(5);  
  
 System.*out*.println("Масив: ");  
 container.display();  
  
 container.findMaxForEachType();  
  
 System.*out*.println("Кількість елементів кожного типу:");  
 container.printTypeCounts();  
  
 container.countStringsT();  
  
 }  
  
 }

**Результат виконання програми**

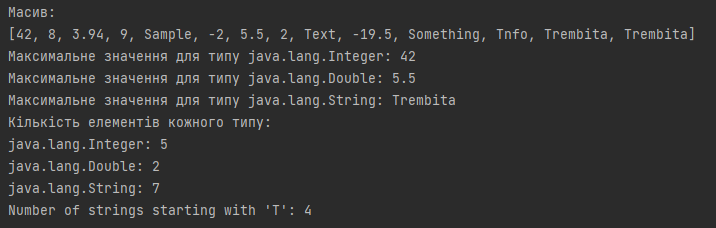
****

Рис.1 – Скріншот виконання програми

**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 «масив». Та розробив клас драйвер який показує роботу параметризованого класу контейнера.