Міністерство освіти і науки України

Національний університет „Львівська політехніка”

Кафедра ЕОМ



**Звіт**

з лабораторної роботи №6

з дисципліни: “Кросплатформні засоби програмування”

на тему: “ ПАРАМЕТРИЗОВАНЕ ПРОГРАМУВАННЯ ”

Виконав:

ст. гр. КІ-307

Магора О.Т.

Прийняв:

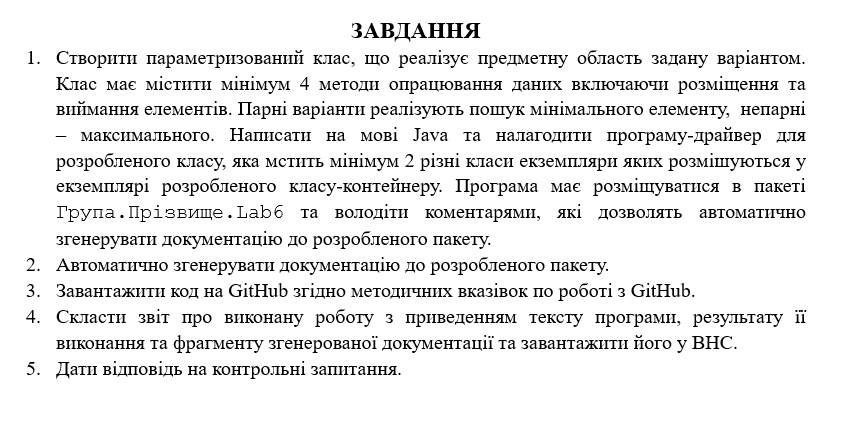
Іванов Ю.С.

Львів – 2023

**Мета:** оволодіти навичками параметризованого програмування мовою Java.

**Теоретичні відомості:**

Параметризоване програмування є аналогом шаблонів у С++. Воно полягає у написанні коду, що можна багаторазово застосовувати з об’єктами різних класів. Користувачів параметризованого програмування можна поділити на 3 рівні кваліфікації: 1. ті, що користуються готовими класами; 2. ті, що користуються готовими класами і вміють виправляти помилки, що виникають при їх використанні; 3. ті, що пишуть власні параметризовані класи. Для успішного застосування параметризованого програмування слід навчитися розуміти помилки, що генерує середовище при компіляції програми, що можуть стосуватися, наприклад, неоднозначності визначення спільного суперкласу для всіх переданих об’єктів. З іншої сторони необхідно передбачити захист від присвоєння об’єктів параметризованого класу, що містять об’єкти підкласу об’єктам параметризованого класу, що містять об’єкти суперкласу і дозволити зворотні присвоєння. Для вирішення цієї проблеми у мові Java введено так звані підстановочні типи. Це далеко не всі «підводні камені», що виникають при застосуванні параметризованого програмування.



**Варіант:**



**Код програми:**

Item1.java

// Файл Item1.java

package KI\_307.Magora.Lab6;

/\*\*

\* Клас представлення елемента для розміщення на поличці, реалізує Comparable для порівняння.

\*/

public class Item1 implements Comparable<Item1> {

private int value;

static int *count*;

/\*\*

\* Конструктор для ініціалізації об'єкта зі значенням.

\*

\* **@param** value Значення елемента.

\*/

public Item1(int value) {

this.value = value;

this.*count* += 1;

}

*@Override*

public int compareTo(Item1 other) {

return Integer.*compare*(this.value, other.value);

}

*@Override*

public String toString() {

return "Item1{" +

"value=" + value +

'}';

}

}

Файл Item2.java

// Файл Item2.java

package KI\_307.Magora.Lab6;

/\*\*

\* Клас представлення елемента для розміщення на поличці, реалізує Comparable для порівняння за ім'ям.

\*/

public class Item2 implements Comparable<Item2> {

private String name;

/\*\*

\* Конструктор для ініціалізації об'єкта з ім'ям.

\*

\* **@param** name Ім'я елемента.

\*/

public Item2(String name) {

this.name = name;

}

*@Override*

public int compareTo(Item2 other) {

return this.name.compareTo(other.name);

}

*@Override*

public String toString() {

return "Item2{" +

"name='" + name + '\'' +

'}';

}

}

Файл Main.java

// Файл Main.java (програма-драйвер)

package KI\_307.Magora.Lab6;

/\*\*

\* Головний клас для використання та демонстрації класів Shelf, Item1 та Item2.

\*/

public class Main {

public static void main(String[] args) {

// Створення полички та розміщення елементів

Shelf<Item1> shelf1 = new Shelf<>();

shelf1.placeItem(new Item1(10));

shelf1.placeItem(new Item1(5));

shelf1.placeItem(new Item1(8));

// Вивід мінімального елемента на поличці

System.***out***.println("Minimum item on Shelf1: " + shelf1.findMinItem());

// Створення іншої полички та розміщення елементів

Shelf<Item2> shelf2 = new Shelf<>();

shelf2.placeItem(new Item2("Book"));

shelf2.placeItem(new Item2("Pen"));

shelf2.placeItem(new Item2("Notebook"));

// Вивід мінімального елемента на іншій поличці

System.***out***.println("Minimum item on Shelf2: " + shelf2.findMinItem());

*main1*();

System.***out***.println("Кількість предметів "+Item1.*count*);

}

public static void main1() {

Shelf<Item1> shelf1 = new Shelf<>();

shelf1.placeItem(new Item1(10));

}

}

// Файл Shelf.java

package KI\_307.Magora.Lab6;

import java.util.ArrayList;

import java.util.Collections;

import java.util.List;

/\*\*

\* Клас, що представляє поличку з можливістю розміщення та опрацювання різних предметів.

\*

\* @param <T> Тип об'єктів, які можна розмістити на поличці та які реалізують Comparable для пошуку мінімального елементу.

\*/

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

private List<T> items;

/\*\*

\* Конструктор для ініціалізації порожньої полички.

\*/

public Shelf() {

this.items = new ArrayList<>();

}

public void placeItem(T item) {

items.add(item);

}

/\*\*

\* Метод для видалення останнього елемента з полички.

\*

\* @return Видалений елемент або null, якщо поличка порожня.

\*/

public T removeItem() {

if (items.isEmpty()) {

System.out.println("Shelf is empty. Cannot remove item.");

return null;

}

return items.remove(items.size() - 1);

}

/\*\*

\* Метод для пошуку мінімального елемента на поличці.

\*

\* @return Мінімальний елемент або null, якщо поличка порожня.

\*/

public T findMinItem() {

if (items.isEmpty()) {

System.out.println("Shelf is empty. No minimum item.");

return null;

}

return Collections.min(items);

}

/\*\*

\* Метод для виведення елементів на поличці.

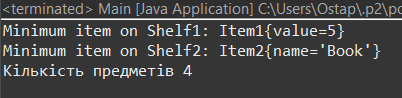
\*/

public void displayItems() {

System.out.println("Items on the shelf: " + items);

}

}



**Висновок:** на цій лабораторній роботі я ознайомився та оволодів навичками параметризованого програмування мовою програмування Java.