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

Глухенький Д. Ю.

**Прийняв:**

доцент кафедри ЕОМ

Іванов Ю. С.

**Львів – 2023**

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

**Завдання ( Варіант 5 «Двозвʼязний список»):**

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**

DoublyLinkedList.java

package KI306.Hlukhenkyi.lab6;  
  
*/\*\*  
 \* Node class represents a node in a doubly linked list.  
 \*  
 \* @param* <*T*> *The type of data stored in the node.  
 \*/*class Node<T> {  
 T data; // Data stored in the node  
 Node<T> next; // Reference to the next node  
 Node<T> prev; // Reference to the previous node  
  
 */\*\*  
 \* Constructor to create a new node with the given data.  
 \*  
 \* @param data The data to be stored in the node.  
 \*/* public Node(T data) {  
 this.data = data;  
 }  
}  
  
*/\*\*  
 \* DoublyLinkedList class represents a doubly linked list.  
 \*  
 \* @param* <*T*> *The type of data stored in the list (must be Comparable).  
 \*/*public class DoublyLinkedList<T extends Comparable<T>> {  
 private Node<T> head; // Reference to the first node in the list  
 private Node<T> tail; // Reference to the last node in the list  
  
 */\*\*  
 \* Adds a new node with the given data to the end of the list.  
 \*  
 \* @param data The data to be added to the list.  
 \*/* public void add(T data) {  
 Node<T> newNode = new Node<>(data);  
 if (head == null) {  
 head = newNode;  
 tail = newNode;  
 } else {  
 tail.next = newNode;  
 newNode.prev = tail;  
 tail = newNode;  
 }  
 }  
  
 */\*\*  
 \* Finds and returns the minimum element in the list.  
 \*  
 \* @return The minimum element in the list.  
 \*/* public T findMin() {  
 if (head == null) {  
 return null;  
 }  
 T min = head.data;  
 Node<T> current = head;  
 while (current != null) {  
 if (current.data.compareTo(min) < 0) {  
 min = current.data;  
 }  
 current = current.next;  
 }  
 return min;  
 }  
  
 */\*\*  
 \* Finds and returns the maximum element in the list.  
 \*  
 \* @return The maximum element in the list.  
 \*/* public T findMax() {  
 if (head == null) {  
 return null;  
 }  
 T max = head.data;  
 Node<T> current = head;  
 while (current != null) {  
 if (current.data.compareTo(max) > 0) {  
 max = current.data;  
 }  
 current = current.next;  
 }  
 return max;  
 }  
  
 */\*\*  
 \* Displays the elements of the list in order.  
 \*/* public void display() {  
 Node<T> current = head;  
 while (current != null) {  
 System.*out*.print(current.data + " ");  
 current = current.next;  
 }  
 System.*out*.println();  
 }  
}

Main.java

import KI306.Hlukhenkyi.lab6.DoublyLinkedList;  
  
public class Main {  
 public static void main(String[] args) {  
 DoublyLinkedList<Integer> list = new DoublyLinkedList<>();  
  
 DoublyLinkedList<String> names = new DoublyLinkedList<>();  
 // Додати деякі елементи до списку  
 list.add(5);  
 list.add(3);  
 list.add(8);  
 list.add(2);  
  
 names.add("Hey");  
 names.add("Apple");  
 names.add("Graz");  
 names.add("Puppy");  
  
 names.display();  
 String min02 = names.findMax();  
 System.*out*.println("Мінімальний елемент: " + min02);  
  
  
 // Вивести елементи списку  
 list.display();  
  
 // Знайти та вивести мінімальний елемент  
 Integer min = list.findMin();  
 System.*out*.println("Мінімальний елемент: " + min);  
  
 // Знайти та вивести максимальний елемент  
 Integer max = list.findMax();  
 System.*out*.println("Максимальний елемент: " + max);  
 }  
}

**Результат виконання програми**

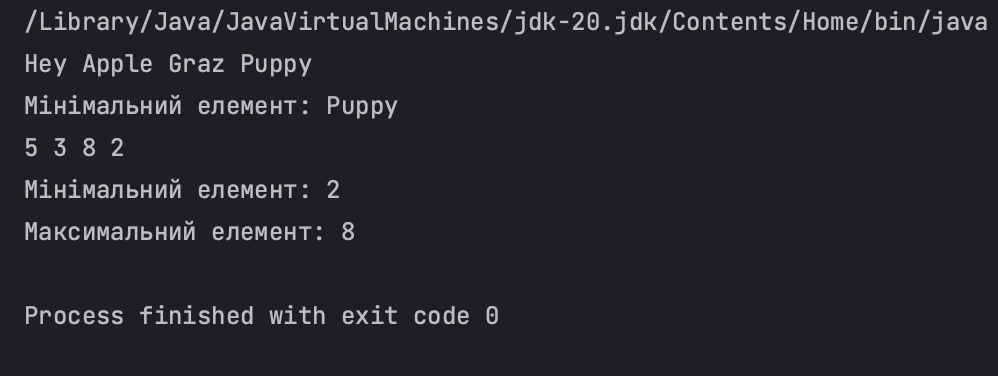
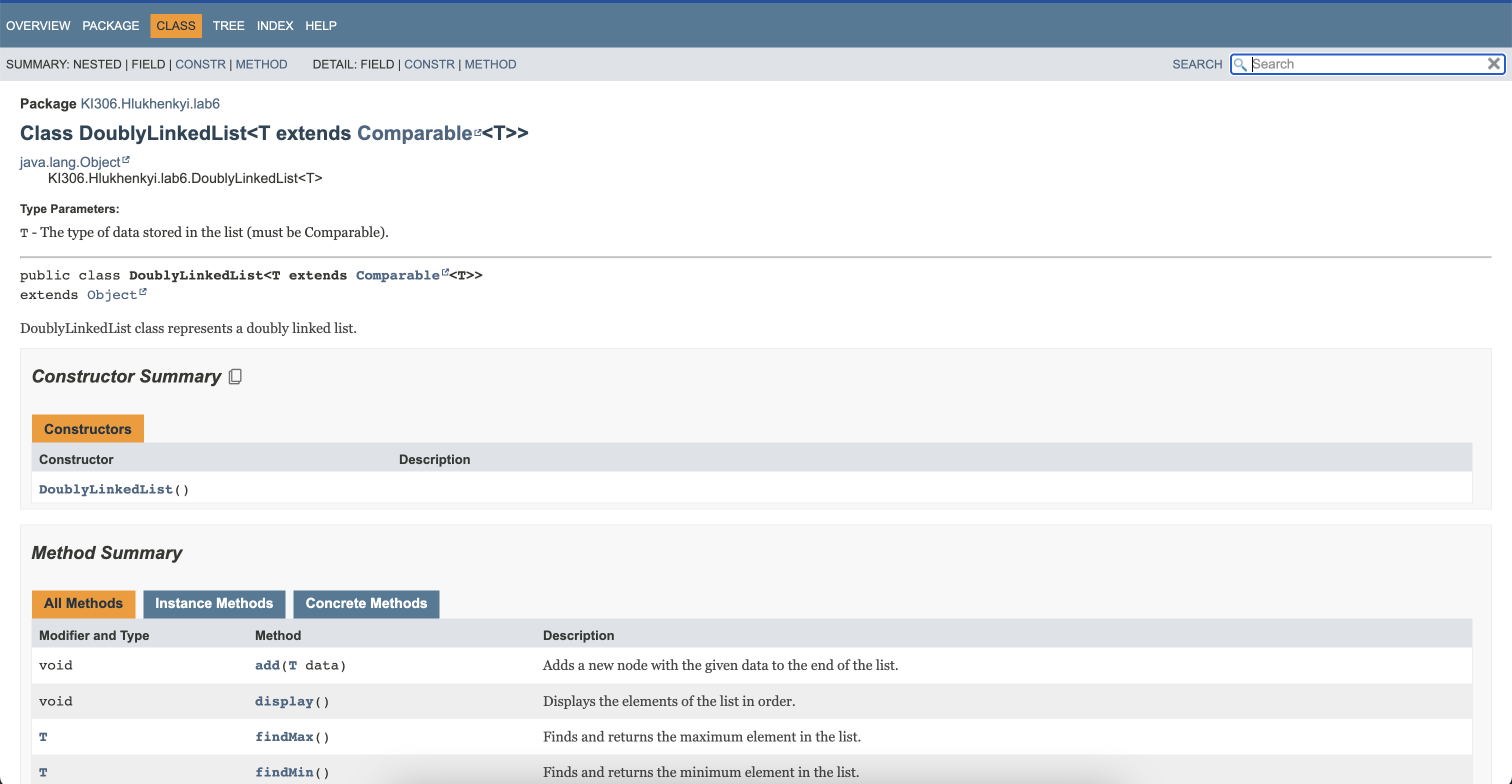
****

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.