

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
Бокало П.М.

Прийняв:
доцент кафедри ЕОМ
Іванов Ю. С.

Мета роботи: оволодіти навиками параметризованого програмування мовою Java.

Завдання(варіант №2):

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 розміщуються 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. Завантажити код на GitHub згідно методичних вказівок по роботі з GitHub.
4. Скласти звіт про виконану роботу з приведенням тексту програми, результату її виконання та фрагменту згенерованої документації та завантажити його у ВНС.
5. Дати відповідь на контрольні запитання.

Індивідуальне завдання:

2. Однозв'язний список

Вихідний код програми:

Файл KI306.Bokalo.Lab6.java:

```
package KI306.Bokalo.Lab6;

import java.io.IOException;

/**
 * Дана програма є класом драйвером який тестує роботу класу My_list
 * @author Petro Bokalo KI-306
 * @version 1.0
 * @since version 1.0
 */

public class Lab6BokaloKI306
{
    public static void main(String[] args)throws IOException
    {
        My_list<String> lst = new My_list();
        lst.add("zppleeeeeeeee");
        lst.add("orange");
        System.out.print(lst.findMin() + "\n");
        lst.display();

        My_list<Integer> lst1 = new My_list();
```

```

        lst1.add(1);
        lst1.add(34);
        lst1.add(-100);
        System.out.print("\n" + lst1.findMin() + "\n");
        lst1.display();
        lst1.remove(34);
        lst1.display();
    }
}

```

Файл My_list.java

```

package KI306.Bokalo.Lab6;

/**
 * Тут реалізовано клас My_list
 * @author Petro Bokalo KI-306
 * @version 1.0
 * @since version 1.0
 */

public class My_list<T extends Comparable<T>> {
    private Node<T> head;

    public My_list() {
        this.head = null;
    }

    class Node<I> {
        T data;
        Node<T> next;

        public Node(T data) {
            this.data = data;
            this.next = null;
        }
    }

    // Додавання елементу в кінець списку
    public void add(T data) {
        Node<T> newNode = new Node<>(data);
        if (head == null) {
            head = newNode;
        } else {
            Node<T> current = head;
            while (current.next != null) {
                current = current.next;
            }
            current.next = newNode;
        }
    }

    // Видалення елементу за значенням
    public void remove(T data) {
        if (head == null) {
            return;
        }
        if (head.data.equals(data)) {
            head = head.next;
            return;
        }
        Node<T> current = head;
    }
}

```

```

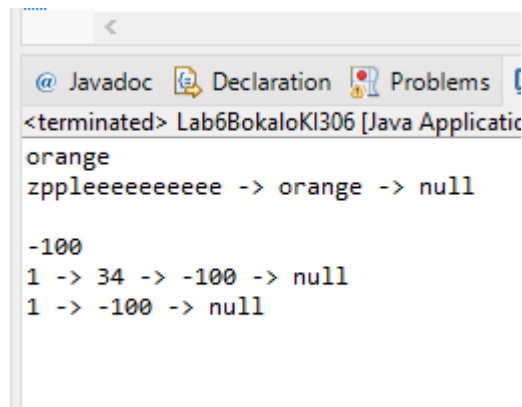
        while (current.next != null) {
            if (current.next.data.equals(data)) {
                current.next = current.next.next;
                return;
            }
            current = current.next;
        }
    }

    // Пошук мінімального елемента
    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;
    }

    // Виведення всього списку
    public void display() {
        Node<T> current = head;
        while (current != null) {
            System.out.print(current.data + " -> ");
            current = current.next;
        }
        System.out.println("null");
    }
}

```

Результат роботи програми:



```

<terminated> Lab6BokaloKI306 [Java Applicati
orange
zppleeeeeeeeeee -> orange -> null

-100
1 -> 34 -> -100 -> null
1 -> -100 -> null

```

Фрагмент згенерованої документації:

Package KI306.Bokalo.Lab6

Class Lab6BokaloKI306

[java.lang.Object](#)

KI306.Bokalo.Lab6.Lab6BokaloKI306

```
public class Lab6BokaloKI306
extends Object
```

Дана програма є класом драйвером який тестує роботу класу My_list

Since:

version 1.0

Version:

1.0

Author:

Petro Bokalo KI-306

Відповідь на контрольні питання

1) Дайте визначення терміну «параметризоване програмування».

Параметризоване програмування - це підхід до програмування, коли код можна написати один раз для різних типів даних або об'єктів, використовуючи параметризовані (загальні) типи або методи.

2) Розкрийте синтаксис визначення простого параметризованого класу.

```
class MyClass<T> {
}
```

3) Розкрийте синтаксис створення об'єкту параметризованого класу.

```
MyClass<int> obj = new MyClass<int>();
```

4) Розкрийте синтаксис визначення параметризованого методу.

```
public void MyMethod<T>(T parameter) {  
}
```

5) Розкрийте синтаксис виклику параметризованого методу.

```
obj.MyMethod(5);
```

6) Яку роль відіграє встановлення обмежень для змінних типів?

Встановлення обмежень для змінних типів дозволяє задати певні умови або вимоги для типів даних, які можуть бути використані в параметризованому коді.

7) Як встановити обмеження для змінних типів?

Відповідь: Обмеження для змінних типів встановлюються за допомогою ключового слова `where`. Приклад:

```
public class MyClass<T> where T : SomeBaseClass {  
}
```

8) Розкрийте правила спадкування параметризованих типів.

Правила спадкування параметризованих типів спираються на ієрархію класів і обмеження, які визначені для типів.

9) Яке призначення підстановочних типів?

Підстановочні типи дозволяють створювати загальні типи, які можуть працювати з різними типами даних, або вказувати обмеження для параметризованих типів.

10) Застосування підстановочних типів.

Застосування підстановочних типів включає створення загальних колекцій, класів, методів і інших структур, які можуть працювати з різними типами даних без необхідності дублювати код.

Висновок: на цій лабораторній роботі, я оволодів навиками параметризованого програмування мовою Java.