Міністерство освіти і науки України

Національний університет «Львівська політехніка»

Кафедра ЕОМ

Звіт

до лабораторної роботи № 6

з дисципліни: «Кросплатформні засоби програмування»

на тему: «Параметризоване програмування»

Варіант №13

Виконав:
ст. гр. КІ-203
Панченко Д. В.

Прийняв:
доцент кафедри ЕОМ
Іванов Ю. С.

Львів 2023

Мета: оволодіти навиками параметризованого програмування мовою Java.

Завдання:

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. 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.

Варіант завдання: 13 – Словник (тип даних)

Вихідний код програми:

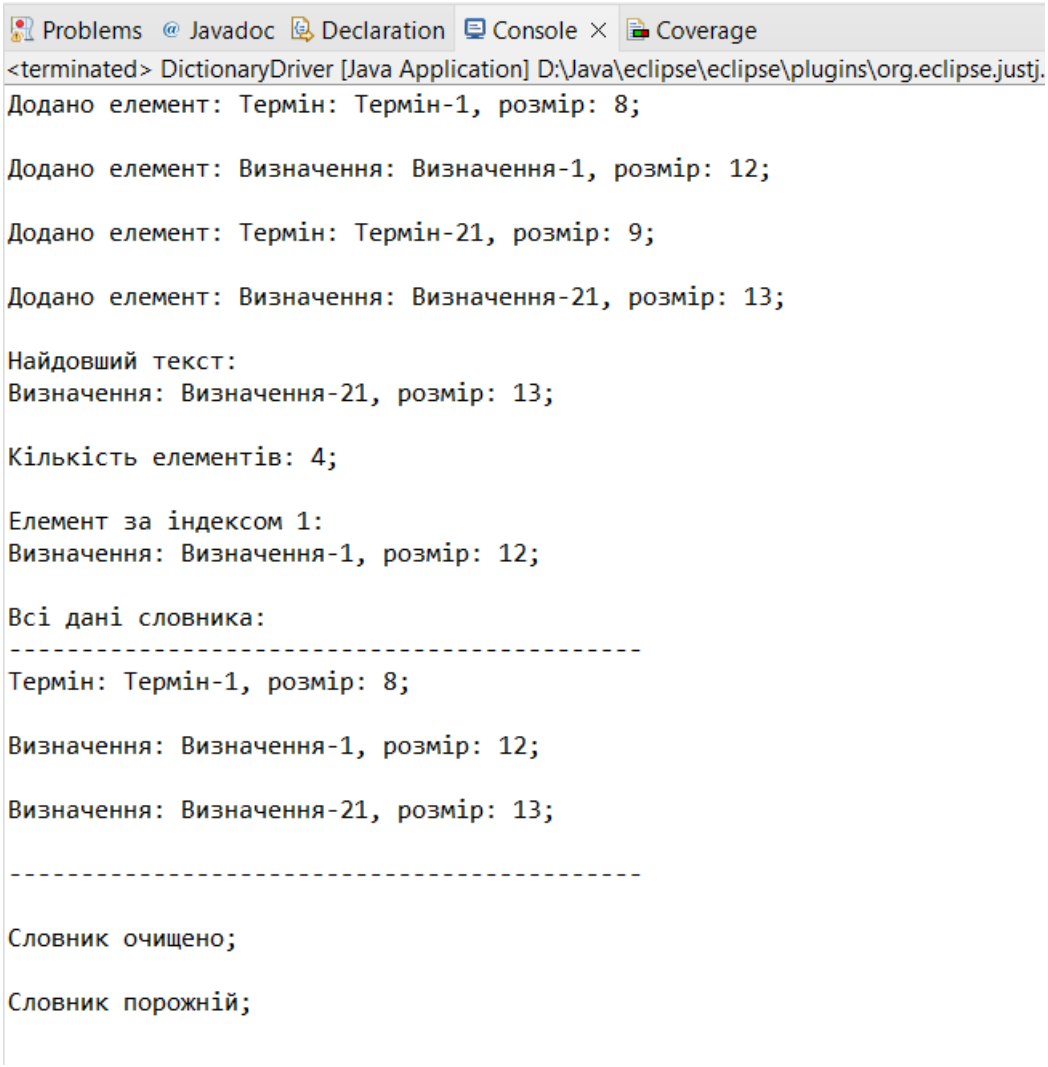
```
package KI_306.Panchenko.Lab6;

import java.util.*;
import java.io.*;


/**
* Class <code>DictionaryDriver</code> Driver class for class Dictionary
* @author Panchenko
*/

public class DictionaryDriver {

	/**
	 * @param args is arguments
	 */

	public static void main(String[] args)
	{
	Dictionary <? super Data> dictionary = new Dictionary <Data>();

	dictionary.AddData(new Term("Термін-1"));
	dictionary.AddData(new Definition("Визначення-1" ));
	dictionary.AddData(new Term("Термін-21" ));
	dictionary.AddData(new Definition("Визначення-21" ));

	Data res = dictionary.findMax();
	System.out.print("Найдовший текст: \n");
	res.print();
```

```
        dictionary.getArrSize();

        dictionary.DeleteData(2);

        dictionary.printDataAtIndex(1);

        dictionary.PrintAllData();

        dictionary.clearDictionary();

        dictionary.PrintAllData();

        }


}

/**
* Class <code>Dictionary</code> Parameterized class
* @author Panchenko
*/

class Dictionary <T extends Data>
{
        private ArrayList<T> arr;
        public Dictionary()
        {
                arr = new ArrayList<T>();
        }

        public T findMax()
        {
                if (!arr.isEmpty())
                {
                        T max = arr.get(0);
                        for (int i=1; i< arr.size(); i++)
                        {
                                if ( arr.get(i).compareTo(max) > 0 )
                                        max = arr.get(i);
                        }
                        return max;
                }
                return null;
        }

        public void AddData(T data)
        {
                arr.add(data);
                System.out.print("Додано елемент: ");
                data.print();
        }

        public void DeleteData(int i)
        {
                arr.remove(i);
        }


        public void getArrSize()
        {
                System.out.print("Кількість елементів: " + arr.size() + ";\n\n");
        }
```

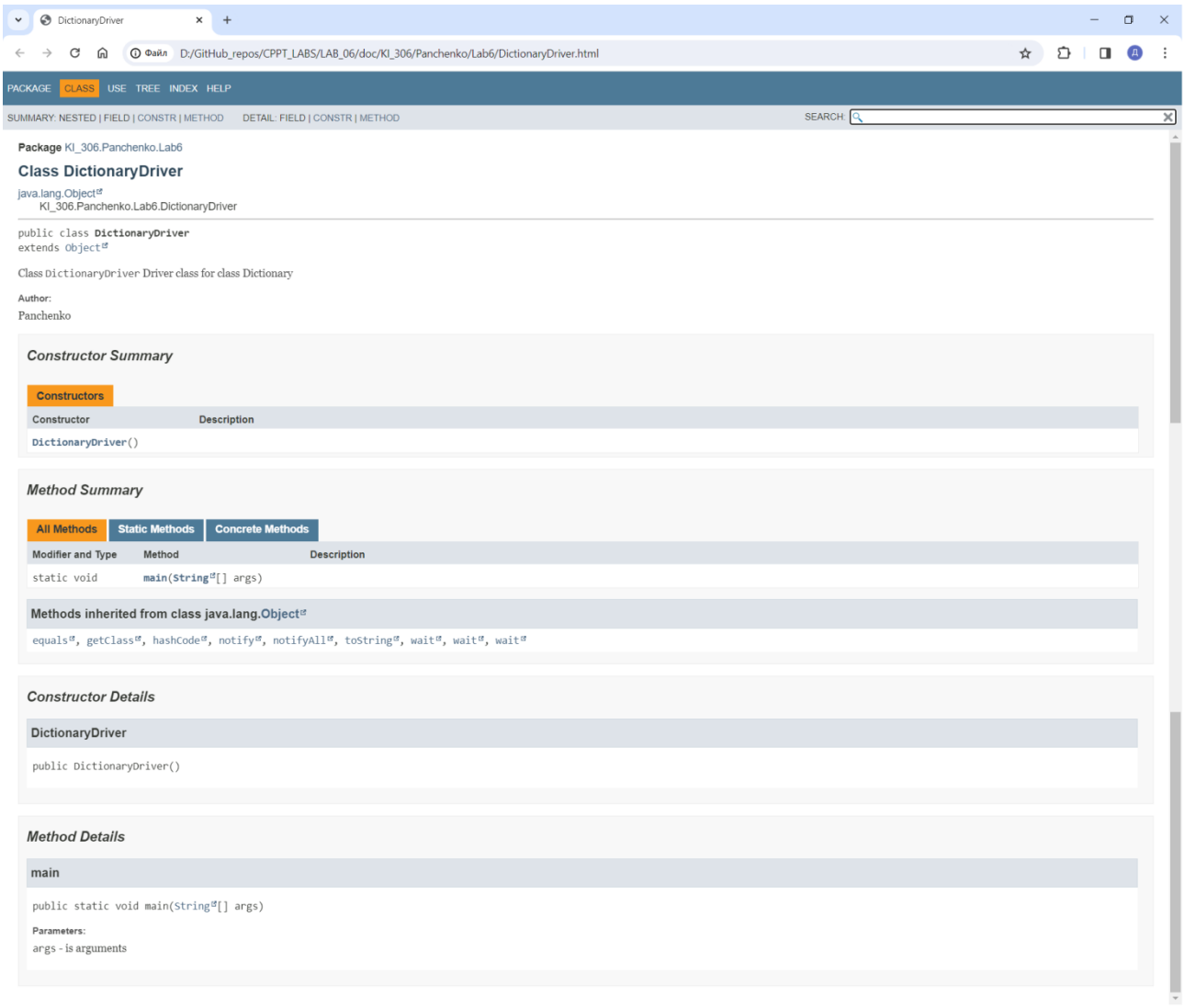
```
        public void printDataAtIndex(int i) {
            if (i >= 0 && i < arr.size()) {
                T data = arr.get(i);
                System.out.println("Елемент за індексом " + i + ": ");
                data.print();
            } else {
                System.out.println("Елементу за цим індексом не знайдено;\n\n");
            }
        }

        public void clearDictionary() {
            arr.clear();
            System.out.println("Словник очищено;\n");
        }

        public void PrintAllData() {
              if (arr.isEmpty()) {
                System.out.println("Словник порожній;\n\n");
            }
              else {
                    System.out.println("Всі дані словника:\n----------------------------
----------------");
                    for (T item : arr) {
                          item.print();
                    }
                    System.out.println("-----------------------------------------
\n");
            }
        }


}

interface Data extends Comparable<Data>
{
        public int getSize();
        public void print();
}

class Term  implements Data
{
        private String word;
        private int size;

        public Term (String w)
        {
              word = w;
              size = w.length();
        }

        public String getTerm()
        {
              return word;
        }
        public void setTerm(String w)
        {
              word = w;
        }

        public int getSize()
        {
        return size;
        }
```

```
	public int compareTo(Data w)
	{
	Integer s = size;
	return s.compareTo(w.getSize());
	}


	public void print()
	{
		System.out.print("Термін: " + word + ", розмір: " + size + ";\n\n");
	}


}

class Definition implements Data
{
	private String description;
	private int size;

	public Definition(String d)
	{
		description = d;
		size = d.length();;
	}

	public String getDefinition()
	{
		return description;
	}
	public void setDefinition(String d)
	{
		description = d;
	}

	public int getSize()
	{
	return size;
	}

	public int compareTo(Data d)
	{
	Integer s = size;
	return s.compareTo(d.getSize());
	}

	public void print()
	{
		System.out.print("Визначення: " + description + ", розмір: " + size  +
";\n\n");
	}

}
```

Результат виконання програми:



Фрагмент згенерованої документації:



Відповіді на контрольні запитання:

1. Дайте визначення терміну «параметризоване програмування».

Параметризоване програмування є аналогом шаблонів у C++. Воно полягає у написанні коду, що можна багаторазово застосовувати з об’єктами різних класів.

2. Розкрийте синтаксис визначення простого параметризованого класу.

Параметризований клас – це клас з однією або більше змінними типу. Синтаксис оголошення параметризованого класу:

[public] class НазваКласу <параметризованийТип{,параметризованийТип}>
{…}

3. Розкрийте синтаксис створення об’єкту параметризованого класу.

Синтаксис створення об’єкту параметризованого класу:

НазваКласу < перелікТипів > = new НазваКласу < перелікТипів > (параметри);

4. Розкрийте синтаксис визначення параметризованого методу.

Синтаксис оголошення параметризованого методу:

Модифікатори <параметризованийТип{,параметризованийТип}> типПовернення назваМетоду(параметри);

5. Розкрийте синтаксис виклику параметризованого методу.

Синтаксис виклику параметризованого методу:

(НазваКласу|НазваОб'єкту).[<перелікТипів>] НазваМетоду(параметри);

6. Яку роль відіграє встановлення обмежень для змінних типів?

Може бути ситуація, коли метод у процесі роботи викликає з-під об'єкта параметризованого типу метод, що визначається у деякому інтерфейсі. У такому випадку немає ніякої гарантії, що цей метод буде реалізований у кожному класі, що передається через змінну типу. Щоб вирішити цю проблему у мові Java можна задати обмеження на множину можливих типів, що можуть бути підставлені замість параметризованого типу.

7. Як встановити обмеження для змінних типів?

Після змінної типу слід використати ключове слово extends і вказати один суперклас, або довільну кількість інтерфейсів (через знак &), від яких має походити реальний тип, що підставляється замість параметризованого типу. Якщо одночасно вказуються інтерфейси і суперклас, то суперклас має стояти першим у списку типів після ключового слова extends.

8. Розкрийте правила спадкування параметризованих типів.

1) Всі класи, що утворені з одного і того ж параметризованого класу з використанням різних значень змінних типів є незалежними.
2) Завжди можна перетворити параметризований клас у «сирий» клас.
3) Параметризовані класи можуть розширювати або реалізовувати інші параметризовані класи.

9. Яке призначення підстановочних типів?

Підстановочні типи були введені у мову Java для збільшення гнучкості жорсткої снуючої системи параметризованих типів. На відміну від неї підстановочні типи дозволяють враховувати залежності між типами, що виступають параметрами для параметризованих типів.

10. Застосування підстановочних типів.

Підстановочні типи застосовуються у вигляді параметру типу, що передається у трикутних дужках при утворені реального типу з параметризованого типу, наприклад, у методі main. Підстановочні типи дозволяють реалізувати:

1) обмеження підтипу;
2) обмеження супертипу;
3) необмежені підстановки.

Висновок: Я оволодів навиками параметризованого програмування мовою Java.