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
Чаус Б.В.
Прийняв:
доцент кафедри ЕОМ
Іванов Ю. С.

Львів – 2023

Мета роботи: оволодіти навиками параметризованого програмування мовою Java.

Завдання (варіант № 24)

24. Коробка для інструментів

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.

Вихідний код програми

Файл MyTool.java

```java
package KI306.Chaus.Lab6; // Specifies the package name

import java.util.ArrayList; // Imports the ArrayList class from the java.util package

public class MyTool<T> implements Tool1<T> {
    // Defines a generic class MyTool that implements the Tool1 interface

    @Override
    public void print_data(T data) {
        // Overrides the method defined in the Tool1 interface to print generic data
        System.out.println(data); // Prints the value of 'data'
    }

    public void print_data(int prefix) {
        // Defines a method to print an integer value (prefix)
        System.out.println(prefix); // Prints the value of 'prefix'
    }

    public void print_data(char suffix) {
        // Defines a method to print a character value (suffix)
        System.out.println(suffix); // Prints the value of 'suffix'
    }
}
```

Файл Tool.java

```java
package KI306.Chaus.Lab6;
```

```java
// Specifies the package name

interface Tool extends Comparable<Tool> {
    // Defines an interface named Tool that extends Comparable for comparison

    int getSize();
    // Declares a method 'getSize' that returns an integer

    void print();
    // Declares a method 'print' with no return value
}

interface Tool1<T> {
    // Defines a generic interface Tool1

    public void print_data(T data);
    // Declares a method 'print_data' that takes a generic parameter 'data'
}
```

Файл ToolBox.java

```java
package KI306.Chaus.Lab6;
import java.util.ArrayList;

// Define a class named ToolBox that can hold objects of type Tool or its subtypes
class ToolBox<T extends Tool> {
    private ArrayList<T> tools; // ArrayList to store tools

    // Constructor for ToolBox
    public ToolBox() {
        tools = new ArrayList<>(); // Initialize the ArrayList of tools
    }

    // Find and return the smallest tool based on comparison
    public T findMin() {
        if (!tools.isEmpty()) {
            T min = tools.get(0);
            for (int i = 1; i < tools.size(); i++) {
                if (tools.get(i).compareTo(min) < 0)
                    min = tools.get(i);
            }
            return min;
        }
        return null;
    }

    // Add a tool to the toolbox
    public void addTool(T tool) {
        tools.add(tool);
        System.out.print("tool added: ");
        tool.print(); // Print information about the added tool
    }

    // Remove a tool at a specified index
    public void removeTool(int i) {
        if (i >= 0 && i < tools.size()) {
            tools.remove(i);
            System.out.println("Tool removed at index " + i);
        } else {
            System.out.println("Invalid index. Cannot remove tool.");
        }
    }

    // Print information about the tools in the toolbox
    public void printContents() {
        if (!tools.isEmpty()) {
```

```
            for (T tool : tools) {
                tool.print(); // Print information about each tool
            }
        } else {
            System.out.println("Tool box is empty. No tools available.");
        }
    }
}

// Define a class named Screwdriver that implements the Tool interface
class Screwdriver implements Tool {
    private String screwdriverType;
    private String screwdriverBrand;
    private double screwdriverCost;
    private int screwdriverSize;

    // Constructor for Screwdriver
    public Screwdriver(String sType, String sBrand, double sCost, int sSize) {
        screwdriverType = sType;
        screwdriverBrand = sBrand;
        screwdriverCost = sCost;
        screwdriverSize = sSize;
    }

    // Getter and setter methods for screwdriver properties

    public String getScrewdriverType() {
        return screwdriverType;
    }

    public void setScrewdriverType(String type) {
        screwdriverType = type;
    }

    public String getScrewdriverBrand() {
        return screwdriverBrand;
    }

    public void setScrewdriverBrand(String brand) {
        screwdriverBrand = brand;
    }

    public double getScrewdriverCost() {
        return screwdriverCost;
    }

    public void setScrewdriverCost(double cost) {
        screwdriverCost = cost;
    }

    public void setScrewdriverSize(int size) {
        screwdriverSize = size;
    }

    // Implement the getSize method from the Tool interface
    public int getSize() {
        return screwdriverSize;
    }

    // Implement the compareTo method from the Comparable interface
    public int compareTo(Tool tool) {
        Integer s = screwdriverSize;
        return s.compareTo(tool.getSize());
    }
```

```java
    // Print information about the screwdriver
    public void print() {
        System.out.println("[Screwdriver]");
        System.out.println("  Type: " + screwdriverType);
        System.out.println("  Brand: " + screwdriverBrand);
        System.out.println("  Cost: " + screwdriverCost + " $");
        System.out.println("  Size: " + screwdriverSize);
        System.out.println();
    }
}

// Define a class named Wrench that implements the Tool interface
class Wrench implements Tool {
    private String wrenchBrand;
    private int wrenchWeight;
    private int wrenchSize;

    // Constructor for Wrench
    public Wrench(String wBrand, int wWeight, int wSize) {
        wrenchBrand = wBrand;
        wrenchWeight = wWeight;
        wrenchSize = wSize;
    }

    // Getter and setter methods for wrench properties

    public String getWrenchBrand() {
        return wrenchBrand;
    }

    public void setWrenchBrand(String brand) {
        wrenchBrand = brand;
    }

    public int getWrenchWeight() {
        return wrenchWeight;
    }

    public void setWrenchWeight(int weight) {
        wrenchWeight = weight;
    }

    public void setWrenchSize(int size) {
        wrenchSize = size;
    }

    // Implement the getSize method from the Tool interface
    public int getSize() {
        return wrenchSize;
    }

    // Implement the compareTo method from the Comparable interface
    public int compareTo(Tool tool) {
        Integer s = wrenchSize;
        return s.compareTo(tool.getSize());
    }

    // Print information about the wrench
    public void print() {
        System.out.println("[Wrench]");
        System.out.println("  Brand: " + wrenchBrand);
        System.out.println("  Weight: " + wrenchWeight);
        System.out.println("  Size: " + wrenchSize);
```

```
        System.out.println();
    }
}
```

Файл ToolBoxApp.java

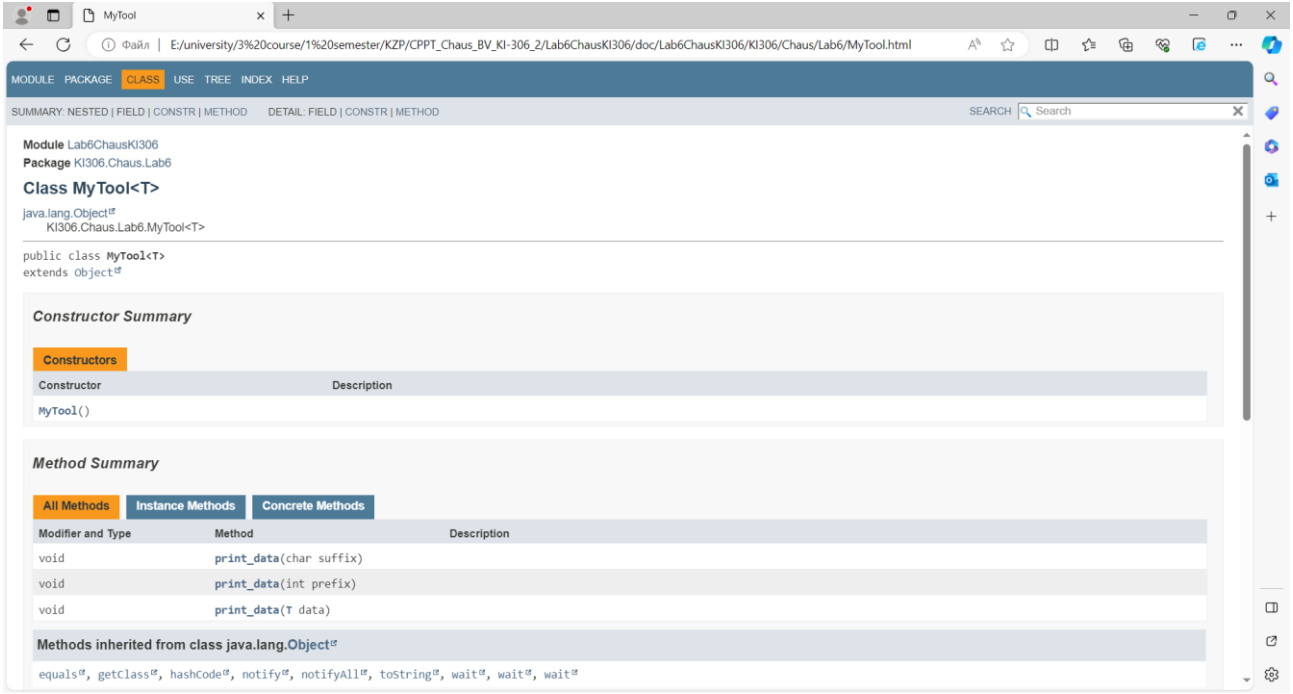
```
package KI306.Chaus.Lab6;

public class ToolBoxApp {
    public static void main(String[] args) {
        // Create a ToolBox with tools that are of type Tool or its super type
        ToolBox<? super Tool> suitcase = new ToolBox<>();

        // Add Screwdrivers and Wrenches to the suitcase ToolBox
        suitcase.addTool(new Screwdriver("Phillips", "ExampleBrand", 25.99, 6));
        suitcase.addTool(new Screwdriver("Philp", "randI", 1.19, 11));
        suitcase.addTool(new Wrench("ExampleBrand", 500, 12));
        suitcase.addTool(new Wrench("LightTools", 300, 8));
        suitcase.addTool(new Wrench("BigTools", 700, 24));

        // Remove a tool from the suitcase ToolBox
        suitcase.removeTool(3);

        // Print the contents of the suitcase ToolBox
        System.out.print("\nContents of Tool Box: \n");
        suitcase.printContents();

        // Find and print the smallest tool in the suitcase ToolBox
        Tool minTool = suitcase.findMin();
        System.out.print("\nThe smallest tool in the Tool box is: ");
        minTool.print();

        // Create an instance of MyTool with String type and print a message
        MyTool<String> stringTool = new MyTool<>();
        stringTool.print_data("Hello, World!");

        // Create an instance of MyTool with Integer type and print a number
        MyTool<Integer> integerTool = new MyTool<>();
        integerTool.print_data(42);
    }
}
```

Результат виконання програми

```
tool added: [Screwdriver]
  Type: Phillips
  Brand: ExampleBrand
  Cost: 25.99 $
  Size: 6

tool added: [Screwdriver]
  Type: Philp
  Brand: randI
  Cost: 1.19 $
  Size: 11

tool added: [Wrench]
  Brand: ExampleBrand
  Weight: 500
  Size: 12

tool added: [Wrench]
  Brand: LightTools
  Weight: 300
  Size: 8
```

```
tool added: [Wrench]
  Brand: BigTools
  Weight: 700
  Size: 24

Tool removed at index 3

Contents of Tool Box:
[Screwdriver]
  Type: Phillips
  Brand: ExampleBrand
  Cost: 25.99 $
  Size: 6

[Screwdriver]
  Type: Philp
  Brand: randI
  Cost: 1.19 $
  Size: 11

[Wrench]
  Brand: ExampleBrand
  Weight: 500
  Size: 12

[Wrench]
  Brand: BigTools
  Weight: 700
  Size: 24


The smallest tool in the Tool box is: [Screwdriver]
  Type: Phillips
  Brand: ExampleBrand
  Cost: 25.99 $
  Size: 6

Hello, World!
42
```

Фрагмент згенерованої документації



Відповіді на контрольні запитання

1. Параметризоване програмування - це підхід до написання програмного коду, де типи даних можуть бути параметрами, що визначаються користувачем.

2. Синтаксис визначення простого параметризованого класу:

csharp

```
class НазваКласу<ТипПараметра> {
   // Вміст класу
}
```

3. Синтаксис створення об’єкту параметризованого класу:

```
НазваКласу<ТипАргументу> об’єкт = new НазваКласу<ТипАргументу>();
```

4. Синтаксис визначення параметризованого методу:

```
ТипРезультату НазваМетоду<ТипПараметра>(Параметри) {
   // Вміст методу
}
```

5. Синтаксис виклику параметризованого методу:

```
НазваМетоду<ТипАргументу>(Аргументи);
```

6. Встановлення обмежень для змінних типів дозволяє обмежити допустимий діапазон типів даних, які можна використовувати як параметри.

7. Обмеження для змінних типів можна встановити за допомогою ключового слова `where`:

```
class НазваКласу<ТипПараметра> where ТипПараметра : Обмеження {
   // Вміст класу
}
```

8. Параметризовані типи спадковуються так само, як і непараметризовані, але з можливістю використання параметрів типу в базовому класі.

9. Підстановочні типи використовуються для заміни конкретного типу абстрактним або інтерфейсним типом.

10. Застосування підстановочних типів дозволяє підключати різні типи, які відповідають певним критеріям, до параметризованого коду.

Висновок

Під час лабораторної роботи, я оволодів навиками параметризованого програмування мовою Java.