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 20

**Виконав:**

ст. групи КІ-306

Петрук О.С.

**Прийняв:**

доцент кафедри ЕОМ

Олексів М.В.

Львів – 2024

**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.

**Завдання згідно варіанту №20 – «Стек»**

**GitHub Repository:**

**Хід роботи**

Код програми:

***Stack.java***

package KI\_306.Petruk.Lab6;

import java.util.ArrayList;

import java.util.List;

/\*\*

\* A parameterized class that implements a stack with basic operations.

\*

\* @param <T> The type of data stored in the stack. Must be comparable.

\*/

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

private List<T> elements;

/\*\*

\* Constructor to create an empty stack.

\*/

public Stack() {

this.elements = new ArrayList<>();

}

/\*\*

\* Adds an element to the stack.

\*

\* @param element The element to add.

\*/

public void push(T element) {

elements.add(element);

}

/\*\*

\* Removes the last element from the stack.

\*

\* @return The removed element.

\* @throws IllegalStateException If the stack is empty.

\*/

public T pop() {

if (isEmpty()) {

throw new IllegalStateException("The stack is empty");

}

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

}

/\*\*

\* Returns the minimum element in the stack.

\*

\* @return The minimum element.

\* @throws IllegalStateException If the stack is empty.

\*/

public T findMin() {

if (isEmpty()) {

throw new IllegalStateException("The stack is empty");

}

T min = elements.get(0);

for (T element : elements) {

if (element.compareTo(min) < 0) {

min = element;

}

}

return min;

}

/\*\*

\* Checks if the stack is empty.

\*

\* @return {@code true} if the stack is empty; {@code false} otherwise.

\*/

public boolean isEmpty() {

return elements.isEmpty();

}

}

***Person.java***

package KI\_306.Petruk.Lab6;

/\*\*

\* A class representing a person with a name and an age.

\*/

public class Person implements Comparable<Person> {

private String name;

private int age;

public Person(String name, int age) {

this.name = name;

this.age = age;

}

public String getName() {

return name;

}

public int getAge() {

return age;

}

@Override

public int compareTo(Person other) {

return Integer.compare(this.age, other.age);

}

@Override

public String toString() {

return "Person{name='" + name + "', age=" + age + "}";

}

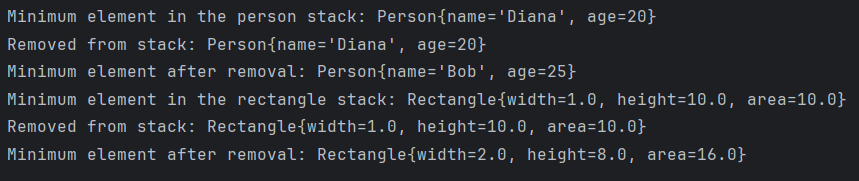
}

***Rectangle.java***

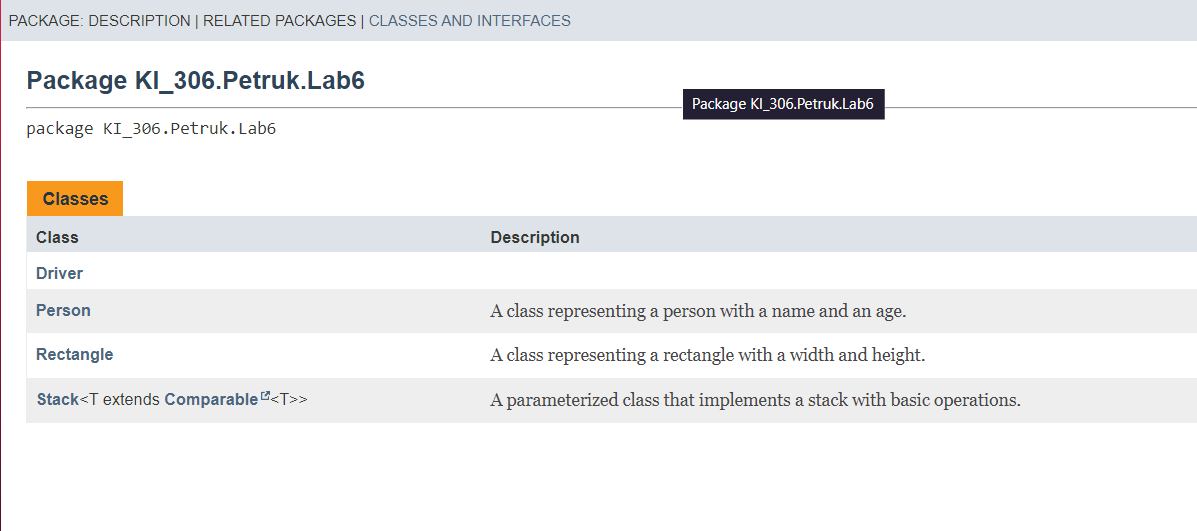
***package KI\_306.Petruk.Lab6;****/\*\*  
 \* A class representing a rectangle with a width and height.  
 \*/  
public class Rectangle implements Comparable<Rectangle> {  
 private double width;  
 private double height;  
  
 public Rectangle(double width, double height) {  
 this.width = width;  
 this.height = height;  
 }  
  
 public double getWidth() {  
 return width;  
 }  
  
 public double getHeight() {  
 return height;  
 }  
  
 public double getArea() {  
 return width \* height;  
 }  
  
 @Override  
 public int compareTo(Rectangle other) {  
 return Double.compare(this.getArea(), other.getArea());  
 }  
  
 @Override  
 public String toString() {  
 return "Rectangle{width=" + width + ", height=" + height + ", area=" + getArea() + "}";  
 }  
}*

***Driver.java***

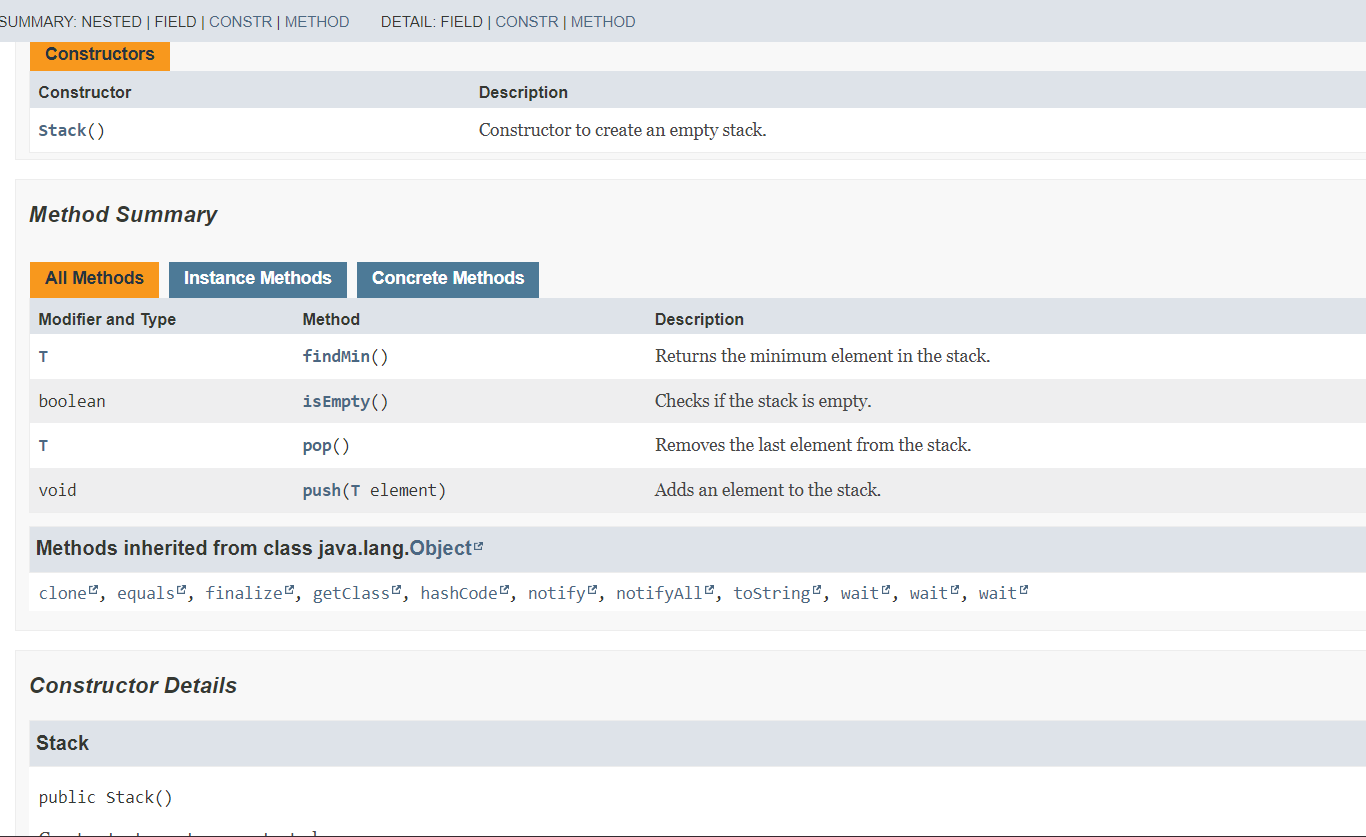
*package KI\_306.Petruk.Lab6;  
  
public class Driver {  
 public static void main(String[] args) {  
 // Example usage with Person  
 Stack<Person> personStack = new Stack<>();  
 personStack.push(new Person("Alice", 30));  
 personStack.push(new Person("Bob", 25));  
 personStack.push(new Person("Charlie", 35));  
 personStack.push(new Person("Diana", 20));  
  
 System.out.println("Minimum element in the person stack: " + personStack.findMin());  
 System.out.println("Removed from stack: " + personStack.pop());  
 System.out.println("Minimum element after removal: " + personStack.findMin());  
  
 // Example usage with Rectangle  
 Stack<Rectangle> rectangleStack = new Stack<>();  
 rectangleStack.push(new Rectangle(4, 5)); // Area = 20  
 rectangleStack.push(new Rectangle(3, 6)); // Area = 18  
 rectangleStack.push(new Rectangle(2, 8)); // Area = 16  
 rectangleStack.push(new Rectangle(1, 10)); // Area = 10  
  
 System.out.println("Minimum element in the rectangle stack: " + rectangleStack.findMin());  
 System.out.println("Removed from stack: " + rectangleStack.pop());  
 System.out.println("Minimum element after removal: " + rectangleStack.findMin());  
 }  
}*



**Рис.1 Вивід у консоль**



**Рис.2.1 Фрагмент згенерованої документації**



**Рис.2.2 Фрагмент згенерованої документації**

**Висновок:** На лабораторній роботі я оволодів навиками параметризованого програмування мовою Java.