

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»

**Виконав:**

студент групи КІ-307

Возний А. О.

**Перевірив:**

доцент кафедри ЕОМ

Іванов Ю. С.

Львів – 2023

**Мета роботи:** 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 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.

**Варіант завдання:** Конвеєр

**Лістинг програми:**

***Клас Conveyor***

```
package ki307.voznyi.lab6;

import java.util.ArrayList;
import java.util.List;

/**
 * Параметризований клас "Конвеєр".
 *
 * @param <T> Тип елементів конвеєра, які реалізують інтерфейс Comparable.
 */
public class Conveyor<T extends Comparable<T>> {
    private List<T> elements = new ArrayList<>();

    /**
     * Додати елемент до конвеєра.
     */
}
```

```

    * @param item Елемент, який додається до конвеєра.
    */
    public void addItem(T item) {
        elements.add(item);
    }

    /**
     * Вийняти елемент з конвеєра.
     *
     * @return Перший елемент конвеєра, або null, якщо конвеєр порожній.
     */
    public T removeItem() {
        if (!elements.isEmpty()) {
            return elements.remove(0);
        } else {
            return null;
        }
    }

    /**
     * Пошук мінімального елементу у конвеєрі.
     *
     * @return Мінімальний елемент у конвеєрі, або null, якщо конвеєр порожній.
     */
    public T processEven() {
        if (!elements.isEmpty()) {
            T minElement = elements.get(0);
            for (T element : elements) {
                if (element.compareTo(minElement) < 0) {
                    minElement = element;
                }
            }
            return minElement;
        } else {
            return null;
        }
    }

    /**
     * Отримати розмір конвеєра.
     *
     * @return Кількість елементів у конвеєрі.
     */
    public int getSize() {
        return elements.size();
    }

    /**
     * Отримати всі елементи конвеєра.
     *
     * @return Список усіх елементів конвеєра.
     */
    public List<T> getAllElements() {
        return elements;
    }
}

```

## Клас *ConveyorApp*

```
package ki307.voznyi.lab6;

/**
 * Додатковий клас для демонстрації використання класу Conveyor у програмі.
 */
public class ConveyorApp {
    public static void main(String[] args) {
        // Створення конвеєра для цілих чисел
        Conveyor<Integer> integerConveyor = new Conveyor<>();

        // Додавання елементів до конвеєра
        integerConveyor.addItem(10);
        integerConveyor.addItem(30);
        integerConveyor.addItem(20);

        // Парний варіант - пошук мінімального елементу
        System.out.println("Min Element: " + integerConveyor.processEven());

        // Виймання елементу з конвеєра
        System.out.println("Removed Element: " + integerConveyor.removeItem());

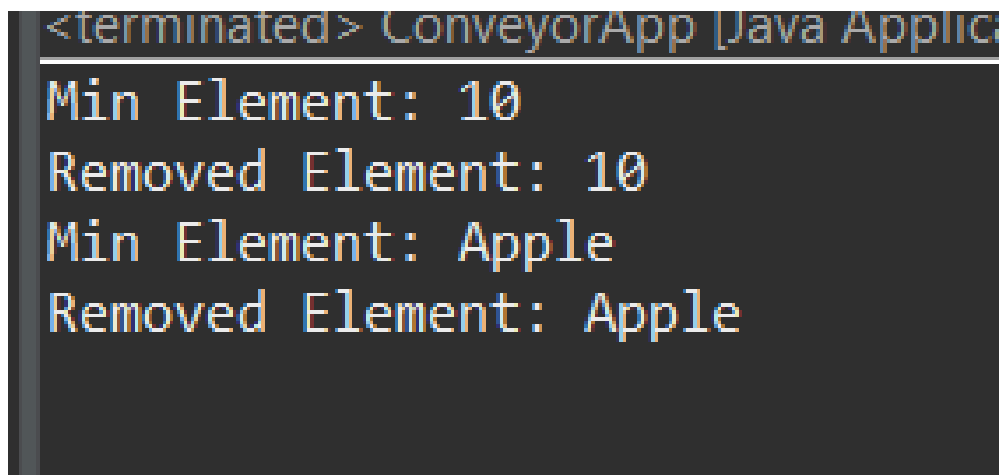
        // Створення конвеєра для рядків
        Conveyor<String> stringConveyor = new Conveyor<>();

        // Додавання елементів до конвеєра
        stringConveyor.addItem("Apple");
        stringConveyor.addItem("Banana");
        stringConveyor.addItem("Orange");

        // Парний варіант - пошук мінімального елементу
        System.out.println("Min Element: " + stringConveyor.processEven());

        // Виймання елементу з конвеєра
        System.out.println("Removed Element: " + stringConveyor.removeItem());
    }
}
```

Результат виконання програми:



```
<terminated> ConveyorApp [Java Applic...
Min Element: 10
Removed Element: 10
Min Element: Apple
Removed Element: Apple
```

Рис. 1. Результат виконання програми

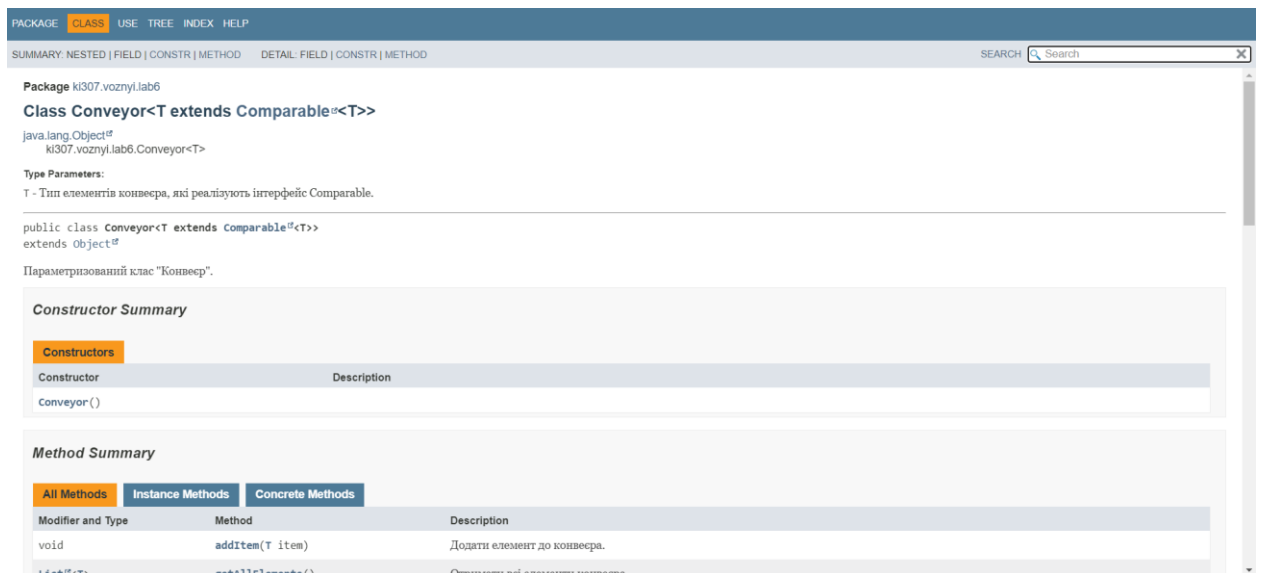


Рис. 2. Згенерована документація до класу Клас Conveyor

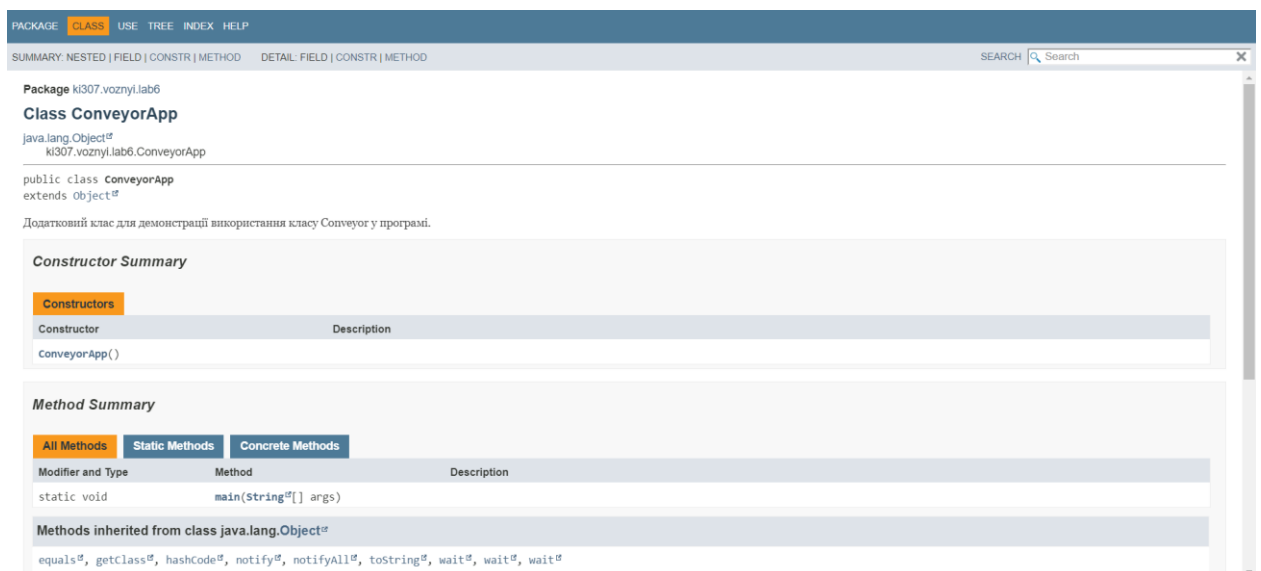


Рис. 3. Згенерована документація до класу ConveyorApp

### ***Відповіді 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:** під час виконання цієї лабораторної роботи я оволодів навиками параметризованого програмування мовою Java.