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 25

Виконав:

ст. групи КІ-306

Тимків Н.В.

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 №25 – « Сховище товарів »
GitHub Repository:

https://github.com/NazarTymkiiv/CPPT_Tymkiv_NV_KI-36_2.git

Хід роботи

Код програми:
StorageDriver.java

```
package KI.Tymkiv.Lab6;

/**
 * Клас для тестування параметризованого сховища товарів.
 */
public class StorageDriver {
    public static void main(String[] args) {
        Storage<Product> productStorage = new Storage<>();

        // Створення та додавання товарів
        productStorage.addItem(new Product("Телефон", 299));
        productStorage.addItem(new Product("Ноутбук",599));
        productStorage.addItem(new Product("Навушники",199));
        productStorage.addItem(new Product( "Планшет", 899));

        // Пошук найдорожчого товару
        System.out.println("Найдорожчий товар: " +
productStorage.findMaxItem());

        // Перевірка роботи інших методів
        Product laptopProduct = new Product("Ноутбук",599);
        System.out.println("Чи є ноутбук у сховищі? " +
productStorage.containsItem(laptopProduct));
        productStorage.addItem(laptopProduct);

        // Виведення всіх товарів
        System.out.println("Всі товари у сховищі: ");
        System.out.println(productStorage);
```

```
        // Видалення товару
        productStorage.removeItem(laptopProduct);
        System.out.println("Список товарів після видалення ноутбука: ");
        System.out.println(productStorage);

        // Додавання нового товару
        productStorage.addItem(new Product("Смарт-годинник", 799));
        System.out.println("Оновлений список товарів: ");
        System.out.println(productStorage);

        // Очищення сховища
        productStorage.clear();
        System.out.println("Кількість товарів після очищення: " +
productStorage.getItemCount());
    }
}
```

Storage.java

```
package KI.Tymkiv.Lab6;

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

/**
 * Клас, що представляє параметризоване сховище товарів.
 * @param <T> Тип об'єктів, що зберігаються в сховищі.
 */
public class Storage<T extends Comparable<T>> {
    private List<T> items;

    /**
     * Конструктор, що ініціалізує порожнє сховище товарів.
     */
    public Storage() {
        this.items = new ArrayList<>();
    }

    /**
     * Додає елемент до сховища.
     * @param item Елемент, що додається.
     */
    public void addItem(T item) {
        items.add(item);
    }

    /**
     * Видаляє елемент зі сховища.
     * @param item Елемент, що видаляється.
     */
    public void removeItem(T item) {
        if(!items.remove(item)) {
            throw new NoSuchElementException("Елемент не знайдено!");
        }
    }

    /**
     * Перевіряє наявність елемента в сховищі.
     * @param item Елемент для перевірки.
     * @return true якщо елемент є в сховищі, інакше false.
     */
    public boolean containsItem(T item) {
        return items.contains(item);
    }
```

```
    /**
     * Очищає всі елементи зі сховища.
     */
    public void clear() {
        items.clear();
    }

    /**
     * Повертає кількість елементів у сховищі.
     * @return Кількість елементів.
     */
    public int getItemCount() {
        return items.size();
    }

    @Override
    public String toString() {
        StringBuilder sb = new StringBuilder();
        sb.append('[');
        for (int i = 0; i < items.size(); i++) {
            sb.append(items.get(i));
            if(i != items.size()-1) {
                sb.append(", ");
            }
        }
        sb.append(']');
        return sb.toString();
    }

    /**
     * Повертає максимальний елемент у сховищі.
     * @return Максимальний елемент.
     */
    public T findMaxItem() {
        if (items.isEmpty()) {
            System.out.println("Сховище порожнє.");
            return null;
        }
        T maxItem = items.get(0);
        for (T item : items) {
            if (item.compareTo(maxItem) > 0) {
                maxItem = item;
            }
        }
        return maxItem;
    }
}
```

Product.java

```
package KI.Tymkiv.Lab6;

/**
 * Клас, що представляє товар.
 * Реалізує інтерфейс Comparable для можливості порівняння товарів за ціною.
 */
public class Product implements Comparable<Product> {
    private String name;
    private int price;


    /**
     * Конструктор для створення товару.
     * @param price ціна товару
     * @param name назва товару
     */
```

```
    public Product(String name, int price) {
        this.name = name;
        this.price = price;
    }

    /**
     * Повертає назву товару.
     * @return назва товару
     */
    public String getName() {
        return name;
    }

    /**
     * Встановлює назву товару.
     * @param name нова назва товару
     */
    public void setName(String name) {
        this.name = name;
    }

    /**
     * Повертає ціну товару.
     * @return ціна товару
     */
    public int getPrice() {
        return price;
    }

    /**
     * Встановлює ціну товару.
     * @param price нова ціна товару
     */
    public void setPrice(int price) {
        this.price = price;
    }

    @Override
    public int compareTo(Product other) {
        // Порівнюємо товари за ціною
        return Integer.compare(this.price, other.price);
    }

    @Override
    public String toString() {
        return "Product{name='" + name + '\'' + ", price=" + price + '}';
    }
}
```

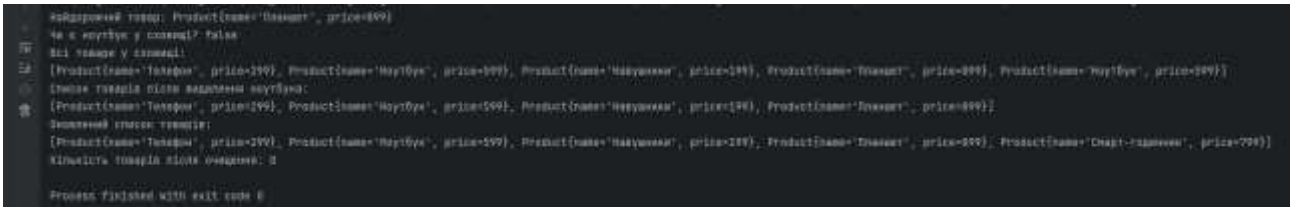

Рис.1 Вивід у консоль

Package KI.Tymkiv.Lab6

package KI.Tymkiv.Lab6

Classes

Class	Description
Product	Клас, що представляє товар.
Storage<T extends Comparable<T>>	Клас, що представляє параметризоване сховище товарів.
StorageDriver	Клас для тестування параметризованого сховища товарів.

Рис.2.1 Фрагмент згенерованої документації

Package KI.Tymkiv.Lab6

Class Storage<T extends Comparable<T>>

java.lang.Object
  KI.Tymkiv.Lab6.Storage<T>

Type Parameters:

T - Тип об'єктів, що зберігаються в сховищі.

```
public class Storage<T extends Comparable<T>>
extends Object
```

Клас, що представляє параметризоване сховище товарів.

Constructor Summary

Constructors

Constructor	Description
Storage()	Конструктор, що ініціалізує порожнє сховище товарів.

Method Summary

All Methods | **Instance Methods** | **Concrete Methods**

Modifier and Type	Method	Description
void	addItem(T item)	Додає елемент до сховища.
void	clear()	Очищає всі елементи зі сховища.
boolean	containsItem(T item)	Перевіряє наявність елемента в сховищі.
T	findMaxItem()	Повертає максимальний елемент у сховищі.
int	getItemCount()	Повертає кількість елементів у сховищі.
void	removeItem(T item)	Видаляє елемент зі сховища.
String	toString()	

Рис.2.2 Фрагмент згенерованої документації

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