

Міністерство освіти і науки України

Національний університет “Львівська політехніка”

Кафедра ЕОМ



Звіт

З лабораторної роботи №6

Варіант – 3

З дисципліни: «Кросплатформні засоби програмування»

На тему: «Параметризоване програмування»

Виконав: ст. гр. КІ-306

Братівник Д. А.

Прийняв:

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: Побутовий пакет

Код програми:

Файл Data.java:

```
package org.example;
/**
 * Інтерфейс, що представляє об'єкти, які можуть бути збережені та
 * порівнювані за їхньою величиною.
 */
public interface Data extends Comparable<Data> {
    /**
     * Повертає розмір об'єкта.
     *
     * @return розмір об'єкта.
     */
    int getSize();
    /**
     * Виводить інформацію про об'єкт.
     */
    void print();
}
```

Файл Householditem.java:

```
package org.example;
```

```

/**
 * Клас, що представляє побутовий предмет.
 * Кожен побутовий предмет має назву (itemName) та розмір (size).
 */
public class HouseholdItem implements Data {
    /**
     * Назва побутового предмета.
     */
    private String itemName;
    /**
     * Розмір побутового предмета.
     */
    private int size;
    /**
     * Конструктор, що ініціалізує назву та розмір побутового
    предмета.
     *
     * @param name назва побутового предмета.
     * @param size розмір побутового предмета.
     */
    public HouseholdItem(String name, int size) {
        this.itemName = name;
        this.size = size;
    }
    /**
     * Повертає назву побутового предмета.
     *
     * @return назва побутового предмета.
     */
    public String getItemName() {
        return itemName;
    }
    /**
     * Повертає розмір побутового предмета.
     *
     * @return розмір побутового предмета.
     */
    public int getSize() {
        return size;
    }
    /**
     * Порівнює побутові предмети за їхнім розміром для
    сортування.
     *
     * @param otherItem інший побутовий предмет для порівняння.
     * @return від'ємне число, нуль або додатне число, якщо цей
    об'єкт менший, рівний або більший за інший предмет.
     */
    @Override
    public int compareTo(Data otherItem) {
        return Integer.compare(this.size, otherItem.getSize());
    }
    /**
     * Виводить інформацію про побутовий предмет (назва та розмір)

```

```

у консоль.
    */
    public void print() {
        System.out.println("Item: " + itemName + ", Size: " +
size);
    }
}

```

Файл HouseholdPackage.java:

```

package org.example;

import java.util.ArrayList;
/**
 * Клас, що представляє пакет для зберігання побутових предметів.
 * Пакет може містити об'єкти, які є підкласами класу
HouseholdItem.
 *
 * @param <T> тип об'єктів, які можуть бути додані до пакету.
 */
public class HouseholdPackage<T extends Data> {
    /**
     * Список елементів у пакеті.
     */
    private ArrayList<T> items;
    /**
     * Конструктор за замовчуванням, створює порожній пакет.
     */
    public HouseholdPackage() {
        items = new ArrayList<>();
    }
    /**
     * Пошук найбільшого об'єкта у пакеті за їхньою величиною.
     *
     * @return найбільший об'єкт у пакеті або null, якщо пакет
порожній.
     */
    public T findMax() {
        if (!items.isEmpty()) {
            T max = items.get(0);
            for (int i = 1; i < items.size(); i++) {
                if (items.get(i).compareTo(max) > 0) {
                    max = items.get(i);
                }
            }
            return max;
        }
        return null;
    }
    /**
     * Додає об'єкт до пакету і виводить повідомлення про успішне
додавання.
     *
     * @param item об'єкт, який додається до пакету.
     */

```

```

    public void addItem(T item) {
        items.add(item);
        System.out.print("Item added: ");
        item.print();
    }
    /**
     * Видаляє об'єкт із пакету за вказаним індексом і виводить
     * повідомлення про успішне видалення.
     *
     * @param index індекс об'єкта, який потрібно видалити.
     */
    public void removeItem(int index) {
        if (index >= 0 && index < items.size()) {
            T removedItem = items.remove(index);
            System.out.print("Item removed: ");
            removedItem.print();
        } else {
            System.out.println("Invalid index. Item not
removed.");
        }
    }
}

```

Файл HouseholdPackageDriver.java:

```

package org.example;

/**
 * Клас, який демонструє використання класів HouseholdItem та
 * HouseholdPackage.
 */

public class HouseholdPackageDriver {
    public static void main(String[] args) {
        HouseholdPackage<HouseholdItem> package1 = new
HouseholdPackage<>();
        HouseholdPackage<HouseholdItem> package2 = new
HouseholdPackage<>();

        HouseholdItem item1 = new HouseholdItem("Sofa", 100);
        HouseholdItem item2 = new HouseholdItem("Microwave", 30);
        HouseholdItem item3 = new HouseholdItem("TV", 50);
        HouseholdItem item4 = new HouseholdItem("Bed", 80);

        package1.addItem(item1);
        package1.addItem(item2);
        package1.addItem(item3);
        System.out.println();
        package2.addItem(item4);
        System.out.println();
        HouseholdItem maxItem1 = package1.findMax();
        if (maxItem1 != null) {
            System.out.print("The largest item in package 1: ");
            maxItem1.print();
        } else {

```

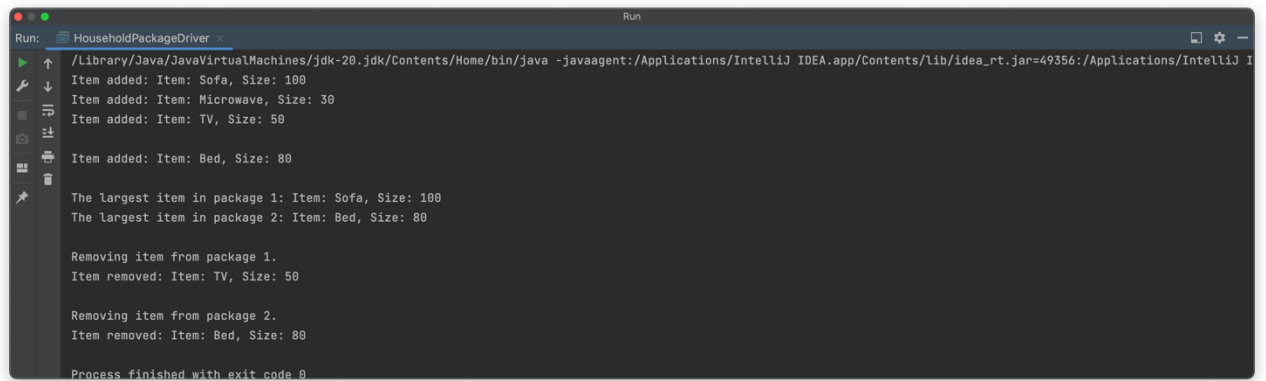
```

        System.out.println("Package 1 is empty.");
    }

    HouseholdItem maxItem2 = package2.findMax();
    if (maxItem2 != null) {
        System.out.print("The largest item in package 2: ");
        maxItem2.print();
    } else {
        System.out.println("Package 2 is empty.");
    }
    System.out.println();
    System.out.println("Removing item from package 1.");
    package1.removeItem(2);
    System.out.println();
    System.out.println("Removing item from package 2.");
    package2.removeItem(0);
}
}

```

Результати роботи програми:



```

Run: HouseholdPackageDriver
/Library/Java/JavaVirtualMachines/jdk-20.jdk/Contents/Home/bin/java -javaagent:/Applications/IntelliJ IDEA.app/Contents/lib/idea_rt.jar=49356:/Applications/IntelliJ I
Item added: Item: Sofa, Size: 100
Item added: Item: Microwave, Size: 30
Item added: Item: TV, Size: 50

Item added: Item: Bed, Size: 80

The largest item in package 1: Item: Sofa, Size: 100
The largest item in package 2: Item: Bed, Size: 80

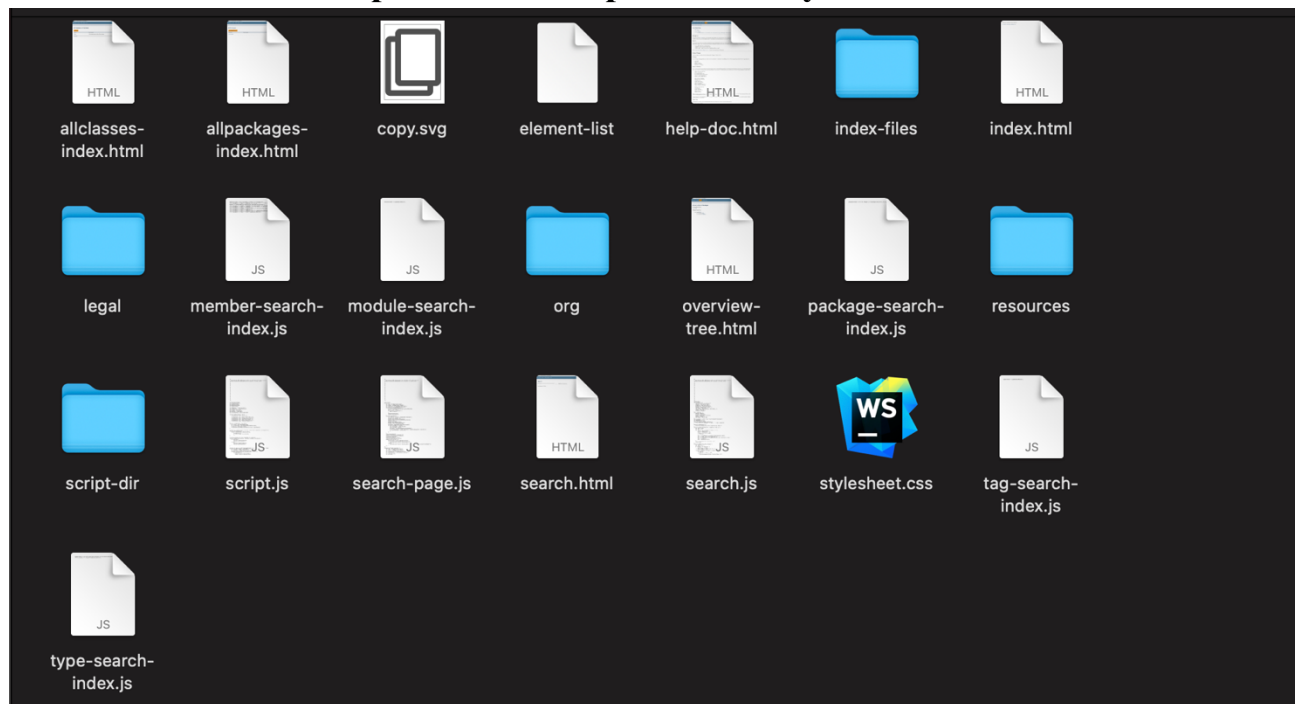
Removing item from package 1.
Item removed: Item: TV, Size: 50

Removing item from package 2.
Item removed: Item: Bed, Size: 80

Process finished with exit code 0

```

Фрагмент згенерованої документації



PACKAGE	CLASS	TREE	INDEX	HELP
---------	-------	------	-------	------

PACKAGE: DESCRIPTION | RELATED PACKAGES | CLASSES AND INTERFACES

SEARCH

Package org.example

package org.example

All Classes and Interfaces

Interfaces

Classes

Class	Description
Data	Інтерфейс, що представляє об'єкти, які можуть бути збережені та порівнювані за їхньою величиною.
HouseholdItem	Клас, що представляє побутовий предмет.
HouseholdPackage<T extends Data>	Клас, що представляє пакет для зберігання побутових предметів.
HouseholdPackageDriver	Клас, який демонструє використання класів HouseholdItem та HouseholdPackage.

Відповіді на контрольні запитання

1. Дайте визначення терміну «параметризоване програмування». - це підхід до програмування, що дозволяє створювати класи і методи, які можна використовувати з різними типами даних, надаючи більшу гнучкість і безпеку типів у програмах.

2. Розкрийте синтаксис визначення простого параметризованого класу. - `public class НазваКласу<параметризованийТип> {`

`// Тіло класу }`

3. Розкрийте синтаксис створення об'єкту параметризованого класу. - `НазваКласу<перелікТипів> зміннаКласу = new`

`НазваКласу<перелікТипів>(параметри);`

4. Розкрийте синтаксис визначення параметризованого методу.

- `public <параметризованийТип> типПовернення назваМетоду(параметри) {`

`// Тіло методу }`

5. Розкрийте синтаксис виклику параметризованого методу.

- `(НазваКласу|НазваОб'єкту).<перелікТипів>назваМетоду(параметри);`

6. Яку роль відіграє встановлення обмежень для змінних типів?

- дозволяє заборонити використання деяких типів або вимагати, щоб тип підставлений за замовчуванням був підкласом або реалізував певний інтерфейс.

7. Як встановити обмеження для змінних типів?
- за допомогою ключового слова `extends` для суперкласу або інтерфейсу, від яких має походити реальний тип.

8. Розкрийте правила спадкування параметризованих типів.

- - Всі класи, створені з параметризованого класу, незалежні один від одного.
- - Зазвичай немає залежності між класами, створеними з різними параметрами типів.

9. Яке призначення підстановочних типів?

- використовуються для забезпечення безпеки типів при використанні параметризованих класів та методів. Вони дозволяють визначити, які типи можна використовувати замість параметризованих типів.

10. Застосування підстановочних типів.

- `<?>` (unbounded wildcard) дозволяє читати об'єкти з колекції без змінення її.

- - `<? extends Тип>` (bounded wildcard) дозволяє читати об'єкти з колекції, але забороняє додавання в неї нових об'єктів.
- - `<? super Тип>` (lower bounded wildcard) дозволяє додавати об'єкти в колекцію, але забороняє їх читання.

Висновок

У ході виконання даної лабораторної роботи, я отримав важливі навички параметризованого програмування мовою Java. Ознайомився з різними аспектами мови, такими як використання параметрів у методах, створення та використання класів та інтерфейсів.