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»

Виконала: ст. гр. КІ-305

Романюк М. Р.

Прийняв: доцент кафедри ЕОМ

Іванов Ю. С.

Львів 2023

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.

Мій варіант №22. Валіза

Код програми:

Файл Valise.java

```java
package KI305RomaniukLab6;

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

/**
 * Клас, що представляє валізу.
 */
public class Valise<T extends Item> {
    private List<T> items;
    List<String> names;

    public Valise() {
        items = new ArrayList<>();
        names = new ArrayList<>();
    }

    public List<String> getListOfNames() {
        return names;
    }

    public void addItem(T item) {
        items.add(item);
        names.add(item.getName());
    }

    public void removeItem(int index) {
        if (index >= 0 && index < items.size()) {
```

```
            names.remove(items.get(index).getName());
            items.remove(index);
        }
    }

    /**
     * Пошук мінімального предмета в валізі.
     *
     *
     * @return Мінімальний предмет в валізі.
     */
    public T findMinItem(boolean b) {
        if (!items.isEmpty()) {
            T minItem = items.get(0);
            for (T item : items) {
                if (item.compareTo(minItem) < 0) {
                    minItem = item;
                }
            }
            return minItem;
        }
        return null;

    }
}
```

Файл Item.java:

```
package KI305RomaniukLab6;

/**
 * Клас, що представляє предмет.
 */
public abstract class Item implements Comparable<Item> {
    private String name;
    private double weight;

    public Item(String name, double weight) {
        this.name = name;
        this.weight = weight;
    }

    public String getName() {
        return name;
    }

    public double getWeight() {
        return weight;
    }

    @Override
    public int compareTo(Item o) {
        return Double.compare(this.weight, o.weight);
    }

    /**
     * Абстрактний метод для отримання опису предмета.
     *
     * @return Опис предмета.
     */
    public abstract String getDescription();
}
```

Файл ValuableItem.java:

```
package KI305RomaniukLab6;

/**
 * Підклас для цінного предмета.
 */
public class ValuableItem extends Item {
    private double value;

    public ValuableItem(String name, double weight, double value) {
        super(name, weight);
        this.value = value;
    }

    public double getValue() {
        return value;
    }

    @Override
    public String getDescription() {
        return "Предмет: " + getName() + " (вага: " + getWeight() + " кг,
вартість: " + value + " грн)";
    }
}
```

Файл ValiseDriver.java:

```
package KI305RomaniukLab6;

public class ValiseDriver {
    public static void main(String[] args) {
        Valise<Item> valise = new Valise<>();

        valise.addItem(new ValuableItem("Дорогий смартфон", 0.2, 12000));
        valise.addItem(new ValuableItem("Планшет", 0.5, 8000));
        valise.addItem(new ValuableItem("Книга 'Java Programming'", 0.5, 500));
        valise.addItem(new ValuableItem("Ноутбук", 3.5, 20000));
        valise.addItem(new ValuableItem("Шампунь", 0.2, 100));
        valise.addItem(new ValuableItem("Щітка для волосся", 0.1, 90));
        valise.addItem(new ValuableItem("Фен", 1.5, 500));
        valise.addItem(new ValuableItem("Фен1", 1.5, 500));
        Item minItem = valise.findMinItem(true);

        if (minItem != null) {
            System.out.println("Мінімальний предмет в валізі: " +
minItem.getDescription());
        } else {
            System.out.println("Валіза порожня. Мінімальний предмет не
знайдено.");
        }
        System.out.println(valise.getListOfNames());
    }
}
```

Результати роботи програми:

```
Мінімальний предмет в валізі: Предмет: Щітка для волосся (вага: 0.1 кг, вартість: 90.0 грн)
[Дорогий смартфон, Планшет, Книга 'Java Programming', Ноутбук, Шампунь, Щітка для волосся, Фен, Фен1]

Process finished with exit code 0
```

Згенерована документація:

Package KI305RomaniukLab6

package KI305RomaniukLab6

Classes

Class	Description
Item	Клас, що представляє предмет.
Valise<T extends Item>	Клас, що представляє валізу.
ValiseDriver	
ValuableItem	Підклас для цінного предмета.

Direct Known Subclasses:
ValuableItem

public abstract class Item
extends Object
implements Comparable<Item>

Клас, що представляє предмет.

Constructor Summary

Constructors

Constructor	Description
Item(String name, double weight)	

Method Summary

All Methods | Instance Methods | Abstract Methods | Concrete Methods

Modifier and Type	Method	Description
int	compareTo(Item o)	
abstract String	getDescription()	Абстрактний метод для отримання опису предмета.
String	getName()	
double	getWeight()	

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Package KI305RomaniukLab6

Class Valise<T extends Item>

java.lang.Object
    KI305RomaniukLab6.Valise<T>

public class **Valise<T extends Item>**
extends Object

Клас, що представляє валізу.

Constructor Summary

Constructors

Constructor	Description
Valise()	

Method Summary

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

Modifier and Type	Method	Description
void	addItem(T item)	
T	findMinItem(boolean b)	Пошук мінімального предмета в валізі.
List<String>	getListOfNames()	
void	removeItem(int index)	

Methods inherited from class java.lang.Object

Package KI305RomaniukLab6

Class ValiseDriver

java.lang.Object
    KI305RomaniukLab6.ValiseDriver

public class **ValiseDriver**
extends Object

Constructor Summary

Constructors

Constructor	Description
ValiseDriver()	

Method Summary

All Methods | **Static Methods** | **Concrete Methods**

Modifier and Type	Method	Description
static void	main(String[] args)	

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Details

Package KI305RomaniukLab6

Class ValuableItem

java.lang.Object
    KI305RomaniukLab6.Item
        KI305RomaniukLab6.ValuableItem

All Implemented Interfaces:

Comparable<Item>

```
public class ValuableItem
extends Item
```

Підклас для цінного предмета.

Constructor Summary

Constructors

Constructor	Description
ValuableItem(String name, double weight, double value)	

Method Summary

All Methods | Instance Methods | Concrete Methods

Modifier and Type	Method	Description
String	getDescription()	Абстрактний метод для отримання опису предмета.
double	getValue()	

Відповіді на контрольні запитання

1. Дайте визначення терміну «параметризоване програмування»:

Параметризоване програмування (або генеричне програмування) - це підхід у програмуванні, коли можна створювати класи, інтерфейси або методи з параметрами типу для забезпечення більшої безпеки типів і перевикористання коду.

2. Розкрийте синтаксис визначення простого параметризованого класу:

```
public class MyGenericClass<T> {
    // Код класу з параметром типу T
}
```

3. Розкрийте синтаксис створення об’єкту параметризованого класу:

```
MyGenericClass<Integer> myObject = new MyGenericClass<>();
```

4. Розкрийте синтаксис визначення параметризованого методу:

```
public <T> void myGenericMethod(T parameter) {
    // Код методу з параметром типу T
}
```

5. Розкрийте синтаксис виклику параметризованого методу:

```
Integer myParameter = 42;
myGenericMethod(myParameter);
```

6. Яку роль відіграє встановлення обмежень для змінних типів?

Встановлення обмежень для змінних типів дозволяє обмежити типи, які можна використовувати як параметри типу в параметризованих класах або методах,

щоб гарантувати валідність операцій та уникнути помилок.

7. Як встановити обмеження для змінних типів?

Обмеження для змінних типу встановлюються за допомогою ключового слова extends або super. Наприклад:

```
public class MyGenericClass<T extends SomeBaseClass> {
    // Код з обмеженням для типу T
}
```

8. Розкрийте правила спадкування параметризованих типів:

Параметризовані типи можуть успадковувати інші параметризовані типи або непараметризовані типи. Параметризований тип може також мати власні параметризовані параметри типу.

9. Яке призначення підстановочних типів?

Підстановочні типи (wildcard types) використовуються в Java для створення загальних типів або методів, які можуть працювати з різними типами даних без конкретного вказування типу. Вони дозволяють підставити конкретний тип при використанні коду.

10. Застосування підстановочних типів:

Підстановочні типи зазвичай використовуються для роботи з колекціями різних типів даних, створення загальних алгоритмів та функцій, які можуть працювати з різними типами, та підтримувати поліморфізм. Вони зроблюють код більш гнучким та загальним.

Висновок: оволоділа навичками параметризованого програмування мовою Java.