Міністерство освіти і науки України

Національний університет „Львівська політехніка”

Звіт з лабораторної роботи №7

з дисципліни «Кросплатформні засоби програмування»

на тему: «ПАРАМЕТРИЗОВАНЕ ПРОГРАМУВАННЯ»

Виконала: ст.групи КІ-36

Гульчевська Є. Л.

Прийняв, та перевірив :

Іванов Ю. С.

Львів 2022

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 83 екземплярі розробленого класуконтейнеру. 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. Скласти звіт про виконану роботу з приведенням тексту програми, результату її виконання та фрагменту згенерованої документації.
4. Дати відповідь на контрольні запитання.

Варіант №28 : Антресолі

Хід роботи:

Код програми:

Main.java

```
package KI_36_Hulchevska_Lab7;

import java.util.ArrayList;
public class Main {
    public static void main(String[] args) {
        Entresol <? super Things> MyEntresol = new Entresol<Things>(); // створення
нового обєкту калсу

        MyEntresol.AddData(new ProThing("Manicure Box", 65, 30, 13, "white"));
        MyEntresol.AddData(new Thing("Docunents", 34, 35, 5));
        MyEntresol.AddData(new Thing("Gloves", 19, 8, 6));
        MyEntresol.AddData(new Thing("Ruler", 20, 1, 1));
        MyEntresol.AddData(new ProThing("Headphone Box", 20, 14, 6, "black"));
```

```
        Things res = MyEntresol.findMin();
        System.out.print("The thing with the smallest capacity: \n");
        res.print();
        System.out.print("The capacity is: "+res.get_ThingCapacity()+"m^3");

    }
}
```

Entresol.java

```
package KI_36_Hulchevska_Lab7;
import java.util.ArrayList;
class Entresol <T extends Things> {  //параметризований клас

    private ArrayList<T> arr;

    public Entresol(){  //конструктор
        arr = new ArrayList<T>();
    }

    public T findMin(){
        if (!arr.isEmpty())
        {
            T min = arr.get(0);
            for (int i=1; i< arr.size(); i++)
            {
                if ( arr.get(i).compareTo(min) < 0 )
                    min = arr.get(i);
            }
            return min;
        }
        return null;
    }

    public void AddData(T data){
        arr.add(data);
        System.out.print("Element added: ");
        data.print();
    }
}
```

Things.java

```
package KI_36_Hulchevska_Lab7;
// Цей інтерфейс який описує 2 методи

interface Things extends Comparable<Things>{
    public int get_ThingCapacity();
    public void print();
}
```

ProThing.java

```
package KI_36_Hulchevska_Lab7;

// Цей клас моделює

class ProThing implements Things
{
    private String Thing;
    private int ThingLength;
    private int ThingWidth;
    private int ThingHeight;
    private String Colour;
```

```
    public ProThing(String Thing, int ThingLength, int ThingWidth, int ThingHeight,
String Colour){ // конструктор
        this.Thing = Thing;
        this.ThingLength = ThingLength;
        this.ThingWidth = ThingWidth;
        this.ThingHeight = ThingHeight;
        this.Colour = Colour;
    }


    // Метод повертає предмет
    public String get_Thing(){
        return Thing;
    }

    // Метод встановлює значення поля Thing
    public void set_Thing(String Th){
        this.Thing = Th;
    }

    // Метод повертає предмет
    public int get_ThingLength(){
        return ThingLength;
    }

    // Метод встановлює значення поля Thing
    public void set_ThingLength(int TL){
        this.ThingLength = TL;
    }

    // Метод повертає предмет
    public int get_ThingWidth(){
        return ThingWidth;
    }

    // Метод встановлює значення поля Thing
    public void set_ThingWidth(int TW){
        this.ThingWidth = TW;
    }

    // Метод повертає предмет
    public int get_ThingHeight(){
        return ThingHeight;
    }

    // Метод встановлює значення поля Thing
    public void set_ThingHeight(int TH){
        this.ThingHeight = TH;
    }

    // Метод повертає предмет
    public String get_Colour(){
        return Colour;
    }

    // Метод встановлює значення поля Thing
    public void set_Colour(String Cr){
        this.Colour = Cr;
    }

    // Метод повертає об'єм предмету

    public int get_ThingCapacity(){
        return ThingHeight*ThingLength*ThingWidth;
```

```
    }

    public int compareTo(Things p){
        Integer s = get_ThingCapacity();
        return s.compareTo(p.get_ThingCapacity());
    }

    // Вивід інформації про слово

    public void print(){
        System.out.print("Thing: " + Thing + ", Size(LHW): " + ThingLength + " " +
ThingHeight + " " + ThingWidth + ", Colour: " + Colour + ";\n");
    }
}
```

Thing.java

```
 package KI_36_Hulchevska_Lab7;
//Клас моделює

class Thing implements Things
{
    private String Thing;
    private int ThingLength;
    private int ThingWidth;
    private int ThingHeight;

    public Thing(String Thing, int ThingLength, int ThingWidth, int ThingHeight){ //
конструктор
        this.Thing = Thing;
        this.ThingLength = ThingLength;
        this.ThingWidth = ThingWidth;
        this.ThingHeight = ThingHeight;
    }


    // Метод повертає предмет
    public String get_Thing(){
        return Thing;
    }

    // Метод встановлює значення поля Thing
    public void set_Thing(String Th){
        this.Thing = Th;
    }

    // Метод повертає предмет
    public int get_ThingLength(){
        return ThingLength;
    }

    // Метод встановлює значення поля Thing
    public void set_ThingLength(int TL){
        this.ThingLength = TL;
    }

    // Метод повертає предмет
    public int get_ThingWidth(){
        return ThingWidth;
    }

    // Метод встановлює значення поля Thing
    public void set_ThingWidth(int TW){
        this.ThingWidth = TW;
    }
```

```
    // Метод повертає предмет
    public int get_ThingHeight(){
        return ThingHeight;
    }

    // Метод встановлює значення поля Thing
    public void set_ThingHeight(int TH){
        this.ThingHeight = TH;
    }

    // Метод повертає об'єм предмету

    public int get_ThingCapacity(){
        return ThingHeight*ThingLength*ThingWidth;
    }

    public int compareTo(Things p){
        Integer s = get_ThingCapacity();
        return s.compareTo(p.get_ThingCapacity());
    }

    // Вивід інформації про слово

    public void print(){
        System.out.print("Thing: " + Thing + ", Size(LHW): " + ThingLength + " " +
ThingHeight + " " + ThingWidth +"\n");
    }
}
```

Результат виконання:

```
Element added: Thing: Manicure Box, Size(LHW): 65 13 30, Colour: white;
Element added: Thing: Docunents, Size(LHW): 34 5 35
Element added: Thing: Gloves, Size(LHW): 19 6 8
Element added: Thing: Ruler, Size(LHW): 20 1 1
Element added: Thing: Headphone Box, Size(LHW): 20 6 14, Colour: black;
The thing with the smallest capacity:
Thing: Ruler, Size(LHW): 20 1 1
The capacity is: 20m^3
Process finished with exit code 0
```

Контрольні питання

1. Дайте визначення терміну «параметризоване програмування».

Параметризоване програмування - це такий підхід до опису даних і алгоритмів, який дозволяє їх використовувати з різними типами даних без зміни їх опису.

2. Розкрийте синтаксис визначення простого параметризованого класу.

[public] class НазваКласу <параметризованийТип{,параметризованийТип}> {…}

3. Розкрийте синтаксис створення об’єкту параметризованого класу.

НазваКласу < перелікТипів > = new НазваКласу < перелікТипів > (параметри);

4. Розкрийте синтаксис визначення параметризованого методу.

Модифікатори<параметризованийТип{,параметризованийТип}>типПовернення назваМетоду(параметри);

5. Розкрийте синтаксис виклику параметризованого методу.

Модифікатори<параметризованийТип{,параметризованийТип}>типПовернення назваМетоду(параметри);

6. Яку роль відіграє встановлення обмежень для змінних типів?

Може бути ситуація, коли метод у процесі роботи викликає з-під об’єкта параметризованого типу метод, що визначається у деякому інтерфейсі. У такому випадку немає ніякої гарантії, що цей метод буде реалізований у кожному класі, що передається через змінну типу. Щоб вирішити цю проблему у мові Java можна задати обмеження на множину можливих типів, що можуть бути підставлені замість параметризованого типу.

7. Як встановити обмеження для змінних типів?

Модифікатори<параметризований тип extends обмежуючийТип {& обмежуючий тип} {, параметризований тип extends обмежуючийТип {& обмежуючий тип} }> типПовернення назваМетоду(параметри);

8. Розкрийте правила спадкування параметризованих типів.

1. Всі класи, що утворені з одного і того ж параметризованого класу з використанням різних значень змінних типів є незалежними навіть якщо між цими типами є залежність спадкування.

2. Завжди можна перетворити параметризований клас у «сирий» клас, при роботі з яким захист від некоректного коду є значно слабшим, що дозволяє

здійснювати небезпечні присвоєння об’єктів параметризованого класу об’єктам «сирого» класу

3. Параметризовані класи можуть розширювати або реалізовувати інші параметризовані класи.

9. Яке призначення підстановочних типів?

Підстановочні типи дозволяють враховувати залежності між типами, що виступають параметрами для параметризованих типів. Це в свою чергу дозволяє застосовувати обмеження для параметрів, що підставляються замість параметризованих типів. Завдяки цьому підвищується надійність параметризованого коду, полегшується робота з ним та розділяється використання безпечних методів доступу і небезпечних модифікуючих методів.

10.Застосування підстановочних типів.

Підстановочні типи застосовуються у вигляді параметру типу, що передається у трикутних дужках при утворені реального типу з параметризованого типу, наприклад, у методі main.

Висновок: на даній лабораторній роботі я оволоділа навиками параметризованого програмування мовою Java.