Міністерство освіти і науки України
Національний університет «Львівська політехніка»
Кафедра «Електронних обчислювальних машин»

Звіт
з лабораторної роботи № 7
з дисципліни: «Кросплатформенні засоби програмування»
на тему: «Параметризоване програмування»

Виконав:
студент групи КІ-35
Андрусяк М.В.
Прийняв:
доцент кафедри ЕОМ
Іванов Ю. С.

Львів – 2022

Мета роботи: оволодіти навиками параметризованого програмування мовою Java.

Завдання (варіант № 27)

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

Варіант N-27 - Шухляда

Текст програми

ShuttleApp

```
package Lab7AndrusiakKI35;

public class ShuttleApp {
    public static void main(String[] args) {
        Shuttle<ShuttleObject> shuttle = new Shuttle<>();

        Phone phone = new Phone("IPhone",8000, "X");
        Wallet wallet = new Wallet("Gucci", 2500, "skin");
        Watch watch = new Watch("Rolex", 4000, "divers");

        shuttle.Add(phone);
        shuttle.Add(wallet);
        shuttle.Add(watch);

        System.out.println("All your objects : ");
        shuttle.printAll();
        System.out.println();

        System.out.println("Max element : " + shuttle.findMaxElement()); //
```

```
Iphone X - price 8000
        System.out.println("Min element : " + shuttle.findMinElement()); //
Wallet - price 2500
        System.out.println();

        System.out.println("Sorting from Max to Min : ");
        shuttle.SortMaxToMin();
        shuttle.printAll();
        System.out.println();

        System.out.println("Sorting from Min to Max : ");
        shuttle.SortMinToMax();
        shuttle.printAll();
        System.out.println();

        System.out.println("You remove " + shuttle.Remove(phone));
        shuttle.printAll();
    }
}
```

Shuttle

```
package Lab7AndrusiakKI35;

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

public class Shuttle<T extends ShuttleObject>{
    private List<T> listOfObj;

    public Shuttle() {
        listOfObj = new ArrayList<>();
    }

    public void Add(T element) {
        listOfObj.add(element);
    }

    public T Remove(T element) {
        int index = listOfObj.indexOf(element);
        T currentVariable = listOfObj.get(index);
        listOfObj.remove(element);
        return currentVariable;
    }

    public void SortMinToMax() {
        Comparator<T> comparator = new Comparator<T>(){

            @Override
            public int compare(T o1, T o2) {
                return o1.getValueObj() - o2.getValueObj();
            }
        };

        listOfObj.sort(comparator);
    }

    public void SortMaxToMin() {
        Comparator<T> comparator = new Comparator<T>(){

            @Override
            public int compare(T o1, T o2) {
```

```
                return o2.getValueObj() - o1.getValueObj();
            }
        };

        listOfObj.sort(comparator);
    }

    public T findMinElement() {
        T current = listOfObj.get(0);

        for (T element: listOfObj) {
            if ( current.getValueObj() > element.getValueObj()) {
                current = element;
            }
        }

        return current;
    }

    public T findMaxElement() {
        T current = listOfObj.get(0);

        for (T element: listOfObj) {
            if ( current.getValueObj() < element.getValueObj()) {
                current = element;
            }
        }

        return current;
    }

    public void printAll() {
        for (T element: listOfObj) {
            System.out.println(element);
        }
    }
}
```

ShuttleObject

```
package Lab7AndrusiakKI35;

public class ShuttleObject {
    private String nameObj;
    private int valueObj;

    public ShuttleObject(String nameObj, int valueObj) {
        this.nameObj = nameObj;
        this.valueObj = valueObj;
    }

    public String getNameObj() {
        return nameObj;
    }

    public void setNameObj(String nameObj) {
        this.nameObj = nameObj;
    }

    public int getValueObj() {
        return valueObj;
    }
```

```
    public void setValueObj(int valueObj) {
        this.valueObj = valueObj;
    }

    @Override
    public String toString() {
        return "ShuttleObject{" +
                "nameObj='" + nameObj + '\'' +
                ", value=" + valueObj +
                '}';
    }
}
```

Phone

```
package Lab7AndrusiakKI35;

public class Phone extends ShuttleObject {
    private String model;

    public Phone(String nameObj, int valueObj, String model) {
        super(nameObj, valueObj);
        this.model = model;
    }

    public String getModel() {
        return model;
    }

    public void setModel(String model) {
        this.model = model;
    }

    @Override
    public String toString() {
        return "Phone{" +
                "name = '" + getNameObj() + '\'' +
                ", value = '" + getValueObj() + '\'' +
                ", model = '" + model + '\'' +
                '}';
    }
}
```

Wallet

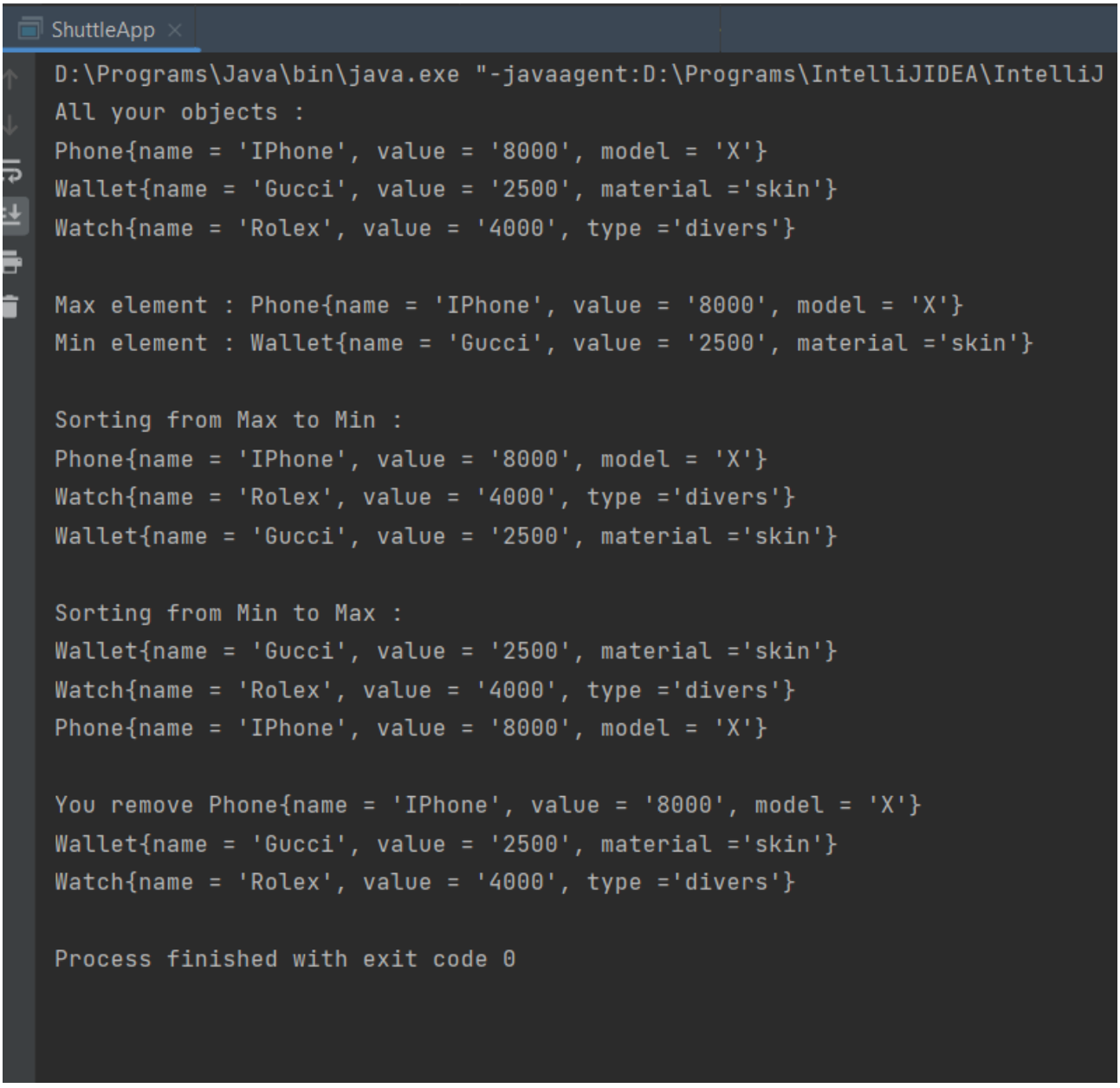
```
package Lab7AndrusiakKI35;

public class Wallet extends ShuttleObject {
    private String material;

    public Wallet(String nameObj, int valueObj, String material) {
        super(nameObj, valueObj);
        this.material = material;
    }

    public String getMaterial() {
        return material;
```

```
    }

    public void setMaterial(String material) {
        this.material = material;
    }

    @Override
    public String toString() {
        return "Wallet{" +
                "name = '" + getNameObj() + '\'' +
                ", value = '" + getValueObj() + '\'' +
                ", material ='" + material + '\'' +
                '}';
    }
}
```

Watch

```
package Lab7AndrusiakKI35;

public class Watch extends ShuttleObject {
    private String type;

    public Watch(String nameObj, int valueObj, String type) {
        super(nameObj, valueObj);
        this.type = type;
    }

    public String getType() {
        return type;
    }

    public void setType(String type) {
        this.type = type;
    }

    @Override
    public String toString() {
        return "Watch{" +
                "name = '" + getNameObj() + '\'' +
                ", value = '" + getValueObj() + '\'' +
                ", type ='" + type + '\'' +
                '}';
    }
}
```

Результат виконання програми



Відповіді на контрольні запитання

Параметризоване програмування є аналогом шаблонів у C++. Воно полягає у написанні коду, що можна багаторазово застосовувати з об’єктами різних класів.

Підстановочні типи були введені у мову Java для збільшення гнучкості жорсткої існуючої системи параметризованих типів. На відміну від неї підстановочні типи дозволяють враховувати залежності між типами, що виступають параметрами для параметризованих типів. Завдяки цьому підвищується надійність параметризованого коду, полегшується робота з ним та розділяється використання безпечних методів доступу і небезпечних модифікуючих методів.

Синтаксис оголошення параметризованого класу:

> *[public] class НазваКласу <параметризованийТип{,параметризованийТип}> {...}*

Висновок : на даній лабораторній роботі я ознайомився зі особливостями параметризованого програмування мови Java . В результаті виконання роботи, я розробив параметризований клас, що реалізує предметну область задану варіантом.