Міністерство освіти і науки України

Національний університет “Львівська політехніка”

Кафедра МКС

Звіт

З лабораторної роботи №7

З дисципліни: «Кросплатформні засоби програмування»

На тему:

«Параметризоване програмування»

Виконав:
ст.гр. КІ-35

Куденчук Владислав

Пр
ийняв:

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

   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.

17. Кошик

Виконання:

Код:

Class BasketDriver

```
package KI_35.Kudenchuk.Lab7;

public class BasketDriver {
    public static void main(String[] args)
    {
        Basket <? super Fruit> basket = new Basket<>();
        basket.AddData(new Pears("Green", 9));
        basket.AddData(new Apples("Red" , 10));
        basket.AddData(new Apples("Green" , 8));
        basket.AddData(new Pears("Yellow" ,10));
        Fruit res = basket.find_by_color("Red");
        System.out.println();
        System.out.print("The amount of red objects: \n");
        System.out.print(res.getAmount());
//        res.print();
    }
}
```

Class Basket

```
package KI_35.Kudenchuk.Lab7;

import java.util.ArrayList;

class Basket <T extends Fruit>
{
    private final ArrayList<T> arr;
    public Basket()
```

```
    {
        arr = new ArrayList<>();
    }
    public T findMax()
    {
        if (!arr.isEmpty())
        {
            T max = arr.get(0);
            for (int i=1; i< arr.size(); i++)
            {
                if ( arr.get(i).compareTo(max) > 0 )
                    max = arr.get(i);
            }
            return max;
        }
        return null;
    }

    public T find_by_color(String color)
    {
        if (!arr.isEmpty())
        {
            T fruit;
            for (int i = 0; i< arr.size(); i++)
            {
                fruit = arr.get(i);
                if (fruit.getColor().equals(color))
                {
                    return fruit;
                }
            }
        }
        return null;
    }
    public void AddData(T fruit)
    {
        arr.add(fruit);
        System.out.print("Fruit added: ");
        fruit.print();
    }
    public void DeleteData(int i)
    {
        arr.remove(i);
    }
}
```

Class Apples

```
package KI_35.Kudenchuk.Lab7;

class Apples implements Fruit
{
    private String appleColor;
    private int appleAmount;

    public Apples(String pColor, int pAmount)
    {
        appleColor = pColor;
        appleAmount = pAmount;
    }

    public String getColor()
```

```
    {
        return appleColor;
    }

    public int getAmount()
    {
        return appleAmount;
    }

    public void setAppleColor(String color)
    {
        appleColor = color;
    }

    public void SetAmount(int n)
    {
        appleAmount = n;
    }

    public int compareTo(Fruit p)
    {
        Integer s = appleAmount;
        return s.compareTo(p.getAmount());
    }
    public void print()
    {
        System.out.print("Apples has color: " + appleColor + ", Amount of
apples: " + appleAmount + ";\n");
    }
}
```

Class Pears

```
package KI_35.Kudenchuk.Lab7;


class Pears implements Fruit
{
    private String pearsColor;
    private final int pearsAmount;


    public Pears(String pColor, int pAmount)
    {
        pearsColor = pColor;
        pearsAmount = pAmount;
    }

    public String getColor()
    {
        return pearsColor;
    }

    public void setPearsColor(String color)
    {
        pearsColor = color;
    }

    public int getAmount()
    {
        return pearsAmount;
    }
```

```
    public int compareTo(Fruit p)
    {
        Integer s = pearsAmount;
        return s.compareTo(p.getAmount());
    }

    public void print()
    {
        System.out.print("Pears are: " + pearsColor + ", Amount of pears: "
+ pearsAmount + ";\n");
    }
}
```

Interface Fruit

```
package KI_35.Kudenchuk.Lab7;

public interface Fruit extends Comparable<Fruit>
{
    int getAmount();
    String getColor();
    void print();
}
```

Console:

```
Fruit added: Pears are: Green, Amount of pears: 9;
Fruit added: Apples has color: Red, Amount of apples: 10;
Fruit added: Apples has color: Green, Amount of apples: 8;
Fruit added: Pears are: Yellow, Amount of pears: 10;

The amount of red objects:
10
```

Відповіді на контрольні запитання

1. **Розкрийте синтаксис визначення простого параметризованого класу.**

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

2. **Розкрийте синтаксис створення об’єкту параметризованого класу.**

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

3. **Розкрийте синтаксис визначення параметризованого методу.**

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

Висновок: оволодів навиками параметризованого програмування мовою Java.