Міністерство освіти і науки України

Національний університет «Львівська Політехніка»

Кафедра ЕОМ

ЗВІТ

з лабораторної роботи №6

з дисципліни: “Кросплатформенні засоби програмування”

на тему: **«Параметризоване програмування»**

Варіант - **19**

Виконав:

ст. гр. КІ-305

Лихограй А. В.

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: **«Сейф»**. Клас має містити мінімум 4 методи опрацювання даних включаючи розміщення та виймання елементів. Реалізувати пошук – максимального об’єкту. 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. Дати відповіді на контрольні запитання:
    - дайте визначення терміну «параметризоване програмування».
    - розкрийте синтаксис визначення простого параметризованого класу.
    - розкрийте синтаксис створення об’єкту параметризованого класу.
    - розкрийте синтаксис визначення параметризованого методу.
    - розкрийте синтаксис виклику параметризованого методу.
    - яку роль відіграє встановлення обмежень для змінних типів?
    - як встановити обмеження для змінних типів?
    - розкрийте правила спадкування параметризованих типів.
    - яке призначення підстановочних типів?
    - застосування підстановочних типів.

Вихідний код програми:

Data.java:

```
package KI305.Lykhohrai.Lab6;
 /** Interface <code>Data</code> extends Comparable
 * @author Andrew Lykhohrai
 * @vrsion 1.0  **/
interface Data extends Comparable<Data>
{
    public int getSize();
    public void printInfoAbout();
    public void setName(String name);
    public String getName();
}
```

Money.java

```
package KI305.Lykhohrai.Lab6;
/**
 * Class <code>Money</code> implements Data
 * @author Andrew Lykhohrai
 * @version 1.0
 **/
public class Money implements Data{
    private String billName;
    private int amount;
    private static int countMoney = 0;
    /**
     * Constructor
     * @param name Name of money
     * @param size Amount of money
     */

    public Money(){};
    public Money (String name,  int size)
    {
        billName = name;
        amount = size;
        countMoney += size;
    }
    /**
     *
     * @return amount of money
     */
    public static int getCountGold() {
        return countMoney;
    }
    /**
     * Method simulates comparing variable size
     */
    public int compareTo(Data p)
    {
        Integer s = amount;
        return s.compareTo(p.getSize());
    }
    /**
     * Method of displaying the added jewelry
     */
    @Override
    public void printInfoAbout() {
        System.out.print("Купюра: \"" + billName + "\", кількість: " +
amount + "\n");
    }
    /**
     *
     * @return name of bill
     */
    public String getName() {
        return billName;
    }
    /**
     * Set name of bill
     */
    public void setName(String billName) {
        this.billName = billName;
    }
    /**
     *
     * @return amount of money
     */
    public int getSize() {
```

```
        return amount;
    }
    /**
     * Set size of jewelry
     */
    public void setSize(int size) { this.amount = size; }
}
```

Precious.java

```
package KI305.Lykhohrai.Lab6;
/**
 * Class <code>Precious</code> implements Data
 * @author Andrew Lykhohrai
 * @version 1.0
 **/
public class Precious implements Data{
    private String preciousName;
    private int nSize;
    private static int countPrecious = 0;
    /**
     * Constructor
     * @param name Name of precious
     * @param size size of precious
     */
    public Precious(){};
    public Precious (String name,  int size)
    {
        preciousName = name;
        nSize = size;
        countPrecious++;
    }
    /**
     *
     * @return count of precious
     */
    public static int getCountPrecious() {
        return countPrecious;
    }
    /**
     * Method simulates comparing variable size
     */
    public int compareTo(Data p)
    {
        Integer s = nSize;
        return s.compareTo(p.getSize());
    }
    /**
     * Method of displaying the added jewelry
     */
    @Override
    public void printInfoAbout() {
        System.out.print("Коштовність: \"" + preciousName + "\", кільсть:
" + nSize + "\n");
    }
    /**
     *
     * @return name of jewelry
     */
    public String getName() {
        return preciousName;
    }
    /**
     * Set name of jewelry
     */
```

```
    public void setName(String preciousName) {
        this.preciousName = preciousName;
    }
    /**
     *
     * @return amount of jewelry
     */
    public int getSize() {
        return nSize;
    }
    /**
     * Set size of jewelry
     */
    public void setSize(int nSize) {
        this.nSize = nSize;
    }
}
```

Safe.java

```
package KI305.Lykhohrai.Lab6;
import java.util.ArrayList;
/**
 * @author Andrew Lykhohrai
 * Class Safe
 * @version 1.0
 */
class Safe<T extends Data>
{
    private ArrayList<T> arr;
    private int capacity;
    private int maxItem = 0;
    private int countItem = 0;
    /**
     * Constructor
     */
    public Safe(int size)
    {
        arr = new ArrayList<T>(size);
        capacity = size;
    }
    /**
     * Method simulates finding the most valuable in Safe
     */
    void getMax()
    {
        maxItem = arr.get(0).getSize();
        int temp = 0;
        // Get the minimum number in the entire stack
        if (arr.isEmpty())
            System.out.println("Сейф пустий!\n");
        else
        {
            for(int i = 0; i < arr.size(); i++)
            {
                if(maxItem < arr.get(i).getSize()) {
                    maxItem = arr.get(i).getSize();;
                    temp = i;
                }
            }
        }
            System.out.print("Найбільша цінність: \"" +
arr.get(temp).getName() + "\", кількість: " + maxItem + "\n\n");
    }
    /**
```

```
     * Method simulates add jewelry
     */
    public int addItem(T Data)
    {
        if (IsFull()) {
            System.out.println("У сейфі не має місця!\n");
            return -1;
        }
        arr.add(Data);
        System.out.print("Додано: \t");
        Data.printInfoAbout();
        countItem++;
        return 0;
    }
    /**
     * Method simulates delete jewelry from Safe
     */
    public void deleteItem(String nameItem) {
        if (IsEmpty())
        {
            System.out.println("Сейф пустий!\n");
            System.exit(-1);
        }
        for(int i = 0; i < arr.size(); i++)
            if(arr.get(i).getName() == nameItem) {
                arr.get(i).printInfoAbout();
                arr.remove(arr.get(i));

                System.out.print(" - був видалений!\n\n");
            }
    }
    /**
     * Method for checking the contents of the safe
     */
    public boolean IsEmpty() {
        return countItem == 0;
    }
    /**
     * Method for checking the contents of the safe
     */
    public boolean IsFull() {
        return countItem == capacity;
    }
    /**
     * Method for printing all jewelry from Safe
     */
    public void printItem() {
        System.out.print("Вміст сейфа: \n");
        for (int i = 0 ; i < arr.size(); i++) {
            System.out.print(i + ". ");
            arr.get(i).printInfoAbout();
        }
    }
}
```

SafeApp.java

```
package KI305.Lykhohrai.Lab6;
/**
 * Class Safe implements main method for Safe
 * Class possibilities demonstration
 * @author Andrew Lykhohrai
 * @version 1.0
 */
public class SafeApp {
```

```
    /**
     * @param args
     */
    public static void main(String[] args)
    {
        Safe <? super Data> MySafe = new Safe <Data>(10);
        MySafe.addItem(new Money("Долар", 4000));
        MySafe.addItem(new Precious("Золото", 20));
        MySafe.addItem(new Money("Євро", 100));
        MySafe.addItem(new Money("Рубін", 1000));
        MySafe.addItem(new Precious("Діамант", 2));
        MySafe.addItem(new Money("Долар", 1000));
        MySafe.addItem(new Money("Гривня", 500));
        MySafe.addItem(new Money("Долар", 190));
        MySafe.addItem(new Money("Євро", 3000));
        MySafe.addItem(new Money("Гривня", 2000));
        MySafe.addItem(new Money("Євро", 15));
        MySafe.addItem(new Money("Рубін", 130));
        MySafe.getMax();
        MySafe.deleteItem("Долар");
        MySafe.deleteItem("Рубін");
        MySafe.getMax();
        MySafe.printItem();
        System.out.print("\nЗагальна кількість купюр: " +
Money.getCountGold());
    }
}
```

Результат виконання програми:

```
"C:\Program Files\Java\jdk-21\bin\java.exe" "-
Додано:     Купюра: "Долар", кількість: 4000
Додано:     Коштовність: "Золото", кільсть: 20
Додано:     Купюра: "Євро", кількість: 100
Додано:     Купюра: "Рубін", кількість: 1000
Додано:     Коштовність: "Діамант", кільсть: 2
Додано:     Купюра: "Долар", кількість: 1000
Додано:     Купюра: "Гривня", кількість: 500
Додано:     Купюра: "Долар", кількість: 190
Додано:     Купюра: "Євро", кількість: 3000
Додано:     Купюра: "Гривня", кількість: 2000
У сейфі не має місця!

У сейфі не має місця!

Найбільша цінність: "Долар", кількість: 4000

Купюра: "Долар", кількість: 4000
 - був видалений!

Купюра: "Долар", кількість: 1000
 - був видалений!

Купюра: "Долар", кількість: 190
 - був видалений!

Купюра: "Рубін", кількість: 1000
 - був видалений!

Купюра: "Долар", кількість: 1000
 - був видалений!

Купюра: "Рубін", кількість: 1000
 - був видалений!

Найбільша цінність: "Євро", кількість: 3000

Вміст сейфа:
0. Коштовність: "Золото", кільсть: 20
1. Купюра: "Євро", кількість: 100
2. Коштовність: "Діамант", кільсть: 2
3. Купюра: "Гривня", кількість: 500
4. Купюра: "Євро", кількість: 3000
5. Купюра: "Гривня", кількість: 2000

Загальна кількість купюр: 11935
Process finished with exit code 0
```

Рис. 1. Результат роботи програми.

Фрагмент згенерованої документації:

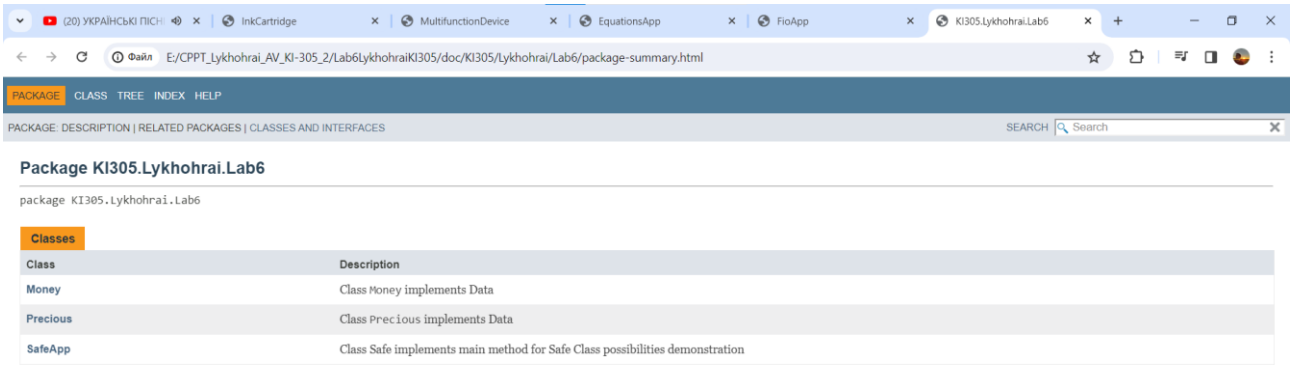

Рис. 2. Фрагмент документації.

Відповіді на контрольні запитання:

1. Визначення терміну «параметризоване програмування»:

- Параметризоване програмування (або generics) - це підхід до програмування, який дозволяє створювати класи, інтерфейси та методи, які можуть працювати з будь-яким типом даних, визначеним користувачем.

2. Синтаксис визначення простого параметризованого класу:

```java
public class ParametrizedClass<T> {
    // тіло класу
}
```

3. Синтаксис створення об’єкту параметризованого класу:

```java
ParametrizedClass<Integer> obj = new ParametrizedClass<>();
```

4. Синтаксис визначення параметризованого методу:

```java
```

```
public <T> void parametrizedMethod(T parameter) {
    // тіло методу
}
```

5. Синтаксис виклику параметризованого методу:

```java
parametrizedMethod("Hello");
```

6. Роль встановлення обмежень для змінних типів:

- Обмеження для змінних типів дозволяють вказати, що тип даних повинен відповідати певним критеріям.

7. Як встановити обмеження для змінних типів:

- Використовуйте ключове слово `extends` для встановлення обмежень верхньої межі (upper bound) або `super` для встановлення обмежень нижньої межі (lower bound).

8. Правила спадкування параметризованих типів:

- Параметризовані типи не підлягають автоматичному спадкуванню між ними. Наприклад, `Class<A>` і `Class<B>` не є підтипами один одного, навіть якщо `A` і `B` - підтипи або нащадки одного класу.

9. Призначення підстановочних типів:

- Підстановочні типи (wildcards) використовуються для створення більш гнучких параметризованих методів і класів, які можуть працювати зі змінними типами.

10. Застосування підстановочних типів:

- Використовуються, наприклад, при оголошенні параметрів методів чи колекцій для забезпечення більшої гнучкості і повторного використання коду.

Висновок: під час виконання лабораторної роботи я ознайомився з параметризованим програмуванням в мові Java. Навчився працювати з ними. Розробив програму, яка реалізовує предметну область – «Сейф».