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

Звіт
з лабораторної роботи № 6
з дисципліни: «Кросплатформенні засоби програмування»
на тему: «Параметризоване програмування»

Виконав:
студент групи КІ-306
Щирба Д.В.
Перевірив:
доцент кафедри ЕОМ
Іванов Ю. С.

Львів – 2023

Мета роботи: оволодіти навиками параметризованого програмування мовою Java.

Завдання (варіант № 28): Антресолі

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.

Вихідний код програми:

Файл Item.java

```java
package KI306.Shchyrba.Lab6;
/**
 * An interface representing an item.
 */
interface Item extends Comparable<Item> {
    /**
     * Gets the size of the item.
     * @return The size of the item.
     */
    int getSize();

    //int compareTo(Item item);

    /**
     * Prints information about the item.
     */
    void print();
}
```

Файл Antresol.java

```java
package KI306.Shchyrba.Lab6;

import java.util.ArrayList;


public class Antresol<T extends Item> {
    private ArrayList<T> items;

    /**
     * Constructs a new Antresol object.
     */
    public Antresol() {
        items = new ArrayList<>();
    }

    /**
     * Finds the item with the minimum size in the Antresol.
     * @return The item with the minimum size, or null if the Antresol is empty.
     */
    public T findMin() {
        if (!items.isEmpty()) {
            T min = items.get(0);
            for (int i = 1; i < items.size(); i++) {
                if (items.get(i).compareTo(min) < 0)
                    min = items.get(i);
            }
            return min;
        }
        return null;
    }
    /**
     * Adds an item to the suitcase.
     * @param item The item to be added.
     */
    public void addItem(T item) {
        items.add(item);
        System.out.print("Item added: ");
        item.print();
    }

    /**
     * Removes an item from the suitcase at the specified index.
     * @param i The index of the item to be removed.
     */
    public void removeItem(int i) {
        if (i >= 0 && i < items.size()) {
            items.remove(i);
            System.out.println("Item removed at index " + i);
        } else {
            System.out.println("Invalid index. Cannot remove item.");
        }
    }


    public void printContents() {
        if (!items.isEmpty()) {
            for (T item : items) {
                item.print();
            }
        } else {
            System.out.println("Antresol is empty. No items available.");
        }
```

```
    }
}


class Hat implements Item {
    private String hatType;
    private String hatBrand;
    private int hatSize;



    public Hat(String hType, String hBrand, int hSize) {
        hatType = hType;
        hatBrand = hBrand;
        hatSize = hSize;
    }

    // SET + GET [TYPE]
    public String getHatType() {
        return hatType;
    }

    public void setHatType(String type) {
        hatType = type;
    }

    // SET + GET [BRAND]
    public String getHatBrand() {
        return hatBrand;
    }

    public void setHatBrand(String brand) {
        hatBrand = brand;
    }

    // SET [SIZE]
    public void setHatSize(int size) {
        hatSize = size;
    }

    // Implementing methods from Item interface:
    public int getSize() {
        return hatSize;
    }

    public int compareTo(Item item) {
        Integer s = hatSize;
        return s.compareTo(item.getSize());
    }

    public void print() {
        System.out.println("[Hat]");
        System.out.println("  Type: " + hatType);
        System.out.println("  Brand: " + hatBrand);
        System.out.println("  Size: " + hatSize);
        System.out.println();
    }
}

/**
* A class representing Scarfs.
*/
class Scarf implements Item {
    private String scarfColor;
```

```java
    private int scarfLength;



    public Scarf(String sColor,  int sLength) {
        scarfColor = sColor;
        scarfLength = sLength;

    }

    // SET + GET [Color]
    public String getScarfColor() {
        return scarfColor;
    }
    public void setScarfColor(String color) {
        scarfColor = color;
    }


    // SET [Length]
    public void SetBookSize(int n) {
        scarfLength = n;
    }

    // Implementing methods from Item interface:
    public int getSize() {
        return scarfLength;
    }
    public int compareTo(Item item) {
        Integer s = scarfLength;
        return s.compareTo(item.getSize());
    }
    public void print() {
        System.out.println("[Scarf]");
        System.out.println("  Color: " + scarfColor);
        System.out.println("  Length: " + scarfLength);
        System.out.println();
    }
}
```

Файл AntresolDriver.java

```java
package KI306.Shchyrba.Lab6;

public class AntresolDriver {

      /**
     * The main entry point for the application.
     * @param args Command line arguments.
     */
        public static void main(String[] args) {
            Antresol<? super Item> suitcase = new Antresol<>();

            System.out.println();
            suitcase.addItem(new Hat("Fedora","Gucci", 35));
            suitcase.addItem(new Hat("Cap", "Gap", 32));

            suitcase.addItem(new Scarf("Brown", 20));
            suitcase.addItem(new Scarf("Amber", 25));


            suitcase.removeItem(3);

            System.out.print("\nContents of Antresol: \n");
            suitcase.printContents();

            Item minItem = suitcase.findMin();
            System.out.print("\nThe smallest item in the Antresol is: ");
            minItem.print();
        }
    }
```

Результат виконання програми:

```
Item added: [Hat]
  Type: Fedora
  Brand: Gucci
  Size: 35

Item added: [Hat]
  Type: Cap
  Brand: Gap
  Size: 32

Item added: [Scarf]
  Color: Brown
  Length: 20

Item added: [Scarf]
  Color: Amber
  Length: 25

Item removed at index 3

Contents of Antresol:
[Hat]
  Type: Fedora
  Brand: Gucci
  Size: 35

[Hat]
  Type: Cap
  Brand: Gap
  Size: 32

[Scarf]
  Color: Brown
  Length: 20


The smallest item in the Antresol is: [Scarf]
  Color: Brown
  Length: 20
```

Фрагмент згенерованої документації:

Package KI306.Shchyrba.Lab6

package KI306.Shchyrba.Lab6

Classes

Class	Description
Antresol<T extends KI306.Shchyrba.Lab6.Item>	
AntresolDriver	

Відповіді на контрольні запитання

1. Дайте визначення терміну «параметризоване програмування».
- це підхід до програмування, що дозволяє створювати класи і методи, які можна використовувати з різними типами даних, надаючи більшу гнучкість і безпеку типів у програмах.

2. Розкрийте синтаксис визначення простого параметризованого класу.
- public class НазваКласу<параметризованийТип> {
  // Тіло класу
  }

3. Розкрийте синтаксис створення об’єкту параметризованого класу.
- НазваКласу<перелікТипів> змінна Класу = new НазваКласу<перелікТипів>(параметри);

4. Розкрийте синтаксис визначення параметризованого методу.
- public <параметризованийТип> типПовернення назваМетоду(параметри) {
  // Тіло методу
  }

5. Розкрийте синтаксис виклику параметризованого методу.
- (НазваКласу|НазваОб’єкту).<перелікТипів>назваМетоду(параметри);

6. Яку роль відіграє встановлення обмежень для змінних типів?
- дозволяє заборонити використання деяких типів або вимагати, щоб тип підставлений за замовчуванням був підкласом або реалізував певний інтерфейс.

7. Як встановити обмеження для змінних типів?
- за допомогою ключового слова extends для суперкласу або інтерфейсу, від яких має походити реальний тип.

8. Розкрийте правила спадкування параметризованих типів.
- Всі класи, створені з параметризованого класу, незалежні один від одного.
- Зазвичай немає залежності між класами, створеними з різними параметрами типів.

9. Яке призначення підстановочних типів?
- використовуються для забезпечення безпеки типів при використанні параметризованих класів та методів. Вони дозволяють визначити, які типи можна використовувати замість параметризованих типів.

10. Застосування підстановочних типів.
- <?> (unbounded wildcard) дозволяє читати об'єкти з колекції без змінення її.
- <? extends Тип> (bounded wildcard) дозволяє читати об'єкти з колекції, але забороняє додавання в неї нових об'єктів.
- <? super Тип> (lower bounded wildcard) дозволяє додавати об'єкти в колекцію, але забороняє їх читання.

Висновок

У ході виконання даної лабораторної роботи, я отримав важливі навички параметризованого програмування мовою Java. Ознайомився з різними аспектами мови, такими як використання параметрів у методах, створення та використання класів та інтерфейсів.