Міністерство освіти і науки України

Національний університет «Львівська політехніка»

Звіт

З лабораторної роботи №7

З дисципліни «Кросплатформенні засоби програмування»
На тему: «Параметризоване програмування»

Виконав:
ст.гр. КІ-36
Литовко С.Г
Прийняв:
Іванов Ю.С.

Львів-2022

Мета роботи: оволодіти навиками параметризованого програмування мовою Java.

ТЕОРЕТИЧНІ ВІДОМОСТІ

Параметризоване програмування є аналогом шаблонів у C++. Воно полягає у написанні коду, що можна багаторазово застосовувати з об'єктами різних класів. Користувачів параметризованого програмування можна поділити на 3 рівні кваліфікації:

1. ті, що користуються готовими класами;
2. ті, що користуються готовими класами і вміють виправляти помилки, що виникають при їх використанні;
3. ті, що пишуть власні параметризовані класи.

Для успішного застосування параметризованого програмування слід навчитися розуміти помилки, що генерує середовище при компіляції програми, що можуть стосуватися, наприклад, неоднозначності визначення спільного суперкласу для всіх переданих об'єктів. З іншої сторони необхідно передбачити захист від присвоєння об'єктів параметризованого класу, що містять об'єкти підкласу об'єктам параметризованого класу, що містять об'єкти суперкласу і дозволити зворотні присвоєння. Для вирішення цієї проблеми у мові Java введено так звані *підстановочні типи*. Це далеко не всі «підводні камені», що виникають при застосуванні параметризованого програмування.

Визначення простого параметризованого класу

Параметризований клас – це клас з однією або більше змінними типу.

Синтаксис оголошення параметризованого класу:

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

Іменувати змінні параметризованих типів прийнято великими літерами. У бібліотеці Java використовується літера E для позначення типу колекції, K і V – для типів ключа і значення таблиці, T, U, S та сусідні літери – для позначення довільних типів.

Приклад оголошення параметризованого класу:

```
class GenericClass<T, U>
{
public GenericClass(T first, U second)
{
this.first = first;
this.second = second;
}
public void setFirst(T first)
```

```
{
this.first = first;
}
public T getFirst()
{
return first;
}
...
private T first;
private U second;
}
```

Тут T і U – це змінні параметризованих типів, що використовуються по всьому тілу класу для специфікації типу повернення методів, типів полів і локальних змінних. При створенні об'єкту параметризованого класу замість них підставляються реальні типи, що визначаються в трикутних дужках у місці створення об'єкту параметризованого класу.

Синтаксис створення об'єкту параметризованого класу:
НазваКласу < перелікТипів > = new НазваКласу < перелікТипів > (параметри);

Приклад створення об'єкту параметризованого класу:
GenericClass<String, Integer> obj = new GenericClass<String, Integer> ();

Параметризовані методи

Параметризовані методи визначаються в середині як звичайних класів так і параметризованих. На відміну від звичайних методів параметризовані методи мають параметризований тип, що дозволяє за їх допомогою опрацьовувати різнотипні набори даних. Реальні типи для методів, як і для класів, визначаються у місці виклику методу шляхом передачі реального типу у трикутних дужках.

Синтаксис оголошення параметризованого методу:

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

Приклад оголошення параметризованого методу:

```
class ArrayAlg
{
public static<T> T getMiddle(T[] a)
{
return a[a.length / 2];
}
}
```

Синтаксис виклику параметризованого методу:
(НазваКласу|НазваОб'єкту).[<перелікТипів>] НазваМетоду(параметри);

У мові Java компілятор здатний самостійно визначати типи, що підставляються замість параметризованих типів, тому у трикутних дужках вказувати реальні типи не обов'язково. Проте це може призвести до помилок, якщо компілятор не зможе однозначно визначити єдиний супертип для всіх параметрів.

Приклад виклику параметризованого методу:

```
String[] names = {"Ivan", "Ivanovych", "Ivanov"};

String middle = ArrayAlg.<String>getMiddle(names);
```

або

```
String middle = ArrayAlg.getMiddle(names);
```

Приклад виклику параметризованого методу, що може призвести до помилок, оскільки параметри методу можна привести як до класу Double так і до інтерфейсу Comparable:

```
ArrayAlg.<String>getMiddle(3.75, 123, 0);
```

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.

Варіант:

6. Шафа

Код програми:

App.java

```java
package KI36.Lytovko.Lab7;

public class App {
    /**
     * @param args
     */
    public static void main(String[] args)
    {
        Wardrobe wardrobe = new Wardrobe();

        wardrobe.add(new Tshirt(200,"Tshirt1"));
        wardrobe.add(new Cap(100,"Cap1"));
        wardrobe.add(new Tshirt(100,"Tshirt2"));
        wardrobe.add(new Cap(250,"Cap2"));
        wardrobe.add(new Tshirt(200,"Tshirt3"));
        wardrobe.add(new Cap(390,"Cap3"));

        System.out.println(wardrobe.get(0).getName());
        System.out.println(wardrobe.get(1).getName());
        System.out.println(wardrobe.get(2).getName());
        System.out.println(wardrobe.get(3).getName());

        System.out.println(wardrobe.getItemWithMinWeight().getName());
        System.out.println(wardrobe.getTotalWeight());

    }
}
```

Cap.java

```java
package KI36.Lytovko.Lab7;

public class Cap extends Item {
    /**
     * Constructor
     * @param weight Cap weight
     * @param name Cap name
     */
    public Cap(int weight, String name)
    {
        super(weight, name);
    }

}
```

Wardrobe.java

App.java | Cap.java | Item.java | Tshirt.java | Wardrobe.java

```java
package KI36.Lytovko.Lab7;

import java.util.ArrayList;
/**
 * Class Wardrobe
 * @version 1.0
 */
public class Wardrobe<T extends Item> {
    private ArrayList<T> storage = new ArrayList();
    /**
     * Method returns an item by the index
     * @param index Element index
     * @throws Exception
     */
    public T get(int index)
    {
        try{
            return storage.get(index);
        }
        catch (Exception e)
        {
            throw e;
        }
    }
    /**
     * Method adds an item
     * @param item Item to be added
     */
    public void add(T item)
    {
        try {
            storage.add(item);
        }
        catch (Exception e)
        {
            throw e;
        }
    }
    /**
     * Method shows storage content
     */
    public void showAll()
    {
        for(int i = 0; i < storage.size(); i++)
        {
            try{
                storage.get(i).printData();
            }
            catch (Exception e)
            {
                System.out.println(e);
            }
        }
    }
    /**
     * Method returns total weight
     */
    public int getTotalWeight()
    {
        int result = 0;
        for(int i = 0; i < storage.size(); i++)
        {
            result += storage.get(i).getWeight();
        }
        return result;
    }
```

```java
    /**
     * Method returns an item with maximal weight
     */
    public T getItemWithMinWeight()
    {
        int minIdx = 0;

        for(int i = 0; i < storage.size(); i++)
        {
            if(i == 0)
            {
                continue;
            }
            if(storage.get(i).getWeight() > storage.get(minIdx).getWeight())
            {
                minIdx = i;
            }
        }
        return storage.get(minIdx);
    }
}
```

Item.java

App.java | Cap.java | Item.java | Tshirt.java | Wardrobe.java

```java
package KI36.Lytovko.Lab7;

public abstract class Item {
    protected int weight;
    protected String name;
    /**
     * Constructor
     * @param weight Item weight
     * @param name Item name
     */
    public Item(int weight, String name)
    {
        this.name = name;
        this.weight = weight;
    }
    /**
     * Methods returns item weight
     */
    public int getWeight() {
        return weight;
    }
    /**
     * Methods returns item name
     */
    public String getName() { return name; }
    /**
     * Methods prints item data
     */
    public void printData()
    {
        System.out.println("Name:" + name);
        System.out.println("Weight:" + weight);
    }
}
```

Tshirt.java

App.java | Cap.java | Item.java | Tshirt.java | Wardrobe.java

```java
package KI36.Lytovko.Lab7;

public class Tshirt extends Item {
    /**
     * Constructor
     * @param weight Tshirt weight
     * @param name Tshirt name
     */
    public Tshirt(int weight, String name)
    {
        super(weight, name);
    }

}
```

Результат програми:

Problems | Javadoc | Declaration | Console

<terminated> App (6) [Java Application] C:\Program Fil

```
Tshirt1
Cap1
Tshirt2
Cap2
Cap3
1240
```

Висновок: Зробивши цю лабораторну роботу, я оволоділа навиками параметризованого програмування мовою Java.