Міністерство освіти і науки України

Національний університет “Львівська політехніка”

Кафедра ЕОМ

Звіт

з лабораторної роботи №7

з дисципліни: «Кросплатформенні засоби програмування»

на тему: «Параметризоване програмування»

Виконав: ст.гр. КІ-34

Степанов А. О.

Прийняв:

викл. каф. ЕОМ

Іванов Ю. С.

Львів 2022

Мета роботи: 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.Lab7 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 фрагмент згенерованої документації.
4. Дати відповідь на контрольні запитання.

Варіант 21

21. Бак для сміття

Лістинг програми:

Файл TrashCanApp.java

```
/**
 * lab 7 package
 */
package KI34.Stepanov.Lab7;

/**
 * Trash can Application class implements main method for Trash can
 * class possibilities demonstration
 * @author Andriy Stepanov
 * @version 1.0
 */
public class TrashCanApp
{
    /**
     * @param args param
     */
    public static void main(String[] args) {
        TrashCan <? super Data> trashCan = new TrashCan <Data>();
        trashCan.AddData(new Thing("Computer", 3, 25));
        trashCan.AddData(new Food("Apples" , 1));
        trashCan.AddData(new Food("Oranges" , 6));
        trashCan.AddData(new Thing("Phone" , 4, 4));
        Data res = trashCan.findMax();
        System.out.print("The greatest data on trash can is: \n");
        res.print();
    }
}
```

Файл TrashCan.java

```java
public class TrashCan <T extends Data>
{
    private ArrayList<T> arr;
    /**
     * Constructor
     */
    public TrashCan() {
        arr = new ArrayList<T>();
    }

    /**
     * Method simulates finding the largest shop
     */
    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;
    }

    /**
     * Method simulates adding data
     */
    public void AddData(T data)
    {
        arr.add(data);
        System.out.print("Element added: ");
        data.print();
    }

    /**
     * Method simulates deleting data
     */
    public void DeleteData(int i)
    {
        arr.remove(i);
    }
}
```

Файл Thing.java

```java
/**
 * lab 7 package
 */
package KI34.Stepanov.Lab7;

/**
 * Class <code>Thing</code> implements Data
 * @author Andriy Stepanov
 * @version 1.0
 **/
public class Thing implements Data
{
    private String thingName;
    private int usedYears;
    private int weight;

    /**
     * Constructor
     * @param tName Name of thing
     * @param tUsed Thing`s years usage
     * @param tWeight Thing`s weight
     */
    public Thing(String tName, int tUsed, int tWeight)
    {
        thingName = tName;
        usedYears = tUsed;
        weight = tWeight;
    }

    /**
     * Method returns thing`s name
     * @return thing`s name
     */
    public String getThingName()
    {
        return thingName;
    }
    /**
     * Method sets the new thing`s name
     * @param name thing`s name
     */
    public void setThingName(String name)
    {
        thingName = name;
    }
    /**
     * Method returns thing`s years usage
     * @return entertainment shop`s size
     */
    public int getUsedYears()
    {
        return usedYears;
    }
    /**
     * Method sets the new thing`s years usage
     * @param n thing`s years usage
     */
    public void setUsedYears(int n)
    {
        usedYears = n;
    }
    /**
     * Method returns thing`s weight
     * @return thing`s weight
     */
```

```
    public int getWeight()
    {
        return weight;
    }
    /**
     * Method simulates comparing thing`s weight
     */
    public int compareTo(Data p)
    {
        Integer s = weight;
        return s.compareTo(p.getWeight());
    }

    /**
     * Method simulates printing info about thing
     */
    public void print()
    {
        System.out.print("Thing: " + thingName + ", Years used: " + usedYears +
                ", Thing Weight: " + weight + ";\n");
    }
}
```

Файл Food.java

```
/**
 * lab 7 package
 */
package KI34.Stepanov.Lab7;

/**
 * Class <code>Food</code> implements Data
 * @author Andriy Stepanov
 * @version 1.0
 **/
class Food implements Data
{
    private String foodName;
    private int weight;

    /**
     * Constructor
     * @param fName Name of Food
     * @param fWeight Food`s weight
     */
    public Food(String fName, int fWeight)
    {
        foodName = fName;
        weight = fWeight;
    }
    /**
     * Method returns food`s name
     * @return Food`s name
     */
    public String getName()
    {
        return foodName;
    }
    /**
     * Method sets the new food`s name
     * @param name Food`s name
     */
    public void SetName(String name)
    {
        foodName = name;
    }
```

```
    /**
     * Method sets the new food`s weight
     * @param n food`s weight
     */
    public void SetWeight(int n)
    {
        weight = n;
    }
    /**
     * Method returns food`s weight
     * @return food`s weight
     */
    public int getWeight()
    {
        return weight;
    }
    /**
     * Method simulates comparing food`s weight
     */
    public int compareTo(Data p)
    {
        Integer s = weight;
        return s.compareTo(p.getWeight());
    }
    /**
     * Method simulates printing info about food
     */
    public void print()
    {
        System.out.print("Eat: " + foodName + ", Eat Weight: " + weight +
";\n");
    }
}
```

Файл Data.java

```
/**
 * lab 7 package
 */
package KI34.Stepanov.Lab7;

/**
 * Interface <code>Data</code> extends Comparable
 * @author Andriy Stepanov
 * @version 1.0
 **/
interface Data extends Comparable<Data>
{
    public int getWeight();
    public void print();
}
```

Результат виконання програми:

```
Element added: Thing: Computer, Years used: 3, Thing Weight: 25;
Element added: Eat: Apples, Eat Weight: 1;
Element added: Eat: Oranges, Eat Weight: 6;
Element added: Thing: Phone, Years used: 4, Thing Weight: 4;
The greatest data on trash can is:
Thing: Computer, Years used: 3, Thing Weight: 25;

Process finished with exit code 0
```

Успішне виконання програми

Package KI34.Stepanov.Lab7

package KI34.Stepanov.Lab7

All Classes and Interfaces | Interfaces | Classes

Class	Description
Data	Interface Data extends Comparable
Food	Class Food implements Data
Thing	Class Thing implements Data
TrashCan<T extends Data>	Class TrashCan implements Trash can
TrashCanApp	Trash can Application class implements main method for Trash can class possibilities demonstration

Згенерована документація

Package KI34.Stepanov.Lab7

Interface Data

All Superinterfaces:

Comparable<Data>

All Known Implementing Classes:

Food, Thing

```
public interface Data
extends Comparable<Data>
```

Interface Data extends Comparable

Version:

1.0

Author:

Andriy Stepanov

Method Summary

All Methods | Instance Methods | Abstract Methods

Modifier and Type	Method	Description
int	getWeight()	
void	print()	

Methods inherited from interface java.lang.Comparable

compareTo

Інформація про клас Data

Package KI34.Stepanov.Lab7

Class Food

java.lang.Object
KI34.Stepanov.Lab7.Food

All Implemented Interfaces:

Comparable<Data>, Data

```
public class Food
extends Object
implements Data
```

Class Food implements Data

Version:

1.0

Author:

Andriy Stepanov

Constructor Summary

Constructors

Constructor	Description
Food(String fName, int fWeight)	Constructor

Method Summary

All Methods | Instance Methods | Concrete Methods

Modifier and Type	Method	Description
int	compareTo(Data p)	Method simulates comparing food`s weight
String	getName()	Method returns food`s name
int	getWeight()	Method returns food`s weight
void	print()	Method simulates printing info about food
void	SetName(String name)	Method sets the new food`s name
void	SetWeight(int n)	Method sets the new food`s weight

Інформація про клас Food

Package KI34.Stepanov.Lab7

Class Thing

java.lang.Object
    KI34.Stepanov.Lab7.Thing

All Implemented Interfaces:

Comparable<Data>, Data

```
public class Thing
extends Object
implements Data
```

Class Thing implements Data

Version:

1.0

Author:

Andriy Stepanov

Constructor Summary

Constructors

Constructor	Description
Thing(String tName, int tUsed, int tWeight)	Constructor

Method Summary

All Methods | Instance Methods | Concrete Methods

Modifier and Type	Method	Description
int	compareTo(Data p)	Method simulates comparing thing`s weight
String	getThingName()	Method returns thing`s name
int	getUsedYears()	Method returns thing`s years usage
int	getWeight()	Method returns thing`s weight
void	print()	Method simulates printing info about thing
void	setThingName(String name)	Method sets the new thing`s name
void	setUsedYears(int n)	Method sets the new thing`s years usage

Інформація про клас Thing

Package KI34.Stepanov.Lab7

Class TrashCan<T extends Data>

java.lang.Object
KI34.Stepanov.Lab7.TrashCan<T>

public class TrashCan<T extends Data>
extends Object

Class TrashCan implements Trash can

Version:

1.0

Author:

Andriy Stepanov

Constructor Summary

Constructors

Constructor	Description
TrashCan()	Constructor

Method Summary

All Methods | Instance Methods | Concrete Methods

Modifier and Type	Method	Description
void	AddData(T data)	Method simulates adding data
void	DeleteData(int i)	Method simulates deleting data
T	findMax()	Method simulates finding the largest shop

Methods inherited from class java.lang.Object

equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Інформація про клас TrashCan

Package KI34.Stepanov.Lab7

Class TrashCanApp

java.lang.Object
KI34.Stepanov.Lab7.TrashCanApp

public class TrashCanApp
extends Object

Trash can Application class implements main method for Trash can class possibilities demonstration

Version:

1.0

Author:

Andriy Stepanov

Constructor Summary

Constructors

Constructor	Description
TrashCanApp()	

Method Summary

All Methods | Static Methods | Concrete Methods

Modifier and Type	Method	Description
static void	main(String[] args)	

Methods inherited from class java.lang.Object

equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Інформація про клас TrashCanApp

Відповіді на контрольні запитання:

1. **Дайте визначення терміну «параметризоване програмування».**

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

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

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

Синтаксис оголошення параметризованого класу:

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

{…}

3. **Розкрийте синтаксис створення об'єкту параметризованого класу.**

Синтаксис створення об'єкту параметризованого класу:

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

4. **Розкрийте синтаксис визначення параметризованого методу.**

Синтаксис оголошення параметризованого методу:

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

5. **Розкрийте синтаксис виклику параметризованого методу.**

Синтаксис виклику параметризованого методу:

(НазваКласу|НазваОб'єкту).[<перелікТипів>] НазваМетоду(параметри);

6. **Яку роль відіграє встановлення обмежень для змінних типів?**

Бувають ситуації, коли клас або метод потребують накладення обмежень на змінні типів. Наприклад, може бути ситуація, коли метод у процесі роботи викликає з-під об'єкта параметризованого типу метод, що визначається у деякому інтерфейсі. У такому випадку немає ніякої гарантії, що цей метод буде реалізований у кожному класі, що передається через змінну типу. Щоб вирішити цю проблему у мові Java можна задати обмеження на множину можливих типів, що можуть бути підставлені замість параметризованого типу.

7. **Як встановити обмеження для змінних типів?**

У мові Java можна задати обмеження на множину можливих типів, що можуть бути підставлені замість параметризованого типу. Для цього після змінної типу слід використати ключове слово extends і вказати один суперклас, або довільну кількість інтерфейсів (через знак &), від яких має походити реальний тип, що підставляється замість параметризованого типу. Якщо одночасно вказуються інтерфейси і суперклас, то суперклас має стояти першим у списку типів після ключового слова extends.Класи DataOutputStream і DataInputStream дозволяють записувати і зчитувати дані примітивних типів.

8. **Розкрийте правила спадкування параметризованих типів.**

Правила спадкування параметризованих типів:

1. Всі класи, що утворені з одного і того ж параметризованого класу з використанням різних значень змінних типів є незалежними навіть якщо між цими типами є залежність спадкування.
2. Завжди можна перетворити параметризований клас у «сирий» клас, при роботі з яким захист від некоректного коду є значно слабшим, що дозволяє здійснювати небезпечні присвоєння об’єктів параметризованого класу об’єктам «сирого» класу.
3. Параметризовані класи можуть розширювати або реалізовувати інші параметризовані класи. В цьому відношенні вони не відрізняються від звичайних класів.

9. **Яке призначення підстановочних типів?**

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. Це в свою чергу дозволяє застосовувати обмеження для параметрів, що підставляються замість параметризованих типів. Завдяки цьому підвищується надійність параметризованого коду, полегшується робота з ним та розділяється використання безпечних методів доступу і небезпечних модифікуючих методів.

10. **Застосування підстановочних типів.**

Підстановочні типи дозволяють реалізувати:

1. обмеження підтипу;
2. обмеження супертипу;
3. необмежені підстановки.

Висновок:

На цій лабораторній роботі я оволодів навиками параметризованого програмування мовою Java.