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

Лис Б. Л.

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 11
Торговий центр

Лістинг програми:

Файл ShopCenterApp.java

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

/**
 * Shopping centre Application class implements main method for Shopping centre
 * class possibilities demonstration
 * @author Lys Bohdan
 * @version 1.0
 */
public class ShopCenterApp {
    /**
     * @param args param
     */
    public static void main(String[] args) {
        ShopCenter <? super Info> obj = new ShopCenter <Info>();

        obj.AddShop(new Shop("Waikiki", 1, 80));
        obj.AddShop(new Shop("Asus" , 1,75));

        obj.AddShop(new Entertainment("Kozachok" , 70));
        obj.AddShop(new Entertainment("Papashon" ,  100));
        obj.AddShop(new Entertainment("Karusel" ,  125));

        /*Кількість магазинів та розваг*/
        System.out.println(Shop.shop);
        System.out.println(Entertainment.entertainmet);

        Info result = obj.findMax();
        System.out.print("\nThe greatest shop in shopping centre is: ");
```

```
        result.print();
    }
}
```

Файл ShopCenter.java

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

import java.util.*;

/**
 * Class <code>ShopCenter</code> implements Shopping Center
 * * @author Lys Bohdan
 *  * @version 1.0
 **/
public class ShopCenter <T extends Info> {
    private ArrayList<T> arr;

    /**
     * Constructor
     */
    public ShopCenter()
    {
        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 shop
     */
    public void AddShop(T info)
    {
        arr.add(info);
        System.out.print("Shop added: ");
        info.print();
    }

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

```
}
```

Файл Shop.java

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

/**
 * Class <code>Shop</code> implements Info
 * * @author Lys Bohdan
 *  * @version 1.0
 **/
public class Shop implements Info{
    private String shopName;
    private int floor;
    private int size;
    static int shop;

    /**
     * Constructor
     * @param shopName <code>shopName</code> Name of shop
     * @param floor <code>floor</code> Number of floor
     * @param size <code>size</code> Size of shop
     */
    public Shop(String shopName, int floor, int size)
    {
        this.shopName = shopName;
        this.floor = floor;
        this.size = size;
        shop++;
    }

    /**
     * Method returns shop`s name
     * @return shop`s name
     */
    public String getName() {
        return shopName;
    }

    /**
     * Method sets the new shop`s name
     * @param shopName <code>shopName</code> shop`s name
     */
    public void setShopName(String shopName) {
        this.shopName = shopName;
    }

    /**
     * Method returns floor`s number
     * @return floor`s number
     */
    public int getFloor() {
        return floor;
    }

    /**
     * Method sets the new floor`s number
     * @param floor <code>floor</code> floor`s number
     */
    public void setFloor(int floor) {
        this.floor = floor;
    }

    /**
```

```
     * Method returns shop`s size
     * @return shop`s size
     */
    @Override
    public int getSize() {
        return size;
    }

    /**
     * Method sets the new shop`s size
     * @param size <code>size</code> shop`s size
     */
    public void setSize(int size) {
        this.size = size;
    }

    /**
     * Method simulates comparing shop`s size
     */
    public int compareTo(Info point)
    {
        Integer s = size;
        return s.compareTo(point.getSize());
    }

    /**
     * Method simulates printing info about shop
     */
    public void print()
    {
        System.out.print("\nShop: " + shopName + ", Floor: " + floor + ", Shop
Size: " + size + "m2;\n");
    }
}
```

Файл Entertainment.java

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

/**
 * Class <code>Entertainment</code> implements Info
 * * @author Lys Bohdan
 *  * @version 1.0
 **/
public class Entertainment implements Info{
    private String entertainmentShopName;
    private int entertainmentSize;;
    public static int entertainmet;
    /**
     * Constructor
     * @param entertainmentShopName <code>entertainmentShopName</code> Name of
entertainment shop
     * @param entertainmentSize <code>entertainmentSize</code> entertainment
shop`s size
     */
    public Entertainment(String entertainmentShopName, int entertainmentSize)
    {
        this.entertainmentShopName = entertainmentShopName;
        this.entertainmentSize = entertainmentSize;
        entertainmet++;
    }

    /**
```

```
     * Method returns entertainment shop`s name
     * @return entertainment shop`s name
     */
    public String getName() {
        return entertainmentShopName;
    }

    /**
     * Method sets the new entertainment shop`s name
     * @param entertainmentShopName <code>entertainmentShopName</code>
entertainment shop`s name
     */
    public void setEntertainmentShopName(String entertainmentShopName) {
        this.entertainmentShopName = entertainmentShopName;
    }

    /**
     * Method returns entertainment shop`s size
     * @return entertainment shop`s size
     */
    public int getSize() {
        return entertainmentSize;
    }

    /**
     * Method sets the new entertainment shop`s size
     * @param entertainmentSize <code>entertainmentSize</code> entertainment
shop`s size
     */
    public void setEntertainmentSize(int entertainmentSize) {
        this.entertainmentSize = entertainmentSize;
    }

    /**
     * Method simulates comparing entertainment shop`s size
     */
    public int compareTo(Info point)
    {
        Integer s = entertainmentSize;
        return s.compareTo(point.getSize());
    }

    /**
     * Method simulates printing info about entertainment shop
     */
    public void print()
    {
        System.out.print("\nEntertainment Shop Name: " + entertainmentShopName +
", Entertainment Shop Size: " + entertainmentSize + "m2;\n");
    }
}
```

Файл Info.java

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

/**
 * Interface <code>Info</code> extends Comparable
 * * @author Lys Bohdan
 *  * @version 1.0
 **/
public interface Info extends Comparable<Info> {
    public String getName();
```

```
    public int getSize();
    public void print();
}
```

Результат виконання програми:

```
C:\Users\mrpet\.jdks\openjdk-18.0.2\bin\java.exe "-javaagent:C:\Program Files\JetB
Shop added:
Shop: Waikiki, Floor: 1, Shop Size: 80m2;
Shop added:
Shop: Asus, Floor: 1, Shop Size: 75m2;
Shop added:
Entertainment Shop Name: Kozachok, Entertainment Shop Size: 70m2;
Shop added:
Entertainment Shop Name: Papashon, Entertainment Shop Size: 100m2;
Shop added:
Entertainment Shop Name: Karusel, Entertainment Shop Size: 125m2;
2
3

The greatest shop in shopping centre is:
Entertainment Shop Name: Karusel, Entertainment Shop Size: 125m2;

Process finished with exit code 0
```

Успішне виконання програми

Package KI34.Lys.Lab7

package KI34.Lys.Lab7

All Classes and Interfaces | Interfaces | Classes

Class	Description
Entertainment	Class Entertainment implements Info * @author Lys Bohdan * @version 1.0
Info	Interface Info extends Comparable * @author Lys Bohdan * @version 1.0
Shop	Class Shop implements Info * @author Lys Bohdan * @version 1.0
ShopCenter<T extends Info>	Class ShopCenter implements Shopping Center * @author Lys Bohdan * @version 1.0
ShopCenterApp	Shopping centre Application class implements main method for Shopping centre class possibilities demonstration

Згенерована документація

Package KI34.Lys.Lab7

Interface Info

All Superinterfaces:

`Comparable<Info>`

All Known Implementing Classes:

`Entertainment`, `Shop`

```
public interface Info
extends Comparable<Info>
```

Interface Info extends Comparable * @author Lys Bohdan * @version 1.0

Method Summary

All Methods | **Instance Methods** | **Abstract Methods**

Modifier and Type	Method	Description
String	getName()	
int	getSize()	
void	print()	

Methods inherited from interface java.lang.Comparable

compareTo

Інформація про клас Info

Package KI34.Lys.Lab7

Class Entertainment

java.lang.Object
    KI34.Lys.Lab7.Entertainment

All Implemented Interfaces:

`Comparable<Info>, Info`

```
public class Entertainment
extends Object
implements Info
```

Class Entertainment implements Info * @author Lys Bohdan * @version 1.0

Constructor Summary

Constructors

Constructor	Description
`Entertainment(String entertainmentShopName, int entertainmentSize)`	Constructor

Method Summary

All Methods | **Instance Methods** | **Concrete Methods**

Modifier and Type	Method	Description
int	`compareTo(Info point)`	Method simulates comparing entertainment shop`s size
String	`getName()`	Method returns entertainment shop`s name
int	`getSize()`	Method returns entertainment shop`s size
void	`print()`	Method simulates printing info about entertainment shop
void	`setEntertainmentShopName(String entertainmentShopName)`	Method sets the new entertainment shop`s name
void	`setEntertainmentSize(int entertainmentSize)`	Method sets the new entertainment shop`s size

Інформація про клас Entertainment

Package KI34.Lys.Lab7

Class Shop

java.lang.Object
    KI34.Lys.Lab7.Shop

All Implemented Interfaces:

`Comparable<Info>, Info`

```
public class Shop
extends Object
implements Info
```

Class Shop implements Info * @author Lys Bohdan * @version 1.0

Constructor Summary

Constructors

Constructor	Description
`Shop(String shopName, int floor, int size)`	Constructor

Інформація про клас Shop

Package KI34.Lys.Lab7

Class ShopCenter<T extends Info>

java.lang.Object
    KI34.Lys.Lab7.ShopCenter<T>

```
public class ShopCenter<T extends Info>
extends Object
```

Class ShopCenter implements Shopping Center * @author Lys Bohdan * @version 1.0

Constructor Summary

Constructors

Constructor	Description
ShopCenter()	Constructor

Method Summary

All Methods | **Instance Methods** | **Concrete Methods**

Modifier and Type	Method	Description
void	AddShop(T info)	Method simulates adding shop
void	DeleteData(int i)	Method simulates deleting shop
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 ShopCenter

Package KI34.Lys.Lab7

Class ShopCenterApp

java.lang.Object
KI34.Lys.Lab7.ShopCenterApp

```
public class ShopCenterApp
extends Object
```

Shopping centre Application class implements main method for Shopping centre class possibilities demonstration

Version:

1.0

Author:

Lys Bohdan

Constructor Summary

Constructors

Constructor	Description
ShopCenterApp()	

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 ShopCenterApp

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.