

Міністерство освіти і науки України

Національний університет «Львівська політехніка»

Кафедра ЕОМ



Звіт

до лабораторної роботи № 7

з дисципліни: «Кросплатформні засоби програмування»

«Параметризоване програмування»

Варіант - 7

Виконав:

студент групи КІ-36

Мазуренко Н.А.

Прийняв:

Іванов Ю.С.

Львів 2022

Мета: 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

8

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

7. Трюм корабля

Код програми:

File ShipsHold.java

```
package example.test.lab7;
import
java.util.ArrayList;
public class ShipsHold<T extends Data >{

    private boolean isOpened=false;
private ArrayList<T> arr;      public
ShipsHold()
    {
        arr = new ArrayList<T>();
    }
}
```



```

    /**
    * Method returns max size value of element from array
    *
    * @return max element size value
    */
    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 openning hold
    */
    public void openHold() {
isOpened=true;
        System.out.println("Ship hold opened successfully!");
    }

    /**
    * Method simulates closing hold
    */
    public void closeHold(){
isOpened=false;
        System.out.println("Ship hold closed successfully!");
    }

    /**
    * Method adds new element to array
    */
    public void addData(T data)
    {
        if(isOpened==true) {
arr.add(data);
            System.out.print("Element added: ");
data.print();
        }else{
            System.out.println("Please open hold to add new item");
        }
    }

    /**
    * Method deletes element from array with i index
    */
    public void deleteData(int i)
    {

```

```
        if(isOpened==true) {  
arr.remove(i);  
        }else{  
            System.out.println("Please open hold to delete item");  
        }  
    }  
}
```

```
}  
} }
```

File ShipsHoldApp.java

```
package example.test.lab7;  
  
import java.io.FileNotFoundException;  
  
public class ShipsHoldApp{  
    /**  
     * @param args  
     */  
    public static void main(String[] args) {  
        ShipsHold<? super Data> shipsHold = new ShipsHold<Data>();  
        shipsHold.openHold();  
        shipsHold.addData(new Cargo(10000,"Fish"));  
        shipsHold.addData(new Cargo(2500,"Car"));  
        shipsHold.addData(new Equipment(100,"Bucket of sand","to stop the  
fire"));  
        shipsHold.addData(new Equipment(200,"Boat","to rescue from drowning"));  
        Data res = shipsHold.findMax();  
        System.out.print("The greatest data on shipsHold is: \n");  
        res.print();  
    }  
}
```

File Equipment.java

```
package example.test.lab7;  
  
public class Equipment implements Data {  
    private int weight;    private String  
name;    private String appointment;  
  
    /**  
     * Constructor  
     *  
     * @param  
     * <code>eWeight</code> equipment weight value  
     * <code>eName</code> equipment name value  
     * <code>eAppointment</code> equipment appointment value  
     */  
    public Equipment(int eWeight, String eName, String eAppointment) {  
        weight = eWeight;
```

```

        name = eName;
        appointment = eAppointment;
    }

    /**
     * Method prints info about cargo
     */
    @Override
    public void print() {
        System.out.println("Equipment: " + name + "\nWeight: " + weight +
"\nAppointment: " + appointment + "\n");
    }

    /**
     * Method returns equipment weight value
     */
    * @return Equipment weight value
    */
    @Override
    public int getWeight() {
return weight;
    }
    public void setWeight(int weight) {
this.weight = weight;
    }

    /**
     * Method returns equipment name value
     */
    * @return Equipment name value
    */
    public String getName() {
return name;
    }
    public void setName(String name) {
this.name = name;
    }
    public String getAppointment() {
return appointment;
    }
    public void setAppointment(String
appointment) {
        this.appointment =
appointment;
    }

    @Override
    public int compareTo(Data e) {
Integer w = weight;
        return w.compareTo(e.getWeight());
    }
}

```

File Cargo.java

```
package example.test.lab7;
```



```

public class Cargo implements Data{
    // cargo weight value
    private int weight;    //
    cargo name value;
    private String name;

    /**
     * Constructor
     *
     * @param
     * <code>cWeight</code> cargo weight value      * @param
     * <code>cName</code> cargo name value
     */
    public Cargo(int cWeight, String cName) {
        weight = cWeight;        name = cName;
    }

    /**
     * Method prints info about cargo
     */
    @Override
    public void print() {
        System.out.println("Cargo: " + name + "\nWeight: " + weight + "\n");
    }

    /**
     * Method returns cargo weight value
     *
     * @return Cargo weight value
     */
    public int getWeight() {
        return weight;
    }
    public void setWeight(int weight) {
        this.weight = weight;
    }

    /**
     * Method returns cargo name value      *
     * @return Cargo name value
     */
    public String getName() {
        return name;
    }
    public void setName(String
name) {        this.name = name;
    }

    @Override

```

```
public int compareTo(Data c) {
    Integer w = weight;
    return w.compareTo(c.getWeight());
} }
```

File Date.java

```
package example.test.lab7; //
оголошуємо інтерфейс Data
public interface Data extends Comparable<Data>{
    void print(); // прототип методу      int
    getWeight(); // прототип методу }
```

Результат виконання програми:

```
"C:\Program Files\Java\jdk-18.0.2.1\bin\java.exe" "-javaagent:C:\Program Files\
Ship hold opened successfully!
Element added: Cargo: Fish
Weight: 10000

Element added: Cargo: Car
Weight: 2500

Element added: Equipment: Bucket of sand
Weight: 100
Appointment: to stop the fire

Element added: Equipment: Boat
Weight: 200
Appointment: to rescue from drowning

The greatest data on shipsHold is:
Cargo: Fish
Weight: 10000

Process finished with exit code 0
```

Висновок: оволодів навиками параметризованого програмування мовою Java.