

Міністерство освіти і науки України  
Національний університет «Львівська політехніка»



### **Звіт**

до лабораторної роботи №6

З дисципліни: «Кросплатформенні засоби програмування»

На тему: « ПАРАМЕТРИЗОВАНЕ ПРОГРАМУВАННЯ »

Виконав:

Ст.гр. КІ-301

Осадчук В.В.

Прийняв:

Майдан М.В.

Львів 2023

**Мета:** оволодіти навиками параметризованого програмування мовою Java

Виконання роботи:

### 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 розміщуються 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. Завантажити код на GitHub згідно методичних вказівок по роботі з GitHub.
4. Скласти звіт про виконану роботу з приведенням тексту програми, результату її виконання та фрагменту згенерованої документації та завантажити його у ВНС.
5. Дати відповідь на контрольні запитання.

## 18. Відсік для зброї

*Код програми:*

Клас GunSafeAllInfo:

```
package KI301_Osadchuk_Lab6;

public class GunSafeAllInfo {

    public static void main(String[] args) {

        GunSafe<? super GunSafeInfo> safe = new GunSafe<GunSafeInfo>();
        safe.WriteToPC(new AmmunitionSafe());
        safe.WriteToPC(new HuntingWeaponSafe());
        safe.WriteToPC(new PistolSafe());
        safe.WriteToPC(new AmmunitionSafe(7, 30, 40, 2450));
        safe.WriteToPC(new PistolSafe(15, 45, 38, 3950));
        safe.WriteToPC(new HuntingWeaponSafe(31,140,190,6200));
        GunSafeInfo smallestsafe = safe.findMinPrice();
        System.out.print("\n Сейф з найменшою ціною: \n\n");
        smallestsafe.print_info();
    }

}
```

## Клас PistolSafe:

```
package KI301_Osadchuk_Lab6;
```

```
public class PistolSafe implements GunSafeInfo{  
    private String mark;  
    private double Weight;  
    private double Size;  
    private int Capacity;  
    private int Price;  
    private String material;
```

```
    public PistolSafe(){  
        this.mark = "Техно-Идеал Плюс";  
        this.Weight = 8.0;  
        this.Size = 50.25;  
        this.Capacity = 45;  
        this.Price = 2800;  
        this.material = "Метал";  
    }
```

```
    public PistolSafe(String mark, double Weight, double Size,int Capacity, int Price,  
        String material){  
        this.mark = mark;  
        this.Weight = Weight;  
        this.Size = Size;  
        this.Capacity = Capacity;  
        this.Price = Price;  
        this.material = material;  
    }
```

```
    public PistolSafe(double Weight, double Size,int Capacity, int Price){  
        this.mark = "New Gun Company";  
        this.material = "Metal";  
        this.Weight = Weight;  
        this.Price = Price;  
        this.Size = Size;  
        this.Capacity = Capacity;  
    }
```

```
    @Override  
    public double getSize() {  
        return this.Size;  
    }
```

```
    @Override  
    public int getCapacity() {
```

```
return this.Capacity;  
}
```

```
@Override  
public double getWeight() {  
return this.Weight;  
}
```

```
@Override  
public int getPrice() {  
return this.Price;  
}
```

```
@Override  
public String getMaterial()  
{  
return this.material;  
}
```

```
@Override  
public void print_info() {  
System.out.println("\tСейф для пістолетів");  
System.out.println("Марка --> " + this.mark);  
System.out.println("Матеріал --> " + this.material);  
System.out.println("Розмір --> " + this.Size + " см^2.");  
System.out.println("Вага --> " + this.Weight + " кг.");  
System.out.println("Вмістимість --> " + this.Capacity + " см^3.");  
System.out.println("Ціна --> " + this.Price + " грн.");  
}  
@Override  
public int compareTo(GunSafeInfo o) {  
Integer to_compare = Price;  
return to_compare.compareTo(o.getPrice());  
}  
}
```

### Клас GunSafe:

```
package KI301_Osadchuk_Lab6;
```

```
import java.util.ArrayList;
```

```
public class GunSafe <T extends GunSafeInfo>{
```

```

private ArrayList<T> arr;

public GunSafe() {
    arr = new ArrayList<T>();
}

//find minimal capacity
public T findMinPrice() {
    if (!arr.isEmpty()) {
        T min = arr.get(0);
        for (int i = 1; i < arr.size(); i++) {
            if (arr.get(i).compareTo(min) < 0)
                min = arr.get(i);
        }
        return min;
    }
    return null;
}

public void WriteToPC(T dani) {
    arr.add(dani);
    System.out.println("\n");
    dani.print_info();
}

public void DeleteData(int x)
{
    arr.remove(x);
}

}

```

## Клас HuntingWeaponSafe:

```
package KI301_Osadchuk_Lab6;
```

```
public class HuntingWeaponSafe implements GunSafeInfo{  
    private String mark;  
    private double Weight;  
    private double Size;  
    private int Capacity;  
    private int Price;  
    private String material;
```

```
  
    public HuntingWeaponSafe(){  
        this.mark = "FEROCON";  
        this.Weight = 16.0;  
        this.Size = 80.25;  
        this.Capacity = 120;  
        this.Price = 4530;  
        this.material = "Метал";  
    }
```

```
  
    public HuntingWeaponSafe(String mark, double Weight, double Size,int Capacity, int  
    Price, String material){  
        this.mark = mark;  
        this.Weight = Weight;  
        this.Size = Size;  
        this.Capacity = Capacity;  
        this.Price = Price;  
        this.material = material;  
    }
```

```
    public HuntingWeaponSafe(double Weight, double Size,int Capacity, int Price){
```

```
this.mark = "New Gun Company";
this.material = "Метал";
this.Weight = Weight;
this.Price = Price;
this.Size = Size;
this.Capacity = Capacity;
}
@Override
public double getSize()
{
return this.Size;
}
```

```
@Override
public int getCapacity()
{
return this.Capacity;
}
```

```
@Override
public double getWeight()
{
return this.Weight;
}
```

```
@Override
public int getPrice()
{
return this.Price;
}
```

```
@Override
```

```

public String getMaterial()
{
return this.material;
}

```

```

@Override
public void print_info() {
System.out.println("\tСейф для мисливської зброї");
System.out.println("Марка --> " + this.mark);
System.out.println("Матеріал --> " + this.material);
System.out.println("Розмір --> " + this.Size + " см^2.");
System.out.println("Вага --> " + this.Weight + " кг.");
System.out.println("Вмістимість --> " + this.Capacity + "см^3.");
System.out.println("Ціна --> " + this.Price + " грн.");
}

```

```

@Override
public int compareTo(GunSafeInfo o) {
Integer to_compare = Price;
return to_compare.compareTo(o.getPrice());
}
}

```

### Клас HuntingWeaponSafe:

```

package KI301_Osadchuk_Lab6;

```

```

public class AmmunitionSafe implements GunSafeInfo{
private String mark;
private double Weight;
private double Size;
private int Capacity;
private int Price;

```



```
private String material;
```

```
public AmmunitionSafe(){  
    this.mark = "Griffon";  
    this.Weight = 5.0;  
    this.Size = 22.25;  
    this.Capacity = 45;  
    this.Price = 2500;  
    this.material = "Metal";  
}
```

```
public AmmunitionSafe(String mark, double Weight, double Size,int Capacity, int  
Price, String material){  
    this.mark = mark;  
    this.Weight = Weight;  
    this.Size = Size;  
    this.Capacity = Capacity;  
    this.Price = Price;  
    this.material = material;  
}
```

```
public AmmunitionSafe(double Weight, double Size,int Capacity, int Price){  
    this.mark = "New Gun Company";  
    this.material = "Metal";  
    this.Weight = Weight;  
    this.Price = Price;  
    this.Size = Size;  
    this.Capacity = Capacity;  
}
```

```
@Override
```

```
public double getSize() {
```

```
return this.Size;  
}
```

```
@Override  
public int getCapacity() {  
    return this.Capacity;  
}
```

```
@Override  
public double getWeight() {  
    return this.Weight;  
}
```

```
@Override  
public int getPrice() {  
    return this.Price;  
}
```

```
@Override  
public String getMaterial() {  
    return this.material;  
}
```

```
@Override  
public void print_info() {  
    System.out.println("\tПатронний сейф");  
    System.out.println("Марка --> " + this.mark);  
    System.out.println("Матеріал --> " + this.material);  
    System.out.println("Розмір --> " + this.Size + " см^2.");  
    System.out.println("Вага --> " + this.Weight + " кг.");  
    System.out.println("Вмістимість --> " + this.Capacity + " см^3.");  
    System.out.println("Ціна --> " + this.Price + " грн.");  
}
```

```
}
```

```
@Override
```

```
public int compareTo(GunSafeInfo o) {
```

```
Integer to_compare = Price;
```

```
return to_compare.compareTo(o.getPrice());
```

```
}
```

```
}
```

### Интерфейс GunSafeInfo:

```
package KI301_Osadchuk_Lab6;
```

```
public interface GunSafeInfo extends Comparable<GunSafeInfo>{
```

```
double getSize();
```

```
int getCapacity();
```

```
double getWeight();
```

```
int getPrice();
```

```
String getMaterial();
```

```
void print_info();
```

```
}
```

Патронний сейф  
Марка --> Griffon  
Матеріал --> Метал  
Розмір --> 22.25 см<sup>2</sup>.  
Вага --> 5.0 кг.  
Вмістимість --> 45см<sup>3</sup>.  
Ціна --> 2500 грн.

Сейф для мисливської зброї  
Марка --> FEROCON  
Матеріал --> Метал  
Розмір --> 80.25 см<sup>2</sup>.  
Вага --> 16.0 кг.  
Вмістимість --> 120см<sup>3</sup>.  
Ціна --> 4530 грн.

Сейф для пістолетів  
Марка --> Техно-Ідеал Плюс  
Матеріал --> Метал  
Розмір --> 50.25 см<sup>2</sup>.  
Вага --> 8.0 кг.  
Вмістимість --> 45см<sup>3</sup>.  
Ціна --> 2800 грн.

Патронний сейф  
Марка --> New Gun Company  
Матеріал --> Метал  
Розмір --> 30.0 см<sup>2</sup>.  
Вага --> 7.0 кг.  
Вмістимість --> 40см<sup>3</sup>.  
Ціна --> 2450 грн.

Сейф для пістолетів  
Марка --> New Gun Company  
Матеріал --> Metal  
Розмір --> 45.0 см<sup>2</sup>.  
Вага --> 15.0 кг.  
Вмістимість --> 38см<sup>3</sup>.  
Ціна --> 3950 грн.

Сейф для мисливської зброї  
Марка --> New Gun Company  
Матеріал --> Метал  
Розмір --> 140.0 см<sup>2</sup>.  
Вага --> 31.0 кг.  
Вмістимість --> 190см<sup>3</sup>.  
Ціна --> 6200 грн.

Сейф з найменшою ціною:

Патронний сейф  
Марка --> New Gun Company  
Матеріал --> Метал  
Розмір --> 30.0 см<sup>2</sup>.  
Вага --> 7.0 кг.  
Вмістимість --> 40см<sup>3</sup>.  
Ціна --> 2450 грн.

### Відповіді на КЗ

1. Параметризоване програмування є аналогом шаблонів у C++. Воно полягає у написанні коду, що можна багаторазово застосовувати з об'єктами різних класів.
2. Параметризований клас – це клас з однією або більше змінними типу.  
Синтаксис оголошення параметризованого класу: [public] class НазваКласу { ... }
3. GenericClass obj = new GenericClass ();
4. (НазваКласу|НазваОб'єкту).[ ] НазваМетоду(параметри);
5. МодифікаторитипПоведіння назваМетоду(параметри);
6. 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. Синтаксис оголошення параметризованого методу з обмеженнями типів:  
Модифікатори тип Повернення назва Методу(параметри);

8.

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. Проте у цьому випадку можна зробити помилки, які генеруватимуть виключення на етапі виконання програми.

3. Параметризовані класи можуть розширювати або реалізовувати інші параметризовані класи. В цьому відношенні вони не відрізняються від звичайних класів. Наприклад, ArrayList реалізує інтерфейс List. Це значить, що ArrayList можна перетворити у List. Але ArrayList це не ArrayList і не List, де SubClass – підклас суперкласу SupClass.

9. – 10. 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. Підстановочні типи застосовуються у вигляді параметру типу, що передається у трикутних дужках при утворенні реального типу з параметризованого типу, наприклад, у методі main.

**Висновок:** на даній лабораторній роботі я оволодів навиками параметризованого програмування мовою Java