Міністерство освіти і науки України

Національний університет «Львівська політехніка»

Кафедра «Електронних обчислювальних машин»



**Звіт**

з лабораторної роботи № 6

**з дисципліни:** «Кросплатформенні засоби програмування»

**на тему:** «параметризоване програмування»

**Виконав:**

студент групи *КІ-306*

*Рудь Володимир*

**Прийняв:**

доцент кафедри ЕОМ

Іванов Ю. С.

**Львів – 2023**

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

**Завдання ( Варіант 18 «Відсік для зброї»):**

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.

**Вихідний код програми**

public interface BattleModule extends Comparable<BattleModule>{  
 double priceOfSecondsOfFire();  
 void Print();  
  
 @Override  
 int compareTo(BattleModule o);  
}

import java.util.ArrayList;  
import java.util.List;  
  
public class IFVMarder <T extends BattleModule>{  
 private List<T> battleModules =new ArrayList<>();  
  
 public List<T> getBattleModules() {  
 return battleModules;  
 }  
  
 public void setBattleModules(List<T> battleModules) {  
 this.battleModules = battleModules;  
 }  
 public void AddBattleModule(T module){  
 battleModules.add(module);  
 }  
 public void DeleteBattleModule(int i)  
 {  
 battleModules.remove(i);  
 }  
 public T findCheaper()  
 {  
 if (!battleModules.isEmpty())  
 {  
 T min = battleModules.get(0);  
 for (int i=1; i< battleModules.size(); i++)  
 {  
 if ( battleModules.get(i).compareTo(min) < 0 )  
 min = battleModules.get(i);  
 }  
 return min;  
 }  
 return null;  
 }  
}

public class Main {  
 public static void main(String[] args) {  
 IFVMarder<? super BattleModule> marder1 = new IFVMarder<BattleModule>();  
 Mk20 mk20 = new Mk20();  
 Pkm akm = new Pkm();  
  
 marder1.AddBattleModule(mk20);  
 marder1.AddBattleModule(mk20);  
 marder1.AddBattleModule(akm);  
 marder1.AddBattleModule(akm);  
 marder1.AddBattleModule(akm);  
  
 marder1.findCheaper().Print();  
 }  
}

import java.util.stream.BaseStream;  
  
public class Mk20 implements BattleModule{  
 private double Price20mmCatrige = 27;  
  
 public double getPrice20mmCatrige() {  
 return Price20mmCatrige;  
 }  
  
 public void setPrice20mmCatrige(double price20mmCatrige) {  
 Price20mmCatrige = price20mmCatrige;  
 }  
 public Mk20(){}  
 public Mk20(double price20mmCatrige)  
 {  
 setPrice20mmCatrige(price20mmCatrige);  
 }  
  
 @Override  
 public void Print() {  
 System.*out*.println("This is MK20 and price of 20 mm catrige is " + Price20mmCatrige);  
 }  
  
 @Override  
 public double priceOfSecondsOfFire() {  
 return Price20mmCatrige\*16.67;  
 }  
  
 @Override  
 public int compareTo(BattleModule o) {  
 if(o.priceOfSecondsOfFire() > this.priceOfSecondsOfFire())  
 {  
 return -1;  
 }  
 else if(o.priceOfSecondsOfFire() < this.priceOfSecondsOfFire())  
 {  
 return 1;  
 }  
 return 0;  
 }  
}

public class Pkm implements BattleModule{  
 private double PriceOfBullet7\_62 = 0.36;  
  
 public double getPriceOfBullet7\_62() {  
 return PriceOfBullet7\_62;  
 }  
  
 public void setPriceOfBullet7\_62(double priceOfBullet7\_62) {  
 PriceOfBullet7\_62 = priceOfBullet7\_62;  
 }  
 public Pkm()  
 {  
  
 }  
 public Pkm(double priceOfBullet7\_62)  
 {  
 setPriceOfBullet7\_62(priceOfBullet7\_62);  
 }  
  
 @Override  
 public void Print() {  
 System.*out*.println("This is AKM and price of 7.62 mm catrige is " + PriceOfBullet7\_62);  
 }  
 @Override  
 public double priceOfSecondsOfFire() {  
 return PriceOfBullet7\_62\*10;  
  
 }  
  
 @Override  
 public int compareTo(BattleModule o) {  
 if(o.priceOfSecondsOfFire() > this.priceOfSecondsOfFire())  
 {  
 return -1;  
 }  
 else if(o.priceOfSecondsOfFire() < this.priceOfSecondsOfFire())  
 {  
 return 1;  
 }  
 return 0;  
 }  
}

**Результат виконання програми**

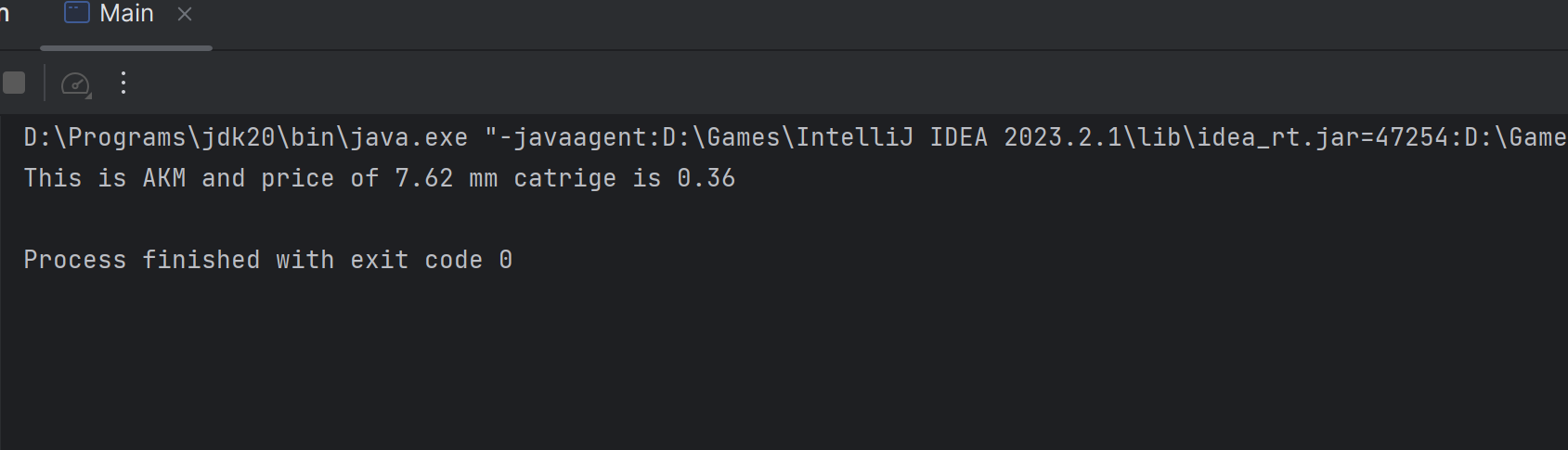
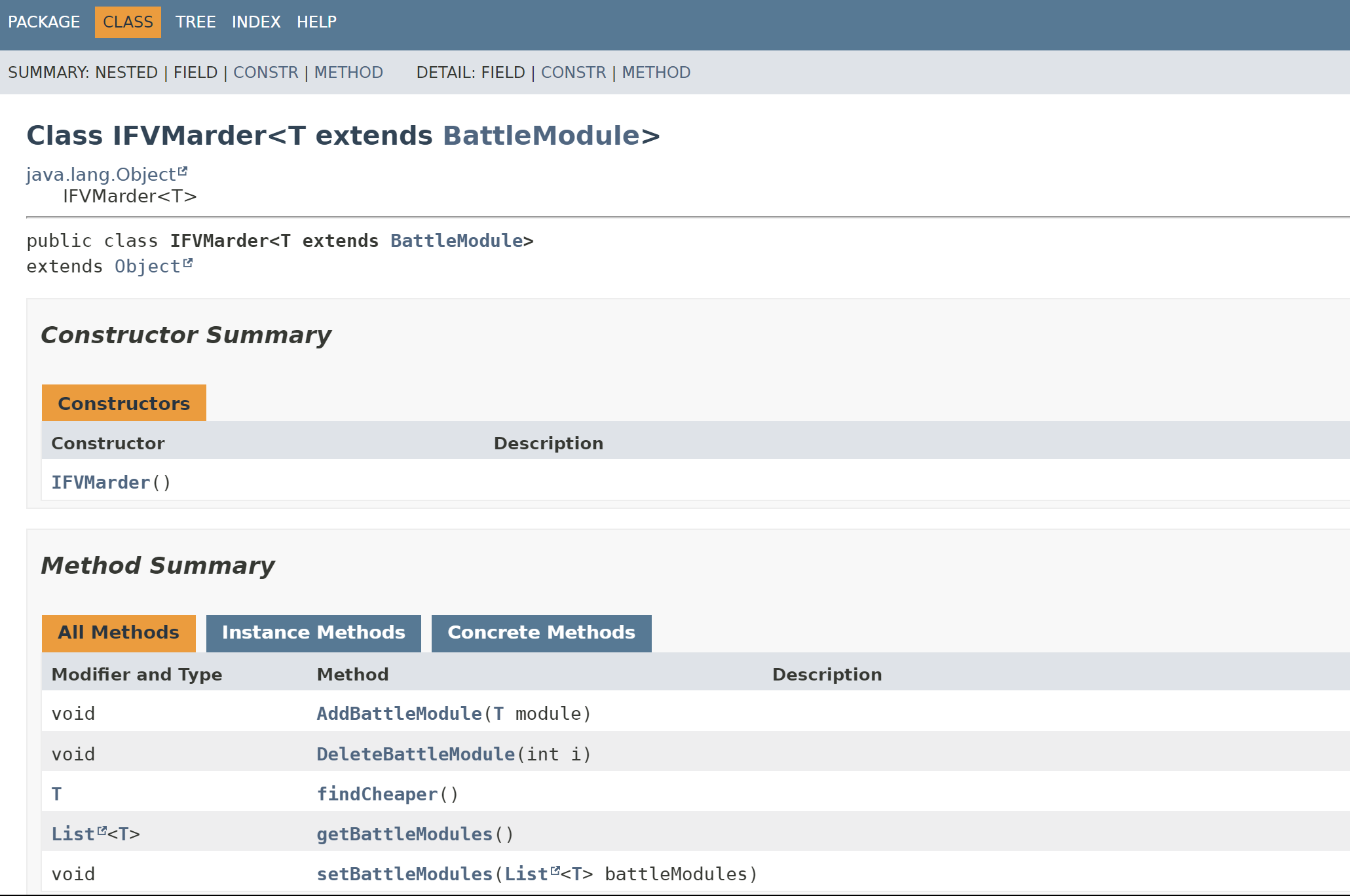
****

Рис.1 – Скріншот виконання програми

**Фрагмент згенерованої документації**



**Відповіді на контрольні запитання**

1. Дайте визначення терміну «параметризоване програмування».

Параметризоване програмування - це підхід до програмування, де класи та методи можуть приймати типи даних як параметри, що дозволяє створювати більш загальні та змінні структури даних та методи.

1. Розкрийте синтаксис визначення простого параметризованого класу.

class MyGenericClass<T> {

// Код класу з використанням параметра T

}

1. Розкрийте синтаксис створення об’єкту параметризованого класу.

MyGenericClass<Integer> obj = new MyGenericClass<>();

1. Розкрийте синтаксис визначення параметризованого методу.

<T> void myGenericMethod(T value) {

// Код методу з використанням параметра T

}

1. Розкрийте синтаксис виклику параметризованого методу.

MyClass.<Integer>myGenericMethod(42); // Виклик методу з параметром Integer

1. Яку роль відіграє встановлення обмежень для змінних типів?

Встановлення обмежень для змінних типів дозволяє обмежити допустимий тип даних, який можна використовувати як параметр типу.

1. Як встановити обмеження для змінних типів?

<T extends Number> void myMethod(T value) { /\* ... \*/ }

1. Розкрийте правила спадкування параметризованих типів.

Правила спадкування параметризованих типів включають здатність успадковувати параметризовані класи та методи з інтерфейсів або суперкласів та можливість використовувати абстрактні типи даних.

1. Яке призначення підстановочних типів?

Підстановочні типи (wildcards) використовуються для створення більш загальних методів та класів, які можуть приймати об'єкти різних типів.

1. Застосування підстановочних типів.

Застосування підстановочних типів включає використання ? для прийняття об'єктів різних типів, таких як <?> для довільного типу або <? extends T> для обмеження типу до підкласу T або <? super T> для обмеження типу до суперкласу T.

**Висновок**

Ознайомився з використанням параметризованого програмування. Створив клас який реалізує предметну область «відсік для зброї». Та розробив клас драйвер який показує роботу параметризованого класу контейнера.