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»

**Виконала:**

студентка групи *КІ-306*

*Максимчук К.С*

**Прийняв:**

доцент кафедри ЕОМ

Іванов Ю. С.

**Львів – 2023**

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

**Завдання ( Варіант 9 «Споруда»):**

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**

import java.util.ArrayList;  
import java.util.Arrays;  
import java.util.List;  
  
public class Building <T extends Floor>{  
 private List<T> floors = new ArrayList<>();  
  
  
 public Building(){  
 };  
 public void AddFloor(T floor)  
 {  
 floors.add(floor);  
 System.*out*.print("floor added: ");  
 floor.Print();  
 }  
 public void DeleteFloor(int i)  
 {  
 floors.remove(i);  
 }  
  
 public T findMax()  
 {  
 if (!floors.isEmpty())  
 {  
 T max = floors.get(0);  
 for (int i=1; i< floors.size(); i++)  
 {  
 if ( floors.get(i).compareTo(max) > 0 )  
 max = floors.get(i);  
 }  
 return max;  
 }  
 return null;  
 }  
 public T findMin()  
 {  
 if (!floors.isEmpty())  
 {  
 T min = floors.get(0);  
 for (int i=1; i< floors.size(); i++)  
 {  
 if ( floors.get(i).compareTo(min) < 0 )  
 min = floors.get(i);  
 }  
 return min;  
 }  
 return null;  
 }  
}

public interface Floor extends Comparable<Floor>{  
 double getPrice();  
 void Print();  
}

public class Office implements Floor {  
 private int computer;  
  
  
 public Office(int Computer)  
 {  
 setComputer(Computer);  
 }  
  
 public int getComputer() {  
 return computer;  
 }  
  
 public void setComputer(int computer) {  
 this.computer = computer;  
 }  
  
 @Override  
 public double getPrice() {  
 return computer\*10000;  
 }  
  
 @Override  
 public void Print() {  
 System.*out*.println("Price of office is: " + getPrice());  
 }  
  
 @Override  
 public int compareTo(Floor o) {  
 if(o.getPrice() > this.getPrice())  
 {  
 return -1;  
 }  
 else if(o.getPrice() < this.getPrice())  
 {  
 return 1;  
 }  
 return 0;  
 }  
}

public class SportGym implements Floor {  
  
 private int equipment;  
  
  
  
 public SportGym(int Equipments)  
 {  
 setEquipment(Equipments);  
 }  
  
 public int getEquipment() {  
 return equipment;  
 }  
  
 public void setEquipment(int equipment) {  
 this.equipment = equipment;  
 }  
  
 @Override  
 public double getPrice() {  
 return equipment\*1000;  
 }  
  
 @Override  
 public void Print() {  
 System.*out*.println("Price of spoooooortgym is: " + getPrice());  
  
  
 }  
  
 @Override  
 public int compareTo(Floor o) {  
 if(o.getPrice() > this.getPrice())  
 {  
 return -1;  
 }  
 else if(o.getPrice() < this.getPrice())  
 {  
 return 1;  
 }  
 return 0;  
 }  
}

public class Main {  
 public static void main(String[] args) {  
 Office OfficeKatya = new Office(4);  
 SportGym SportGymVoloda = new SportGym(3);  
 SportGym SportGymDog = new SportGym(50);  
 Office OfficeKatya3 = new Office(4);  
  
 Building <? super Floor> OurHome = new Building<Floor>();  
  
 OurHome.AddFloor(SportGymVoloda);  
 OurHome.AddFloor(OfficeKatya);  
 OurHome.AddFloor(OfficeKatya);  
 OurHome.AddFloor(OfficeKatya);  
 OurHome.AddFloor(OfficeKatya);  
 OurHome.AddFloor(SportGymDog);  
  
  
  
  
 System.*out*.println("The most expensive floor price: " + OurHome.findMax().getPrice());  
 System.*out*.println("The most poor floor price: " + OurHome.findMin().getPrice());  
  
  
  
 }  
}

**Результат виконання програми**

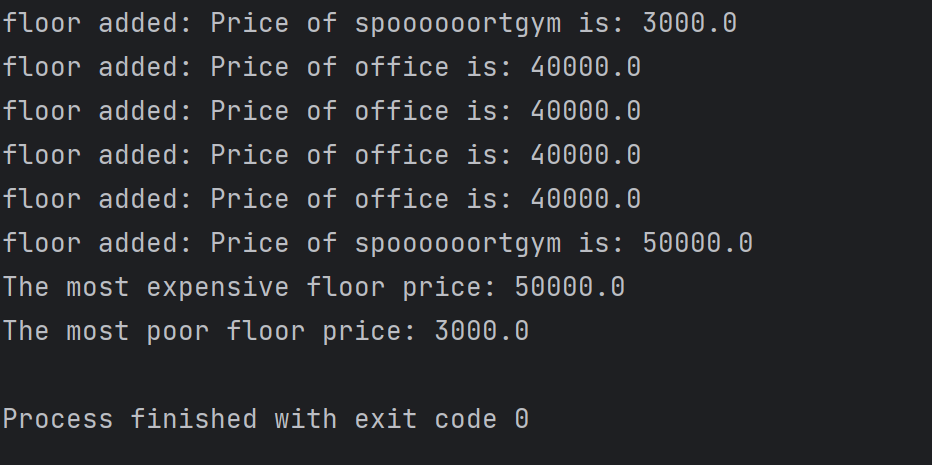
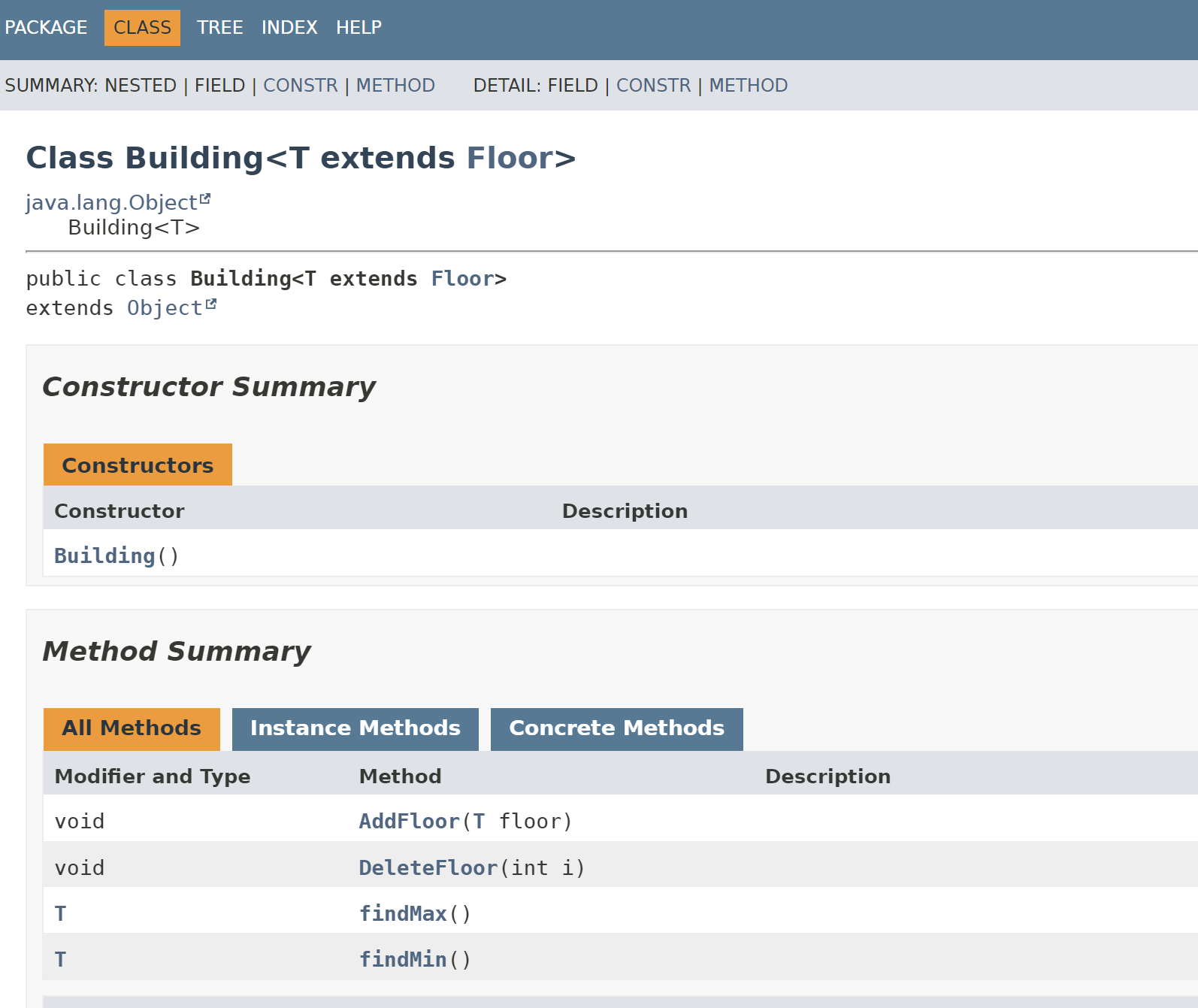
****

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**

Ознайомилася з використанням параметризованого програмування. Створила клас який реалізує предметну область «споруда». Та розробила клас драйвер який показує роботу параметризованого класу контейнера.