Національний технічний університет України
“Київський політехнічний інститут імені Ігоря Сікорського”
Факультет інформатики та обчислювальної техніки
Кафедра інформаційні систем та технологій

ЛАБОРАТОРНА РОБОТА №8
з дисципліни «Основи програмування - 2»

Тема: «Вкладені та внутрішні класи»

Виконали:
студенти групи ІА-31
Клим’юк В.Л, Самелюк А.С,
Дук М.Д, Сакун Д.С

Перевірив:
асистент кафедри ІСТ
Степанов А. С.

Київ-2024

Тема: Вкладені та внутрішні класи

Мета: Мета цієї лабораторної роботи полягає у засвоєнні та практичному застосуванні концепцій вкладених та внутрішніх класів, а також у навичках створення компараторів та їхнього використання для сортування об'єктів.

Хід роботи

1. Пригадати як використовувати вкладені (nested) та внутрішні (inner) класи.

2. Для класів свого варіанту з л/р №8 першого семестру створити компаратори для того щоб була можливість сортувати об’єкти цих класів за допомогою Arrays.sort(T[] a, Comparator<? super T> c) та зберігати у колекції TreeSet<T>. Потрібно реалізовувати принаймні два компаратори: один як статичний вкладений клас, а інший як анонімний клас. Продемонструвати використання обох компараторів (відсортувати масив об’єктів та/або зберегти об’єкти в колекції TreeSet).

```
import java.util.Arrays;
import java.util.Comparator;
import java.util.TreeSet;

public class Main {
    public static void main(String[] args) {
        Comparator<Pixel> pixelCompAnon = new Comparator<Pixel>() {
            @Override
            public int compare(Pixel o1, Pixel o2) {
                int v1 = o1.x + o1.y;
                int v2 = o2.x + o2.y;

                if (v1 == v2)
                    return 0;
                if (v1 < v2)
                    return -1;
                return 1;
            }
        };
        Comparator<ColorPixel> colorPixelCompAnon = new Comparator<ColorPixel>() {
            @Override
            public int compare(ColorPixel o1, ColorPixel o2) {
                int v1 = o1.x + o1.y;
                int v2 = o2.x + o2.y;

                if (v1 < v2)
                    return -1;
                if (v1 > v2)
                    return 1;

                if (o1.r == o2.r && o1.g == o2.g && o1.b == o2.b)
                    return 0;
```

```
                return -1;
            }
        };

        testSort(pixelCompAnon, colorPixelCompAnon);
        testTreeSet();
    }

    public static void testSort(Comparator<Pixel> comp1, Comparator<ColorPixel> comp2){
        Pixel[] pixels = new Pixel[] {
                new Pixel(100,0), new Pixel(0,100), new Pixel(50,50), new Pixel(25,6),
                new Pixel(30,30), new Pixel(40,20), new Pixel(66,33), new Pixel(1,2),
                new Pixel(40,20), new Pixel(20,40), new Pixel(0,0)
        };
        ColorPixel[] colorPixels = new ColorPixel[] {
                new ColorPixel(10,0,1,1,1), new ColorPixel(6,6,10,10,10), new
ColorPixel(0,10,0,0,0),
                new ColorPixel(12,0,3,3,3), new ColorPixel(0,24, 0,0,0), new
ColorPixel(1,1,1,1,1)
        };

        System.out.println("Unsorted pixels array:");
        for(Pixel p : pixels){
            System.out.println(p);
        }

        Arrays.sort(pixels, comp1);

        System.out.println("Sorted pixels array:");
        for(Pixel p : pixels){
            System.out.println(p);
        }

        System.out.println("Unsorted colorPixels array:");
        for(Pixel p : colorPixels){
            System.out.println(p);
        }

        Arrays.sort(colorPixels, comp2);

        System.out.println("Sorted colorPixels array:");
        for(Pixel p : colorPixels){
            System.out.println(p);
        }
    }

    public static void testTreeSet(){
        Pixel[] pixels = new Pixel[] {
                new Pixel(100,0), new Pixel(0,100), new Pixel(50,50), new Pixel(25,6),
                new Pixel(30,30), new Pixel(40,20), new Pixel(66,33), new Pixel(1,2),
                new Pixel(40,20), new Pixel(20,40), new Pixel(0,0)
        };
        ColorPixel[] colorPixels = new ColorPixel[] {
                new ColorPixel(10,0,1,1,1), new ColorPixel(6,6,10,10,10), new
ColorPixel(0,10,0,0,0),
                new ColorPixel(12,0,3,3,3), new ColorPixel(0,24, 0,0,0), new
ColorPixel(1,1,1,1,1)
        };

        TreeSet<Pixel> pixelsSet = new TreeSet<>(new Pixel.PixelComparatorStatic());
        for(Pixel p : pixels){
            pixelsSet.add(p);
        }
        TreeSet<ColorPixel> colorPixelsSet = new TreeSet<>(new
```

```
ColorPixel.ColorPixelComparatorStatic());
        for(ColorPixel p : colorPixels){
            colorPixelsSet.add(p);
        }

        System.out.println("Pixels set:");
        for(Pixel p : pixelsSet){
            System.out.println(p);
        }

        System.out.println("ColorPixels set:");
        for(ColorPixel p : colorPixelsSet){
            System.out.println(p);
        }
    }
}
```

Програмний код 1.1

```
import java.util.Comparator;

public class ColorPixel extends Pixel{
    static class ColorPixelComparatorStatic implements Comparator<ColorPixel> {
        @Override
        public int compare(ColorPixel o1, ColorPixel o2) {
            int v1 = o1.x + o1.y;
            int v2 = o2.x + o2.y;

            if (v1 < v2)
                return -1;
            if (v1 > v2)
                return 1;

            if (o1.r == o2.r && o1.g == o2.g && o1.b == o2.b)
                return 0;

            return -1;
        }
    }



    public int getR() {
        return r;
    }

    public void setR(int r) {
        this.r = r;
    }

    public int getG() {
        return g;
    }

    public void setG(int g) {
        this.g = g;
    }

    public int getB() {
        return b;
    }

    public void setB(int b) {
        this.b = b;
```

```
    }

    protected int r;
    protected int g;
    protected int b;

    public ColorPixel(int x, int y, int r, int g, int b){
        super(x, y);

        if ((r < 0 || r > 255) || (g < 0 || g > 255) || (b < 0 || b > 255))
            throw new IllegalArgumentException("RGB values must be in range from 0 to
255");

        this.r = r;
        this.g = g;
        this.b = b;
    }

    public ColorPixel(int r, int g, int b){
        super(0,0);

        if ((r < 0 || r > 255) || (g < 0 || g > 255) || (b < 0 || b > 255))
            throw new IllegalArgumentException("RGB values must be in range from 0 to
255");

        this.r = r;
        this.g = g;
        this.b = b;
    }

    @Override
    public boolean equals(Object o){
        if (!(o instanceof ColorPixel other))
            return false;

        if (this.x == other.x && this.y == other.y && this.r == other.r && this.g ==
other.g && this.b == other.b)
            return true;
        return false;
    }

    @Override
    public String toString(){
        return String.format("[X:%d ; Y:%d ; RGB: #%02x%02x%02x]", this.x, this.y,
this.r, this.g, this.b);
    }
}
```

Програмний код 1.2

```
import java.util.Comparator;

public class Pixel {
    static class PixelComparatorStatic implements Comparator<Pixel> {
        @Override
        public int compare(Pixel o1, Pixel o2) {
            int v1 = o1.x + o1.y;
            int v2 = o2.x + o2.y;

            if (v1 == v2)
                return 0;
            if (v1 < v2)
                return -1;
            return 1;
```

```
        }
    }

    public int getX() {
        return x;
    }

    public void setX(int x) {
        this.x = x;
    }

    public int getY() {
        return y;
    }

    public void setY(int y) {
        this.y = y;
    }

    protected int x;
    protected int y;

    public Pixel(int x, int y){
        this.x = x;
        this.y = y;
    }

    @Override
    public boolean equals(Object o){
        if (!(o instanceof Pixel other))
            return false;

        if (this.x == other.x && this.y == other.y)
            return true;
        return false;
    }

    @Override
    public String toString(){
        return String.format("[X:%d ; Y:%d]", this.x, this.y);
    }
}
```

Програмний код 1.3

```
Unsorted pixels array:
[X:100 ; Y:0]
[X:0 ; Y:100]
[X:50 ; Y:50]
[X:25 ; Y:6]
[X:30 ; Y:30]
[X:40 ; Y:20]
[X:66 ; Y:33]
[X:1 ; Y:2]
[X:40 ; Y:20]
[X:20 ; Y:40]
[X:0 ; Y:0]
Sorted pixels array:
[X:0 ; Y:0]
[X:1 ; Y:2]
[X:25 ; Y:6]
[X:30 ; Y:30]
[X:40 ; Y:20]
[X:40 ; Y:20]
[X:20 ; Y:40]
[X:66 ; Y:33]
[X:100 ; Y:0]
[X:0 ; Y:100]
[X:50 ; Y:50]
Unsorted colorPixels array:
[X:10 ; Y:0 ; RGB: #010101]
[X:6 ; Y:6 ; RGB: #0a0a0a]
[X:0 ; Y:10 ; RGB: #000000]
[X:12 ; Y:0 ; RGB: #030303]
[X:0 ; Y:24 ; RGB: #000000]
[X:1 ; Y:1 ; RGB: #010101]
```

Результат роботи програми 1.1

```
Sorted colorPixels array:
[X:1 ; Y:1 ; RGB: #010101]
[X:0 ; Y:10 ; RGB: #000000]
[X:10 ; Y:0 ; RGB: #010101]
[X:12 ; Y:0 ; RGB: #030303]
[X:6 ; Y:6 ; RGB: #0a0a0a]
[X:0 ; Y:24 ; RGB: #000000]
Pixels set:
[X:0 ; Y:0]
[X:1 ; Y:2]
[X:25 ; Y:6]
[X:30 ; Y:30]
[X:66 ; Y:33]
[X:100 ; Y:0]
ColorPixels set:
[X:1 ; Y:1 ; RGB: #010101]
[X:0 ; Y:10 ; RGB: #000000]
[X:10 ; Y:0 ; RGB: #010101]
[X:12 ; Y:0 ; RGB: #030303]
[X:6 ; Y:6 ; RGB: #0a0a0a]
[X:0 ; Y:24 ; RGB: #000000]

Process finished with exit code 0
```

Результат роботи програми 1.2

Висновки: Ця лабораторна робота дозволила нам глибше розібратися з використанням вкладених та внутрішніх класів в мові програмування Java, а також ознайомитися з практичними аспектами їхнього застосування, продемонструвавши використання як статичного вкладеного класу, так і анонімного класу для компараторів. В процесі виконання роботи ми змогли пригадати наші знання з цих тем і застосувати їх у практичних завданнях.