Лабораторна робота 3

1. Створити класи, специфікації яких наведені нижче. Визначити конструктори і методи setТип(), getТип(), toString(). Визначити додатково методи в класі, що створюють масив об'єктів. Задати критерій вибору даних і вивести ці дані на консоль. У кожному класі, що володіє інформацією, має бути оголошено кілька конструкторів. House: id, Номер квартири, Площа, Поверх, Кількість кімнат, Вулиця, Тип будівлі, Термін експлуатації. Створити масив об'єктів. Вивести: a) список квартир, що мають задане число кімнат; b) список квартир, що мають задане число кімнат і розташованих на поверсі, який знаходиться в заданому проміжку; c) список квартир, що мають площу, яка перевищує задану

public class House {

private int id;

private int apartmentNumber;

private double area;

private int floor;

private int numberOfRooms;

private String street;

private String buildingType;

private int yearsInUse;

public House(int id, int apartmentNumber, double area, int floor, int numberOfRooms, String street, String buildingType, int yearsInUse) {

this.id = id;

this.apartmentNumber = apartmentNumber;

this.area = area;

this.floor = floor;

this.numberOfRooms = numberOfRooms;

this.street = street;

this.buildingType = buildingType;

this.yearsInUse = yearsInUse;

}

public void setId(int id) {

this.id = id;

}

public int getId() {

return id;

}

public void setApartmentNumber(int apartmentNumber) {

this.apartmentNumber = apartmentNumber;

}

public int getApartmentNumber() {

return apartmentNumber;

}

public static House[] createHouseArray() {

House[] houses = new House[5];

houses[0] = new House(1, 101, 75.5, 2, 3, "Street 1", "Bagatokvarturni", 10);

houses[1] = new House(2, 202, 90.0, 5, 4, "Street 2", "Privatni", 5);

houses[2] = new House(3, 303, 60.0, 3, 2, "Street 3", "Bagatokvartirni", 15);

houses[3] = new House(4, 404, 110.0, 7, 5, "Street 4", "Privatni", 8);

houses[4] = new House(5, 505, 80.0, 4, 3, "Street 5", "Bagatokvartirni", 12);

return houses;

}

@Override

public String toString() {

return "House{" +

"id=" + id +

", apartmentNumber=" + apartmentNumber +

", area=" + area +

", floor=" + floor +

", numberOfRooms=" + numberOfRooms +

", street='" + street + '\'' +

", buildingType='" + buildingType + '\'' +

", yearsInUse=" + yearsInUse +

'}';

}

}

1. Реалізувати методи додавання, віднімання, множення і ділення об'єктів (для тих класів, об'єкти яких можуть підтримувати арифметичні дії). Визначити клас Інтервал з урахуванням включення/невключення кінців. Створити методи по визначенню перетину і об'єднання інтервалів, причому інтервали, що не мають загальних точок, перетинатися/об'єднуватися не можуть. Оголосити масив і n інтервалів і визначити відстань між самими віддаленими кінцями

public class Interval {

private double start;

private double end;

public Interval(double start, double end) {

this.start = start;

this.end = end;

}

public Interval add(Interval other) {

return new Interval(this.start + other.start, this.end + other.end);

}

public Interval subtract(Interval other) {

return new Interval(this.start - other.end, this.end - other.start);

}

public Interval multiply(Interval other) {

double min = Math.min(Math.min(this.start \* other.start, this.start \* other.end),

Math.min(this.end \* other.start, this.end \* other.end));

double max = Math.max(Math.max(this.start \* other.start, this.start \* other.end),

Math.max(this.end \* other.start, this.end \* other.end));

return new Interval(min, max);

}

public Interval divide(Interval other) {

if (other.start <= 0 || other.end >= 0) {

throw new IllegalArgumentException("Division by interval containing zero.");

}

double min = Math.min(Math.min(this.start / other.start, this.start / other.end),

Math.min(this.end / other.start, this.end / other.end));

double max = Math.max(Math.max(this.start / other.start, this.start / other.end),

Math.max(this.end / other.start, this.end / other.end));

return new Interval(min, max);

}

public Interval intersect(Interval other) {

double newStart = Math.max(this.start, other.start);

double newEnd = Math.min(this.end, other.end);

if (newStart > newEnd) {

return null;

}

return new Interval(newStart, newEnd);

}

public Interval union(Interval other) {

double newStart = Math.min(this.start, other.start);

double newEnd = Math.max(this.end, other.end);

return new Interval(newStart, newEnd);

}

public static double findMaxDistance(Interval[] intervals) {

double maxDistance = 0;

for (int i = 0; i < intervals.length; i++) {

for (int j = i + 1; j < intervals.length; j++) {

double distance = Math.abs(intervals[i].start - intervals[j].end);

maxDistance = Math.max(maxDistance, distance);

}

}

return maxDistance;

}

public static void main(String[] args) {

Interval interval1 = new Interval(2, 5);

Interval interval2 = new Interval(4, 7);

Interval sum = interval1.add(interval2);

System.out.println("Suma intervaliv: " + sum.start + " - " + sum.end);

Interval intersection = interval1.intersect(interval2);

if (intersection != null) {

System.out.println("Peretun intervaliv: " + intersection.start + " - " + intersection.end);

} else {

System.out.println("Intervalu ne peretunautsa.");

}

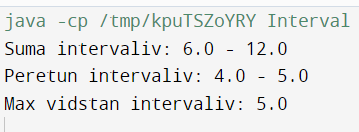
Interval[] intervals = {interval1, interval2};

double maxDistance = findMaxDistance(intervals);

System.out.println("Max vidstan intervaliv: " + maxDistance);

}

}



1. Визначити клас Коло на площині, координати центру якого задаються з допомогою Раціонального Дробу. Визначити площу і периметр. Створити масив об'єктів і визначити групи кіл, центри яких лежать на одній прямій. Визначити найбільший і найменший за площею (периметром) об'єкт.

class RationalFraction {

private int numerator;

private int denominator;

public RationalFraction(int numerator, int denominator) {

this.numerator = numerator;

if (denominator != 0) {

this.denominator = denominator;

} else {

throw new IllegalArgumentException("Denominator cannot be zero.");

}

}

public double toDouble() {

return (double) numerator / denominator;

}

}

class Circle {

private RationalFraction centerX;

private RationalFraction centerY;

private double radius;

public Circle(RationalFraction centerX, RationalFraction centerY, double radius) {

this.centerX = centerX;

this.centerY = centerY;

this.radius = radius;

}

public double getArea() {

return Math.PI \* radius \* radius;

}

public double getPerimeter() {

return 2 \* Math.PI \* radius;

}

}

public class Main {

public static void main(String[] args) {

Circle circle1 = new Circle(new RationalFraction(1, 2), new RationalFraction(3, 4), 2.5);

Circle circle2 = new Circle(new RationalFraction(2, 3), new RationalFraction(5, 6), 3.0);

Circle circle3 = new Circle(new RationalFraction(1, 2), new RationalFraction(7, 8), 1.5);

System.out.println("Ploshcha kola 1: " + circle1.getArea());

System.out.println("Perimetr kola 1: " + circle1.getPerimeter());

Circle[] circles = {circle1, circle2, circle3};

findCollinearCircles(circles);

findLargestAndSmallest(circles);

}

public static void findCollinearCircles(Circle[] circles) {

}

public static void findLargestAndSmallest(Circle[] circles) {

Circle largestCircle = null;

Circle smallestCircle = null;

double maxArea = Double.MIN\_VALUE;

double minArea = Double.MAX\_VALUE;

for (Circle circle : circles) {

double area = circle.getArea();

if (area > maxArea) {

maxArea = area;

largestCircle = circle;

}

if (area < minArea) {

minArea = area;

smallestCircle = circle;

}

}

System.out.println("Naibilshiy za ploshcheu kolo: " + largestCircle.getArea());

System.out.println("Naimenshiy za ploshcheu kolo: " + smallestCircle.getArea());

}

}

