# **Лабораторна робота №2**

**Класи**

**Мета:** Використовуючи теоретичне підґрунтя про об’єктно орієнтоване програмування виконати дії що будуть вказано в завданні до лабораторної роботи.

**Хід роботи**

**Завдання 1.** Вам необхідно реалізувати метод intersection у класі Line. Він повинен повертати точку перетину двох ліній (клас Point). Якщо лінії збігаються або перетинаються, метод повинен повертати значення null. Функція, що описує пряму y = k X + b. Користувач вводить значення k та b для двох прямих.

package lab\_02.task\_01;

public class Point {

double x;

double y;

public Point(double x, double y) {

this.x = x;

this.y = y;

}

*@Override*

public String toString() {

return "(" + x + ";" + y + ")";

}

}

Пояснення:

* Ключове слово public значить, що клас Point доступний і в інших класах.
* double x та double y - дані змінні представляють координати точки.
* public Point(double x, double y) - конструктор, який приймає два аргументи double (x та y).
* this.x = x та this.y = y - аргументи конструктора x та y присвоюються полю x та y відповідно.
* @Override - позначаємо, що метод перевизначає метод з класу Object.
* public String toString() - перевизначаємо метод toString(), який повертає рядок.
* return "(" + x + ";" + y + ")" - форматуємо рядок, а саме значення змінних x та y додаються до рядка у вигляді координат. І в резульатті цей рядок повертається з методу toString().

package lab\_02.task\_01;

import java.util.Scanner;

public class Line {

double k;

double b;

public Line(double k, double b) {

this.k = k;

this.b = b;

}

public Point intersection(Line other) {

if (this.k == other.k) {

return null;

}

double x = (other.b - this.b) / (this.k - other.k);

double y = this.k \* x + this.b;

return new Point(x, y);

}

public static void main(String[] args) {

Scanner console = new Scanner(System.***in***);

System.***out***.println("Введіть значення k та b для першої прямої:");

double k1 = console.nextDouble();

double b1 = console.nextDouble();

System.***out***.println("Введіть значення k та b для другої прямої:");

double k2 = console.nextDouble();

double b2 = console.nextDouble();

Line line1 = new Line(k1, b1);

Line line2 = new Line(k2, b2);

if (line1.intersection(line2) != null) {

System.***out***.println("Точка перетину: " + line1.intersection(line2));

} else {

System.***out***.println("Прямі збігаються або паралельні, немає точки перетину.");

}

console.close();

}

}

Пояснення:

* double та double b - змінні, що представляють коефіцієнти прямої у.
* public Line(double k, double b) - конструктор класу Line, який приймає два аргументи double і присвоює їх відповідним полям класу Line.
* public Point intersection(Line other) - метод приймає як параметр об'єкт класу Line і повертає об'єкт класу Point (точку перетину двох прямих).
* Перевіряємо якщо коефіцієнти k обох прямих рівні, тобто прямі паралельні і не мають точки перетину. Отже, метод повертає null.
* double x = (other.b - this.b) / (this.k - other.k) - обчислюємо координату x точки перетину за відповіною формулою.
* double y = this.k \* x + this.b - обчислюємо координату y точки перетину за відповіною формулою.
* Починаємо метод main.
* Зчитуємо з консолі значення k та b для двох прямих та записуємо їх у відповідні змінні.
* Створюємо об'єкти класу Line за допомогою введених значень.
* Викликаємо метод intersection для перевірки точки перетину прямих.
* Та виводимо точку перетину або повідомлення про те, що прямі паралельні або збігаються.

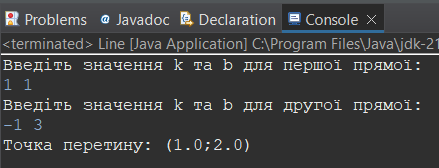


Рис. 1 - виконання завдання 1

**Завдання 2**. Реалізуйте методи класу Segment (відрізок):

Конструктор, в який як параметри передаються координати точок початку і кінця відрізка (використовуйте клас Point). Переконайтеся, що створений відрізок існує і не є виродженим, що означає, що початок і кінець відрізка не є однією точкою. Реалізуйте метод double length() – повертає довжину відрізка. Реалізуйте метода Point middle() – повертає середню точку відрізка. Реалізуйте метод Point intersection(Segment another) – повертає точку перетину поточного відрізка з іншим.

package lab\_02.task\_02;

import java.util.Scanner;

public class Segment {

private Point start;

private Point end;

public Segment(Point start, Point end) {

if (start.x == end.x && start.y == end.y) {

System.***err***.println("Початкова та кінцева точки відрізка не можуть співпадати");

return;

}

this.start = start;

this.end = end;

}

public double length() {

return Math.*sqrt*(Math.*pow*(end.x - start.x, 2) + Math.*pow*(end.y - start.y, 2));

}

public Point middle() {

double midX = (start.x + end.x) / 2;

double midY = (start.y + end.y) / 2;

return new Point(midX, midY);

}

public Point intersection(Segment another) {

double x1 = start.x;

double y1 = start.y;

double x2 = end.x;

double y2 = end.y;

double x3 = another.start.x;

double y3 = another.start.y;

double x4 = another.end.x;

double y4 = another.end.y;

double d = (x1 - x2) \* (y3 - y4) - (y1 - y2) \* (x3 - x4);

if (d == 0) {

return null;

}

double xi = ((x1 \* y2 - y1 \* x2) \* (x3 - x4) - (x1 - x2) \* (x3 \* y4 - y3 \* x4)) / d;

double yi = ((x1 \* y2 - y1 \* x2) \* (y3 - y4) - (y1 - y2) \* (x3 \* y4 - y3 \* x4)) / d;

Point intersectionPoint = new Point(xi, yi);

if (xi < Math.*min*(x1, x2) || xi > Math.*max*(x1, x2) ||

xi < Math.*min*(x3, x4) || xi > Math.*max*(x3, x4) ||

yi < Math.*min*(y1, y2) || yi > Math.*max*(y1, y2) ||

yi < Math.*min*(y3, y4) || yi > Math.*max*(y3, y4)) {

return null;

}

return intersectionPoint;

}

public static void main(String[] args) {

Scanner console = new Scanner(System.***in***);

System.***out***.println("Введіть координати початкової точки першого відрізка:");

double x1 = console.nextDouble();

double y1 = console.nextDouble();

System.***out***.println("Введіть координати кінцевої точки першого відрізка:");

double x2 = console.nextDouble();

double y2 = console.nextDouble();

System.***out***.println("Введіть координати точки другого відрізка:");

double x3 = console.nextDouble();

double y3 = console.nextDouble();

System.***out***.println("Введіть координати кінцевої точки другого відрізка:");

double x4 = console.nextDouble();

double y4 = console.nextDouble();

Segment segment1 = new Segment(new Point(x1, y1), new Point(x2, y2));

Segment segment2 = new Segment(new Point(x3, y3), new Point(x4, y4));

System.***out***.printf("Довжина першого відрізка: %.2f\n", segment1.length());

Point middlePoint = segment1.middle();

System.***out***.println("Середня точка першого відрізка: " + middlePoint);

Point intersectionPoint = segment1.intersection(segment2);

if (intersectionPoint != null) {

System.***out***.println("Точка перетину відрізків: " + intersectionPoint);

} else {

System.***out***.println("Відрізки не перетинаються або паралельні.");

}

console.close();

}

}

Пояснення:

* private Point start та private Point end - приватні поля, що використовуються для зберігання початкової та кінцевої точки відрізка. До них можна звертатися лише з інших методів класу Segment.
* public Segment(Point start, Point end) - конструктор, який приймає два параметри Point, а саме початкову точку та кінцуву.
* if (start.x == end.x && start.y == end.y) - перевіряємо, чи координати початкової точки start співпадають з координатами кінцевої точки end. Якщо так, то виводиться повідомлення про помилку у вихідний потік помилок System.err і виконання конструктору завершується за допомогою return.
* this.start = start та this.end = end - присвоюємо значення параметра start та end приватному полю start.
* Створємо метод length(), що обчислює довжину відрізка.
* Math.pow(end.x - start.x, 2) - обчислюємо квадрат різниці між координатами x кінцевої точки і початкової точки .
* Аналогічно обчислюємо для y.
* Math.sqrt(Math.pow(end.x - start.x, 2) + Math.pow(end.y - start.y, 2)) - знаходимо квадратний корінь цбого виразу.
* Далі створюємо метод middle(), що обчислює середню точку відрізка.
* double midX = (start.x + end.x) / 2 - знаходимо середнє значення координат x початкової точки і кінцевої точки.
* Аналогічно шукаємо y середньої точки.
* return new Point(midX, midY) - створюємо новий об'єкт класу Point з координатами midX та midY, що і є середньою точкою.
* Створюємо метод intersection(Segment another) для знаходження точки перетину поточного відрізка з іншим відрізком.
* Спочатку отримуємо координати початкової та кінцевої точок поточного відрізка і відрізка another.
* double d = (x1 - x2) \* (y3 - y4) - (y1 - y2) \* (x3 - x4) -визначаємо за відповідноїю формулою значення d. Якщо d дорівнює 0, nj відрізки паралельні або лежать на одній прямій, тому метод повертає null.
* Якщо d не дорівнює 0, обчислюємо координати точки перетину xi та yi за відповідними формулами.
* Та створюємо новий об'єкт intersectionPoint класу Point з координатами (xi, yi).
* Далі перевіряємо, чи знаходиться точка перетину на відрізках. Для цього порівнянюємо координати точки перетину з максимальними та мінімальними координатами кожного відрізка. Якщо точка перетину виходить за межі будь-якого з відрізків, то повертається null.
* В методі main створюємо об'єкт класу Scanner, який дозволяє зчитувати введення з консолі.
* Користувач виводить координати початкової та кінцевої точок першого та другого відрізків.
* Створюємо об'єкти segment1 та segment2 класу Segment з введеними координатами.
* Виводиимо довжину першого відрізка за допомогою методу length().
* Виводимо середню точку першого відрізка за допомогою методу middle().
* Викликаємо метод intersection() першого відрізка з другим відрізком для знаходження точки перетину. Якщо точка перетину існує, вона виводиться на екран. В іншому випадку виводиться повідомлення, що відрізки не перетинаються або паралельні.

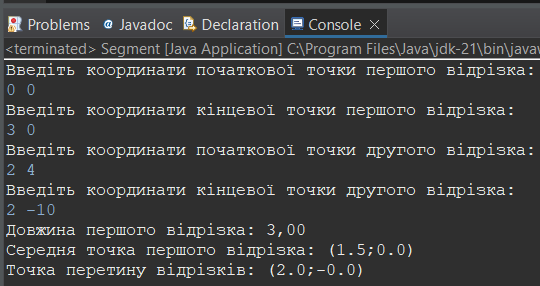


Рис. 2 - виконання завдання 2

**Завдання 3.** Реалізуйте методи класу Triangle: Конструктор, що має як параметри координати трьох вершин (клас Point). Переконайтеся, що ці точки належать до вершин трикутника. Перевірте, що створений трикутник існує і не вироджений. Реалізуйте метод double area() – повертає площу трикутника. Реалізуйте метод Point centroid() – повертає центроїд трикутника.

package lab\_02.task\_03;

import java.util.Scanner;

public class Triangle {

private Point vertex1;

private Point vertex2;

private Point vertex3;

public Triangle(Point vertex1, Point vertex2, Point vertex3) {

if (areVerticesCollinear(vertex1, vertex2, vertex3)) {

throw new IllegalArgumentException("Три точки не можуть бути на о дній прямій");

}

if (!isValidTriangle(vertex1, vertex2, vertex3)) {

throw new IllegalArgumentException("Три точки не утворюють трикутник");

}

this.vertex1 = vertex1;

this.vertex2 = vertex2;

this.vertex3 = vertex3;

}

private boolean areVerticesCollinear(Point vertex1, Point vertex2, Point vertex3) {

double area = 0.5 \* Math.*abs*((vertex2.x - vertex1.x) \* (vertex3.y - vertex1.y) - (vertex3.x - vertex1.x) \* (vertex2.y - vertex1.y));

return area == 0;

}

private boolean isValidTriangle(Point vertex1, Point vertex2, Point vertex3) {

double side1 = calculateDistance(vertex1, vertex2);

double side2 = calculateDistance(vertex1, vertex3);

double side3 = calculateDistance(vertex2, vertex3);

return side1 + side2 > side3 && side1 + side3 > side2 && side2 + side3 > side1;

}

private double calculateDistance(Point p1, Point p2) {

return Math.*sqrt*(Math.*pow*(p2.x - p1.x, 2) + Math.*pow*(p2.y - p1.y, 2));

}

public double area() {

double side1 = calculateDistance(vertex1, vertex2);

double side2 = calculateDistance(vertex1, vertex3);

double side3 = calculateDistance(vertex2, vertex3);

double p = (side1 + side2 + side3) / 2;

return Math.*sqrt*(p \* (p - side1) \* (p - side2) \* (p - side3));

}

public Point centroid() {

double centroidX = (vertex1.x + vertex2.x + vertex3.x) / 3;

double centroidY = (vertex1.y + vertex2.y + vertex3.y) / 3;

return new Point(centroidX, centroidY);

}

public static void main(String[] args) {

Scanner console = new Scanner(System.***in***);

System.***out***.println("Введіть координати трьох вершин трикутника:");

System.***out***.println("Вершина A:");

double x1 = console.nextDouble();

double y1 = console.nextDouble();

System.***out***.println("Вершина B:");

double x2 = console.nextDouble();

double y2 = console.nextDouble();

System.***out***.println("Вершина C:");

double x3 = console.nextDouble();

double y3 = console.nextDouble();

Triangle triangle = new Triangle(new Point(x1, y1), new Point(x2, y2), new Point(x3, y3));

System.***out***.printf("Площа трикутника: %.2f\n", triangle.area());

Point centroid = triangle.centroid();

System.***out***.println("Координати центроїду трикутника: (" + centroid.x + ", " + centroid.y + ")");

console.close();

}

}

Пояснення:

* Створємо клас Triangle та визначаємо приватні поля vertex1, vertex2 і vertex3, які представляють вершини трикутника.
* Далі створюємо клас Triangle, який приймає вершини трикутника.
* if (areVerticesCollinear(vertex1, vertex2, vertex3)) - перевіряємо чи лежать передані вершини на одній прямій, викликаючи метод areVerticesCollinear(). Якщо вершини лежать на одній прямій, генерується виключення IllegalArgumentException з повідомленням "Три точки не можуть бути на одній прямій".
* if (!isValidTriangle(vertex1, vertex2, vertex3)) - перевіряємо чи утворюють передані вершини дійсний трикутник, викликаючи метод isValidTriangle(). Якщо вершини не утворюють дійсний трикутник, генерується виключення IllegalArgumentException з повідомленням "Три точки не утворюють трикутник".
* this.vertex1 = vertex1, this.vertex2 = vertex2 та this.vertex3 = vertex3 - ініціалізуємо поля об'єкта значеннями вершин, що були переданими конструктору.
* Створюємо приватний метод areVerticesCollinear для перевірки чи лежать три вершини трикутника на одній прямій.
* Обчислюємо площу трикутника використовуючи координати трьох вершин.
* Якщо площа трикутника, обчислена за допомогою формули дорівнює нулю, то повертається true. Якщо площа не дорівнює нулю, метод повертає false, що вказує на те, що трикутник існує.
* Створюємо приватний метод isValidTriangle для перевірки чи задані три точки утворюють дійсний трикутник.
* Обчислюємо довжину усіх відрізків, які з'єднують вершини трикутника. Для цього використовуємо метод calculateDistance.
* return side1 + side2 > side3 && side1 + side3 > side2 && side2 + side3 > side1 - перевіряємо умову існування трикутника, використовуючи теорему, що кожна сторона трикутника менша за суму двох інших сторін. Якщо ця умова виконується, метод повертає true, тобто три точки утворюють дійсний трикутник. В іншому випадку метод повертає false.
* Створюємо метод CalculateDistance для обчислення відстані між двома.
* Math.pow(p2.x - p1.x, 2) - обчислюємо квадрат різниці між значеннями координати x другої точки і координати x першої точки.
* Аналогічно обчислюємо для y.
* Та знаходимо корінь суми цих квадратів.
* Створюємо метод area(), який обчислює площу трикутника.
* Знаходимо довжини сторін трикутника за допомогою методу calculateDistance.
* double p = (side1 + side2 + side3) / 2 - знаходимо півпериметр трикутника.
* Та обчислюємо площу за формулою Герона.
* Створюємо метод centroid(), який обчислює центроїд трикутника.
* double centroidX = (vertex1.x + vertex2.x + vertex3.x) / 3 - знаходимо координату x центроїда, що обчислюється як середнє арифметичне x-координат вершин трикутника.
* Аналогічно знаходимо координату y.
* Створюємо нову точку, яка і є центроїдом трикутника.
* В методі main зчитуємо введені користувачем координати вершин трикутника.
* Ці координати передаємо як аргументи конструктору класу Triangle. Конструктор перевіряє, чи можна створити трикутник з такими вершинами, та ініціалізує поля об'єкта Triangle.
* Для обчислення площі викликаємо метод area(), для обчислення центроїду викликаємо метод centroid().
* Виводимо відповідний результат на екран.

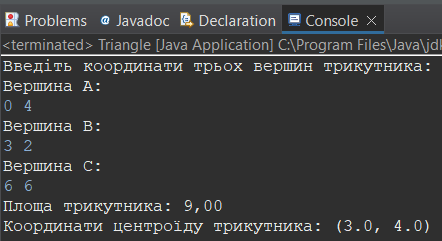


Рис. 3 - виконання завдання 3

**Висновок.** Під час виконання даної лабораторної роботи я икористовуючи теоретичне підґрунтя про об’єктно орієнтоване програмування виконала дії що були вказані в завданні до лабораторної роботи.