**ЛАБОРАТОРНА РОБОТА №2**

**Класи**

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

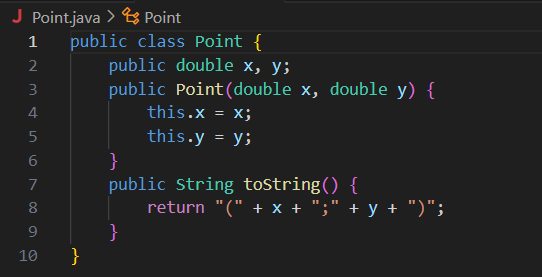
Line line1 = new Line(1,1);

Line line2 = new Line(-1,3);

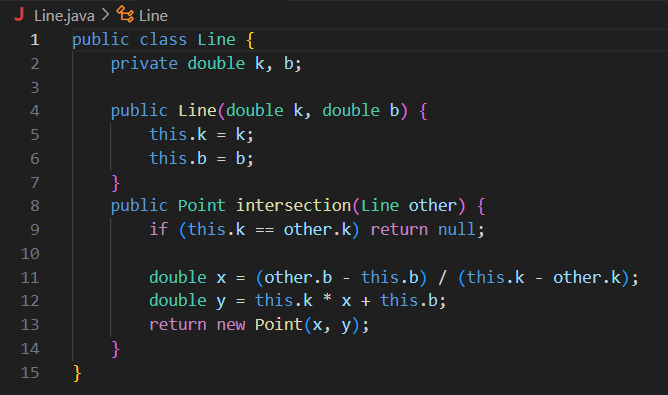
System.out.println(line1.intersection(line2)); // (1;2)

**Хід виконання завдання:**

**Файл Point.java**

****

**Файл Line.java**

****

**Файл Main.java**

****

**Вивід:**

****

**Line l1 = new Line(1, 1), Line l2 = new Line(1, 1):**

****

**Текст програми:**

**Файл Point.java**

|  |
| --- |
| public class Point {  public double x, y;  public Point(double x, double y) {  this.x = x;  this.y = y;  }  public String toString() {  return "(" + x + ";" + y + ")";  }  } |

**Файл Line.java**

|  |
| --- |
| public class Line {  private double k, 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);  }  } |

**Файл Main.java**

|  |
| --- |
| public class Main {  public static void main(String[] args) {  System.out.println("--- Line ---");  Line l1 = new Line(1, 1);  Line l2 = new Line(-1, 3);  System.out.println("Перетин: " + l1.intersection(l2)); // (1.0;2.0)  System.out.println("\n--- Segment ---");  Segment s1 = new Segment(new Point(0, 0), new Point(2, 2));  Segment s2 = new Segment(new Point(0, 2), new Point(2, 0));  System.out.println("Довжина: " + s1.length());  System.out.println("Середина: " + s1.middle());  System.out.println("Перетин: " + s1.intersection(s2)); // (1.0;1.0)  System.out.println("\n--- Triangle ---");  Triangle t = new Triangle(new Point(0, 0), new Point(4, 0), new Point(2, 3));  System.out.println("Площа: " + t.area());  System.out.println("Центроїд: " + t.centroid()); // (2.0;1.0)  }  } |

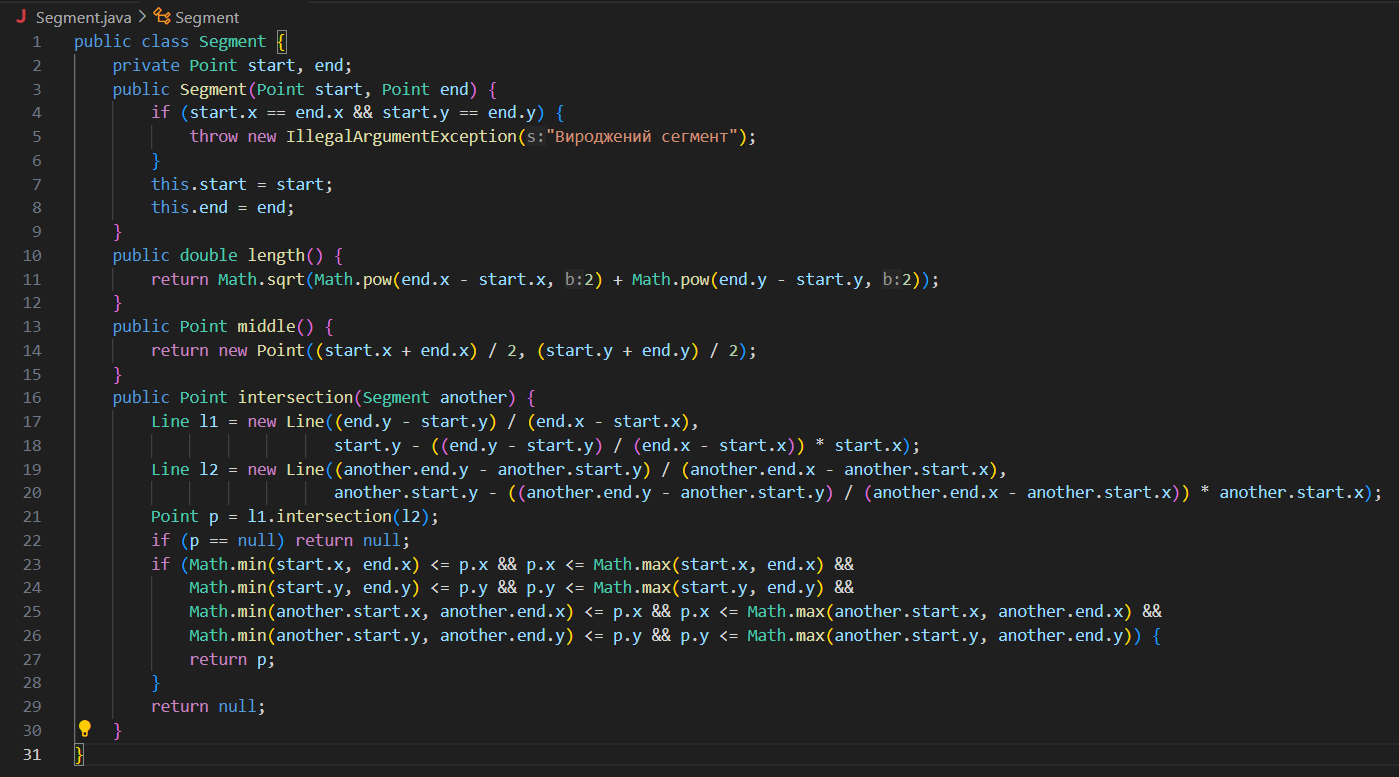
**Завдання №2.** Реалізуйте методи класу Segment (відрізок): Конструктор, в який як параметри передаються координати точок початку і кінця відрізка (використовуйте клас Point). Переконайтеся, що створений відрізок існує і не є виродженим, що означає, що початок і кінець відрізка не є однією точкою.

Реалізуйте метод double length() – повертає довжину сегмента.

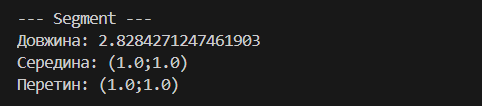
Реалізуйте метода Point middle() – повертає середню точку сегмента.

Реалізуйте метод Point intersection(Segment another) – повертає точку перетину поточного відрізка з іншим.

**Хід виконання завдання:**

****

**Вивід:**

****

**Текст програми:**

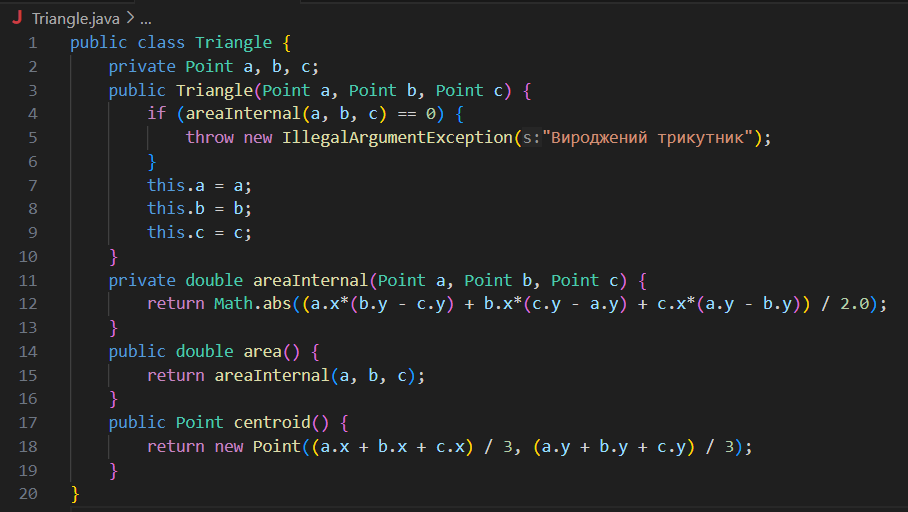
|  |
| --- |
| public class Segment {  private Point start, end;  public Segment(Point start, Point end) {  if (start.x == end.x && start.y == end.y) {  throw new IllegalArgumentException("Вироджений сегмент");  }  this.start = start;  this.end = end;  }  public double length() {  return Math.sqrt(Math.pow(end.x - start.x, 2) + Math.pow(end.y - start.y, 2));  }  public Point middle() {  return new Point((start.x + end.x) / 2, (start.y + end.y) / 2);  }  public Point intersection(Segment another) {  Line l1 = new Line((end.y - start.y) / (end.x - start.x),  start.y - ((end.y - start.y) / (end.x - start.x)) \* start.x);  Line l2 = new Line((another.end.y - another.start.y) / (another.end.x - another.start.x),  another.start.y - ((another.end.y - another.start.y) / (another.end.x - another.start.x)) \* another.start.x);  Point p = l1.intersection(l2);  if (p == null) return null;  if (Math.min(start.x, end.x) <= p.x && p.x <= Math.max(start.x, end.x) &&  Math.min(start.y, end.y) <= p.y && p.y <= Math.max(start.y, end.y) &&  Math.min(another.start.x, another.end.x) <= p.x && p.x <= Math.max(another.start.x, another.end.x) &&  Math.min(another.start.y, another.end.y) <= p.y && p.y <= Math.max(another.start.y, another.end.y)) {  return p;  }  return null;  }  } |

**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.

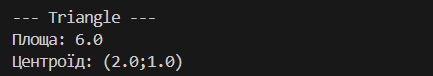
Реалізуйте метод double area() – повертає площу трикутника.

Реалізуйте метод Point centroid() – повертає центроїд трикутника.

**Хід виконання завдання:**

****

**Вивід:**

****

**Текст програми:**

|  |
| --- |
| public class Triangle {  private Point a, b, c;  public Triangle(Point a, Point b, Point c) {  if (areaInternal(a, b, c) == 0) {  throw new IllegalArgumentException("Вироджений трикутник");  }  this.a = a;  this.b = b;  this.c = c;  }  private double areaInternal(Point a, Point b, Point c) {  return Math.abs((a.x\*(b.y - c.y) + b.x\*(c.y - a.y) + c.x\*(a.y - b.y)) / 2.0);  }  public double area() {  return areaInternal(a, b, c);  }  public Point centroid() {  return new Point((a.x + b.x + c.x) / 3, (a.y + b.y + c.y) / 3);  }  } |

Посилання на github: <https://github.com/Lizarebenokk/OOP-KB-232-Liza-Rebenok/tree/main>