**Лабораторна робота №2**

**Тема**: Класи.

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

**Хід роботи**

**Завдання 1**

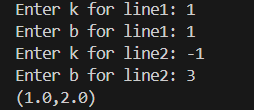
**Умова:** Вам необхідно реалізувати метод intersection у класі Line. Він повинен повертати точку перетину двох ліній (клас Point). Якщо лінії збігаються або перетинаються, метод повинен повертати значення null. Функція, що описує пряму y = k X + b. Користувач вводить значення k та b для двох прямих.

**Виконання:** користувач вводить дані для двох прямих, після цього створюється 2 об’єкти класу Line, які приймають атрибутами введені значення, після цього створюється об’єкт класу Point, його значення повертається методом intersection одного з об’єктів класу Line, куди, як аргумент вводиться інший об’єкт класу Line. Метод intersection поверне null, якщо прямі паралельні, або накладаються, також в об’єкта класу Point є метод, що повертає координати точки в зручному форматі запису.

**Текст програми**:

|  |
| --- |
| import java.util.Scanner;  public class task\_1 {      public static void main(String[] args) {          Scanner input = new Scanner(System.in);          System.out.print("Enter k for line1: ");          float k = input.nextFloat();          System.out.print("Enter b for line1: ");          float b = input.nextFloat();          Line line\_1 = new Line(k, b);          System.out.print("Enter k for line2: ");           k = input.nextFloat();          System.out.print("Enter b for line2: ");           b = input.nextFloat();          Line line\_2 = new Line(k, b);            Point point = line\_1.intersection(line\_2);          if (point == null){              System.out.println("Lines are parallel or coincident");          } else{              System.out.println(point.Output());          }         input.close();      }      private static class Point{          private float x, y;          public Point(float x, float y){              this.x = x;              this.y = y;          }          public String Output(){              return ("(" + x + "," + y + ")");          }      }      private static class Line{          private float k, b;          public Line(float k, float b){              this.k = k;              this.b = b;          }          public Point intersection(Line line\_2){              if (this.k == line\_2.k){                  return null;              }              float x = ((line\_2.b - this.b) / (this.k - line\_2.k));              float y = this.k \* x + this.b;              return new Point(x, y);          }      }  } |

**Результат виконання:**



**Завдання 2**

**Умова:** Реалізуйте методи класу Segment (відрізок):

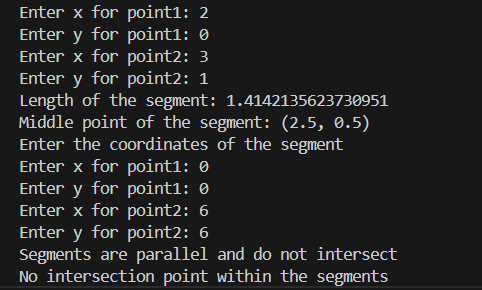
* Конструктор, в який як параметри передаються координати точок початку і кінця відрізка (використовуйте клас Point).
* Переконайтеся, що створений відрізок існує і не є виродженим, що означає, що початок і кінець відрізка не є однією точкою.
* Реалізуйте метод double length() – повертає довжину сегмента.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

**Виконання:** користувач вводить дані для 1 відрізка, і створюється об’єкт класу Segment, який приймає атрибутами 2 об’єкти классу Point, які в свою чергу приймають атрибутами введені значення, далі за допомогою методів класу Segment виводиться довжина та середина цього відрізка, після цього вводяться дані для 2 відрізка, потім створюється об’єкт класу Point, що є точкою перетину цих відрізків, його значення отримується методом intersection класу Segment, який виводить рівняння прямих за допомогою методу equation та шукає точку перетину, якщо вона знаходиться в межах відрізків, то вона існує, якщо ні, то вона є тільки точкою перетину прямих цих відрізків.

**Текст програми**:

|  |
| --- |
| import java.util.Scanner;  public class task\_2 {      public static void main(String[] args) {          Scanner input = new Scanner(System.in);          Segment seg1 = readSegment(input);          System.out.println("Length of the segment: " + seg1.length());          System.out.println("Middle point of the segment: " + seg1.middle().Output());          Segment seg2 = readSegment(input);          Point intersection = seg1.intersection(seg2);          if (intersection != null) {              System.out.println("Point of intersection of segments: " + intersection.Output());          } else {              System.out.println("No intersection point within the segments");          }          input.close();      }      private static Segment readSegment(Scanner input) {          Segment segment;          while (true) {              System.out.println("Enter the coordinates of the segment");              System.out.print("Enter x for point1: ");              double x = input.nextDouble();              System.out.print("Enter y for point1: ");              double y = input.nextDouble();              Point p1 = new Point(x, y);              System.out.print("Enter x for point2: ");               x = input.nextDouble();              System.out.print("Enter y for point2: ");               y = input.nextDouble();              Point p2 = new Point(x, y);              segment = new Segment(p1, p2);              if (segment.isValid()) {                  break;              } else {                  System.out.println("Points coincide. Try again.");              }          }          return segment;      }      private static class Point {          private double x, y;          public Point(double x, double y) {              this.x = x;              this.y = y;          }          public boolean equals(Point other) {              return Math.abs(this.x - other.x) < 1e-6 && Math.abs(this.y - other.y) < 1e-6;          }          public String Output() {              return "(" + x + ", " + y + ")";          }          public boolean isBetween(Point a, Point b) {              return Math.min(a.x, b.x) - 1e-6 <= x && x <= Math.max(a.x, b.x) + 1e-6 &&                     Math.min(a.y, b.y) - 1e-6 <= y && y <= Math.max(a.y, b.y) + 1e-6;          }      }      private static class Segment {          private Point p1, p2;          public Segment(Point p1, Point p2) {              this.p1 = p1;              this.p2 = p2;          }          public boolean isValid() {              return !p1.equals(p2);          }          public double length() {              return Math.sqrt(Math.pow(p1.x - p2.x, 2) + Math.pow(p1.y - p2.y, 2));          }          public Point middle() {              return new Point((p1.x + p2.x) / 2, (p1.y + p2.y) / 2);          }          public Point intersection(Segment other) {              double a1 = p2.y - p1.y;              double b1 = p1.x - p2.x;              double c1 = a1 \* p1.x + b1 \* p1.y;              double a2 = other.p2.y - other.p1.y;              double b2 = other.p1.x - other.p2.x;              double c2 = a2 \* other.p1.x + b2 \* other.p1.y;              double determinant = a1 \* b2 - a2 \* b1;              if (Math.abs(determinant) < 1e-6) {                  if (Math.abs(a1 \* other.p1.x + b1 \* other.p1.y - c1) < 1e-6) {                      System.out.println("Segments lie on the same line (may overlap)");                  } else {                      System.out.println("Segments are parallel and do not intersect");                  }                  return null;              }              double x = (b2 \* c1 - b1 \* c2) / determinant;              double y = (a1 \* c2 - a2 \* c1) / determinant;              Point intersect = new Point(x, y);              if (intersect.isBetween(p1, p2) && intersect.isBetween(other.p1, other.p2)) {                  return intersect;              }              return null;          }      }  } |

**Результат виконання:**



**Завдання 3**

**Умова:** Реалізуйте методи класу Triangle:

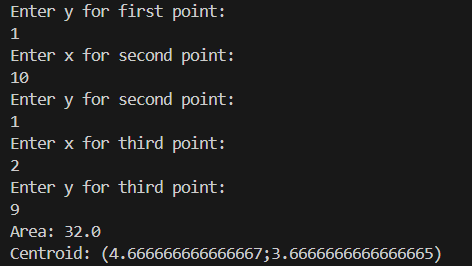
* Конструктор, що має як параметри координати трьох вершин (клас Point).
* Переконайтеся, що ці точки належать до вершин трикутника.
* Перевірте, що створений трикутник існує і не вироджений.
* Реалізуйте метод double area() – повертає площу трикутника.
* Реалізуйте метод Point centroid() – повертає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

**Виконання:** користувач вводить дані для 3 точок, що є вершинами , і створюється об’єкт класу Triangle, далі за допомогою метода area класу Triangle за формулою Герона обчислюється площа трикутника, після чого за допомогою метода centroid знаходяться координати центроїду трикутника, це точка перетину медіан трикутника, для цього достатньо знайти координати середин двох сторін та скласи рівняння прямих за допомогою вершин, що лежать навпроти цих сторін і знайти перетин цих прямих.

**Текст програми**:

|  |
| --- |
| import java.util.Scanner;  public class task\_3 {      public static void main(String[] args) {          Scanner input = new Scanner(System.in);            System.out.println("Enter x for first point: ");          double x = input.nextDouble();          System.out.println("Enter y for first point: ");          double y = input.nextDouble();          Point p1 = new Point(x, y);          System.out.println("Enter x for second point: ");          x = input.nextDouble();          System.out.println("Enter y for second point: ");          y = input.nextDouble();          Point p2 = new Point(x, y);          System.out.println("Enter x for third point: ");          x = input.nextDouble();          System.out.println("Enter y for third point: ");          y = input.nextDouble();          Point p3 = new Point(x, y);          Triangle t = new Triangle(p1, p2, p3);          System.out.println("Area: " + t.area());          System.out.println("Centroid: " + t.centroid().Output());            input.close();      }      private static class Point {          double x, y;        public Point(double x, double y) {              this.x = x;              this.y = y;          }          public String Output() {              return "(" + x + ";" + y + ")";          }      }     private static class Triangle {          Point a, b, c;          Triangle(Point a, Point b, Point c) {              if (area(a, b, c) == 0) {                  throw new IllegalArgumentException("Degenerate triangle");              }              this.a = a;              this.b = b;              this.c = c;          }         public double area() {              return area(a, b, c);          }          public static double area(Point p1, Point p2, Point p3) {              return Math.abs((p1.x \* (p2.y - p3.y) + p2.x \* (p3.y - p1.y) + p3.x \* (p1.y - p2.y)) / 2);          }         public Point centroid() {              double x = (a.x + b.x + c.x) / 3;              double y = (a.y + b.y + c.y) / 3;              return new Point(x, y);          }      }  } |

**Результат виконання:**



**Висновок:** під час виконання роботи я використовуючи теоретичне підґрунтя про об’єктно орієнтоване програмування виконував завдання до лабораторної роботи.