# 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 до лабораторної роботи

Для виконання лабораторної роботи необхідно використовувати репозиторій на github, що був створений при виконанні завдання до першої лабораторної роботи. Результати роботи необхідно буде завантажити на github для перевірки.

При використанні IDE створити проект під назвою lab02 для розміщення результатів виконання завдань. У випадку використання звичайного текстового редактору створити окрему директорію lab02 для виконання завдань.

Виконайте наступні завдання(кожне завдання має бути виконано в окремому файлі):

|  |  |
| --- | --- |
| 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)  [додаткова інформація](https://en.wikipedia.org/wiki/Line%E2%80%93line_intersection#Given_two_line_equations) |
| 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.  [Length-of-a-Line](https://www.wikihow.com/Use-Distance-Formula-to-Find-the-Length-of-a-Line)  [Midpoint](https://www.wikihow.com/Find-the-Midpoint-of-a-Line-Segment#Use-the-Midpoint-Formula)  [two\_points\_on\_each\_line](https://en.wikipedia.org/wiki/Line%E2%80%93line_intersection#Given_two_points_on_each_line) |
| 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.  [Existence\_of\_a\_triangle](https://en.wikipedia.org/wiki/Triangle#Existence_of_a_triangle)  [Computing\_the\_area\_of\_a\_triangle](https://en.wikipedia.org/wiki/Triangle#Computing_the_area_of_a_triangle)  [Centroid](https://en.wikipedia.org/wiki/Centroid) |
|  |  |
|  |  |

Сформуйте звіт який містить код програм та результати тестування.

Надайте посилання на github для перевірки.

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)

Код:

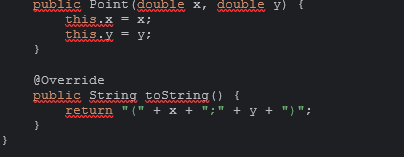
package lab2;  
  
class Point {  
 double x, y;  
  
 public Point(double x, double y) {  
 this.x = x;  
 this.y = y;  
 }  
  
 @Override  
 public String toString() {  
 return "(" + x + ";" + y + ")";  
 }  
}  
  
class Line {  
 double k, b;  
  
 public Line(double k, double b) {  
 this.k = k;  
 this.b = b;  
 }  
  
 public Point intersection(Line other) {  
 // Якщо коефіцієнти k однакові  
 if (this.k == other.k) {  
 // Якщо вільні члени також однакові — лінії збігаються  
 if (this.b == other.b) {  
 return null; // нескінченно багато точок  
 } else {  
 return null; // паралельні — не перетинаються  
 }  
 }  
  
 // Знаходимо x, y  
 double x = (other.b - this.b) / (this.k - other.k);  
 double y = this.k \* x + this.b;  
  
 return new Point(x, y);  
 }  
}  
  
public class task1{public static void main(String[] args) {  
 Line line1 = new Line(1, 1);  
 Line line2 = new Line(-1, 3);  
  
 System.out.println(line1.intersection(line2)); // (1.0;2.0)  
}  
}

Цей код:

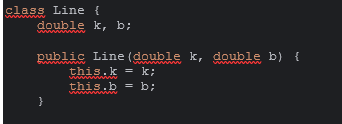
визначає класи **Point** (точка) і **Line** (пряма),

обчислює точку перетину двох ліній виду y=kx+by = kx + by=kx+b,

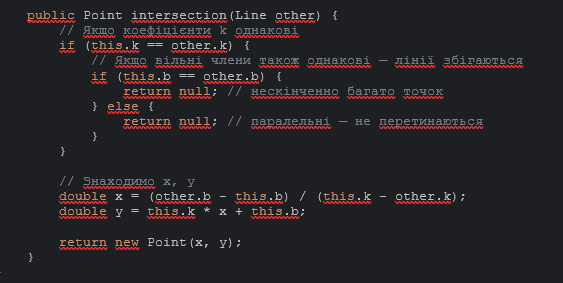
виводить координати цієї точки.



Клас Point,описує точку на координатній площині. x, y — координати точки.Метод toString() повертає координати як рядок, наприклад: (1.0;2.0).



Клас Line, Описує лінію рівняння: **y = kx + b** k — кутовий коефіцієнт (нахил),b — вільний член (зсув по y).



Метод intersection(...)

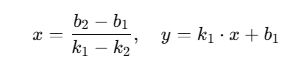
Якщо **кутові коефіцієнти однакові**:

Якщо ще й b однакові → лінії **збігаються** → повертає null.

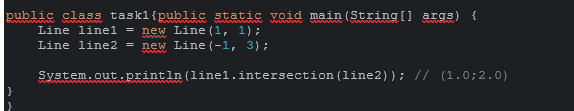
Якщо b різні → **паралельні** → також null.

Інакше — **перетинаються**:

Обчислюється координата перетину:



Повертається точка як Point.

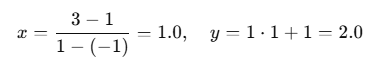


Створюються дві прямі:

line1: y=1x+1y = 1x + 1y=1x+1

line2: y=−1x+3y = -1x + 3y=−1x+3

Викликається intersection(), яка повертає точку перетину:

Результат: 

Виводиться: (1.0;2.0)

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.

[Length-of-a-Line](https://www.wikihow.com/Use-Distance-Formula-to-Find-the-Length-of-a-Line)

[Midpoint](https://www.wikihow.com/Find-the-Midpoint-of-a-Line-Segment#Use-the-Midpoint-Formula)

[two\_points\_on\_each\_line](https://en.wikipedia.org/wiki/Line%E2%80%93line_intersection#Given_two_points_on_each_line)

package lab2;  
  
class Segment {  
 Point p1, p2;  
  
 public Segment(Point p1, Point p2) {  
 if (p1.x == p2.x && p1.y == p2.y) {  
 throw new IllegalArgumentException("Segment cannot be degenerate (same start and end point)");  
 }  
 this.p1 = p1;  
 this.p2 = p2;  
 }  
  
 public double length() {  
 double dx = p2.x - p1.x;  
 double dy = p2.y - p1.y;  
 return Math.sqrt(dx \* dx + dy \* dy);  
 }  
  
 public Point middle() {  
 return new Point((p1.x + p2.x) / 2, (p1.y + p2.y) / 2);  
 }  
  
 public Point intersection(Segment other) {  
 double x1 = this.p1.x, y1 = this.p1.y;  
 double x2 = this.p2.x, y2 = this.p2.y;  
 double x3 = other.p1.x, y3 = other.p1.y;  
 double x4 = other.p2.x, y4 = other.p2.y;  
  
 double denom = (x1 - x2) \* (y3 - y4) - (y1 - y2) \* (x3 - x4);  
  
 if (denom == 0) {  
 return null; // лінії паралельні або збігаються  
 }  
  
 double px = ((x1\*y2 - y1\*x2)\*(x3 - x4) - (x1 - x2)\*(x3\*y4 - y3\*x4)) / denom;  
 double py = ((x1\*y2 - y1\*x2)\*(y3 - y4) - (y1 - y2)\*(x3\*y4 - y3\*x4)) / denom;  
  
 if (isBetween(px, py, x1, y1, x2, y2) && isBetween(px, py, x3, y3, x4, y4)) {  
 return new Point(px, py);  
 } else {  
 return null;  
 }  
 }  
  
 private boolean isBetween(double px, double py, double x1, double y1, double x2, double y2) {  
 return px >= Math.min(x1, x2) && px <= Math.max(x1, x2) &&  
 py >= Math.min(y1, y2) && py <= Math.max(y1, y2);  
 }  
}  
  
public class task2 {  
 public static void main(String[] args) {  
 Segment seg1 = new Segment(new Point(0, 0), new Point(4, 4));  
 Segment seg2 = new Segment(new Point(0, 4), new Point(4, 0));  
  
 System.out.println("Length seg1: " + seg1.length());  
 System.out.println("Middle seg2: " + seg2.middle());  
 System.out.println("Intersection: " + seg1.intersection(seg2)); // (2.0;2.0)  
 }  
}

Цей код описує клас Segment — відрізок на площині, заданий двома точками p1 і p2. Конструктор перевіряє, щоб точки не збігалися (відрізок не вироджений).

Метод length() обчислює довжину відрізка за теоремою Піфагора.

Метод middle() знаходить середину відрізка як середнє між координатами.

Метод intersection() обчислює точку перетину з іншим відрізком, якщо вона існує та лежить в межах обох відрізків.

У main() створюються два перехрещені відрізки, для яких виводиться довжина, середина та точка перетину.

Результат: (2.0;2.0) — це точка, де вони перетинаються.

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.

[Existence\_of\_a\_triangle](https://en.wikipedia.org/wiki/Triangle#Existence_of_a_triangle)

[Computing\_the\_area\_of\_a\_triangle](https://en.wikipedia.org/wiki/Triangle#Computing_the_area_of_a_triangle)

package lab2;  
  
class Triangle {  
 Point a, b, c;  
  
 public Triangle(Point a, Point b, Point c) {  
 if (areColinear(a, b, c)) {  
 throw new IllegalArgumentException("Points must not lie on the same line — triangle is degenerate.");  
 }  
 this.a = a;  
 this.b = b;  
 this.c = c;  
 }  
  
 private boolean areColinear(Point p1, Point p2, Point p3) {  
 // Використовуємо площу паралелограма, щоб перевірити на колінеарність  
 double area = p1.x \* (p2.y - p3.y) +  
 p2.x \* (p3.y - p1.y) +  
 p3.x \* (p1.y - p2.y);  
 return area == 0;  
 }  
  
 public double area() {  
 return Math.abs(  
 a.x \* (b.y - c.y) +  
 b.x \* (c.y - a.y) +  
 c.x \* (a.y - b.y)  
 ) / 2.0;  
 }  
  
 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);  
 }  
}  
  
public class task3 {public static void main(String[] args) {  
 Triangle t = new Triangle(  
 new Point(0, 0),  
 new Point(4, 0),  
 new Point(2, 3)  
 );  
  
 System.out.println("Area: " + t.area()); // 6.0  
 System.out.println("Centroid: " + t.centroid()); // (2.0;1.0)  
}  
}

Цей код описує клас Triangle, який представляє трикутник на площині за трьома точками a, b, c.

**Конструктор** перевіряє, що точки **не лежать на одній прямій** (тобто не вироджений трикутник).

**Метод area()** обчислює площу трикутника за координатами (формула площі через детермінант).

**Метод centroid()** обчислює координати центроїда (середини мас) — це середнє арифметичне трьох вершин.

main():

* Створюється трикутник з вершинами (0,0), (4,0), (2,3).
* Виводиться площа: 6.0.
* Виводиться центроїд: (2.0;1.0).