**Лабораторна робота №4**

**Абстрактні класи та інтерфейси**

**Мета:** Використовуючи теоретичне підґрунтя абстракті класи та інтерфейси у мові Java, виконати дії що будуть вказано в завданні до лабораторної роботи.

**Хід роботи**

**Посилання на GitHub:** [OOP-KB-222-Ann-Dryzhak/lab\_04 at main · aanutadrzhk/OOP-KB-222-Ann-Dryzhak · GitHub](https://github.com/aanutadrzhk/OOP-KB-222-Ann-Dryzhak/tree/main/lab_04)

1. Клас Point:

package lab\_04;

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:

* Створюмємо поля типу double x та y, що визначають горизонтальну та вертикальну координати точки.
* Створюємо конструктор Point, який приймає два параметри типу double: x та y.
* Перевизначений метод toString(), який призначений для створення рядкового представлення об'єкта типу Point.

1. Клас Figure:

package lab\_04;

abstract class Figure {

abstract double area();

abstract Point centroid();

}

Пояснення:

* Створюємо абстрактний клас Figure.
* Створюємо абстрактний метод abstract double area(), який призначений для обчислення площі фігури.
* Створюємо абстрактний метод abstract Point centroid(), який повертає центроїд фігури.

1. Клас Triangle:

package lab\_04;

class Triangle extends Figure {

private Point a;

private Point b;

private Point c;

Triangle(Point a, Point b, Point c) {

if (!isValidTriangle(a, b, c)) {

throw new IllegalArgumentException("Invalid triangle vertices");

}

this.a = a;

this.b = b;

this.c = c;

}

private boolean isValidTriangle(Point a, Point b, Point c) {

double ab = distance(a, b);

double bc = distance(b, c);

double ac = distance(a, c);

return (ab + bc > ac) && (ab + ac > bc) && (ac + bc > ab);

}

private double distance(Point p1, Point p2) {

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

}

*@Override*

double area() {

return Math.*abs*((a.x \* (b.y - c.y) + b.x \* (c.y - a.y) + c.x \* (a.y - b.y)) / 2);

}

*@Override*

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);

}

*@Override*

public String toString() {

return "Triangle[A" + a + " B" + b + " C" + c + "]";

}

}

Пояснення:

* Створюємо клас Triangle, який реалізує абстрактний клас Figure.
* В класі створюємо поля private Point a, private Point b, private Point c, вони представляють вершини трикутника.
* Створюємо конструктор Triangle(Point a, Point b, Point c), який ініціалізує вершини трикутника.
* Далі за допомогою методу isValidTriangle, перевіряємо, чи є задані вершини валідними для утворення трикутника. Якщо вершини не утворюють трикутник, генерується виняток IllegalArgumentException.
* Після перевірки трикутника, поля a, b і c ініціалізуються заданими вершинами.
* В Методі isValidTriangle спочатку обчислюємо відстань між вершинами трикутника, для цього використовуємо метод distance(Point p1, Point p2), який обчислює відстань між двома точками. І далі рядок return (ab + bc > ac) && (ab + ac > bc) && (ac + bc > ab) повертає значення true або false в залежності від того, чи можна побудувати трикутник з вказаними вершинами (сума будь-яких двох сторін трикутника завжди більша за третю сторону).
* Метод area() перевизначає абстрактний метод класу Figure. Він обчислює площу трикутника за допомогою формули . Модуль береться, щоб було позитивне значення площі.
* Метод centroid() перевизначає абстрактний метод класу Figure. Він обчислює центроїд трикутника, який є середнім значенням координат вершин трикутника.
* І далі перевизначаємо метод toString(), який повертає задане в завданні рядкове представлення трикутника.

1. Клас Quadrilateral:

package lab\_04;

import java.util.ArrayList;

class Quadrilateral extends Figure {

private Point a;

private Point b;

private Point c;

private Point d;

Quadrilateral(Point a, Point b, Point c, Point d) {

if (!isValidQuadrilateral(a, b, c, d)) {

throw new IllegalArgumentException("Invalid quadrilateral vertices");

}

this.a = a;

this.b = b;

this.c = c;

this.d = d;

}

private boolean isValidQuadrilateral(Point a, Point b, Point c, Point d) {

ArrayList<Point> points = new ArrayList<>();

points.add(a);

points.add(b);

points.add(c);

points.add(d);

for (int i = 0; i < 4; i++) {

Point p1 = points.get(i);

Point p2 = points.get((i + 1) % 4);

Point p3 = points.get((i + 2) % 4);

if (isCollinear(p1, p2, p3)) {

return false;

}

}

return true;

}

private boolean isCollinear(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))) < 1e-9;

}

*@Override*

double area() {

double areaTriangle1 = triangleArea(a, b, c);

double areaTriangle2 = triangleArea(c, b, d);

return areaTriangle1 + areaTriangle2;

}

private double triangleArea(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);

}

*@Override*

Point centroid() {

double centroidX = (a.x + b.x + c.x + d.x) / 4;

double centroidY = (a.y + b.y + c.y + d.y) / 4;

return new Point(centroidX, centroidY);

}

*@Override*

public String toString() {

return "Quadrilateral[A" + a + " B" + b + " C" + c + " D" + d + "]";

}

}

Пояснення:

* Створяюємо поля класу Point a, b, c, d, вони представляють вершини чотирикутника.
* Конструктор Quadrilateral(Point a, Point b, Point c, Point d) - приймає ці чотири точки як параметри. І далі використовуємо метод isValidQuadrilateral, який перевіряє, чи задані вершини утворюють дійсний чотирикутник. Якщо вершини не утворюють чотирикутник, генерується виняток IllegalArgumentException.
* Після перевірки вершин, поля a, b, c і d ініціалізуються заданими вершинами.
* Приватний метод isValidQuadrilateral(Point a, Point b, Point c, Point d) перевіряє, чи можна побудувати чотирикутник з заданими вершинами. Для цього створюємо список точок points, до якого додаються всі вершини чотирикутника. Потім перевіряємо, чи є які-небудь три точки колінеарними. Якщо такі три точки знаходяться, це означає, що чотирикутник не може бути сформований, тому метод повертає false. В іншому випадку повертається true.
* Приватний метод isCollinear(Point a, Point b, Point c) - перевіряє, чи є три точки a, b і c колінеарними, тобто чи лежать вони на одній прямій. Перевірка виконується обчислюючи площу трикутника утвореного цими трьома точками, якщо площа трикутника дуже близька до нуля, то точки колінеарні, і метод повертає true.
* В методі area() обчислюємо площу чотирикутника, розбиваючи його на два трикутники та сумуючи їх площу.
* Площа кожного з трикутників обчислюється за допомогою приватного методу triangleArea.
* Та в методі centroid() обчислюємо центроїд чотирикутника, рахуючи середнє значення координат вершин чотирикутника.
* Перевизначаємо метод toString(), який повертає задане в завданні рядкове представлення чотирикутника.

1. Клас Circle:

package lab\_04;

class Circle extends Figure {

private Point center;

private double radius;

Circle(Point center, double radius) {

if (radius <= 0) {

throw new IllegalArgumentException("Invalid radius");

}

this.center = center;

this.radius = radius;

}

*@Override*

double area() {

return Math.***PI*** \* radius \* radius;

}

*@Override*

Point centroid() {

return center;

}

*@Override*

public String toString() {

return "Circle[" + center + " Radius " + radius + "]";

}

}

Пояснення:

* Створюємо поле private Point center, яке представляє центр кола і є об'єктом класу Point.
* Та поле private double radius, яке вказує радіус кола.
* Конструктор Circle(Point center, double radius) приймає координати центра кола і радіус кола як параметри. В цьому конструкторі перевіряємо чи є заданий радіус додатним числом. Якщо радіус менше або дорівнює нулю генерується виняток IllegalArgumentException. Після перевірки координати центра і радіус кола ініціалізуються відповідними полями класу.
* В методі area() перевизначаємо відповідний метод абстрактного класу Figure. Він обчислює площу кола за формулою πr².
* Далі в методі centroid() перевизначаємо відповідний метод абстрактного класу Figure. Цей метод просто повертає об'єкт center, який представляє координати центра кола.
* Перевизначаємо метод toString(), який повертає задане в завданні рядкове представлення чотирикутника.

1. Клас Main:

package lab\_04;

import java.util.Scanner;

public class Main {

public static void main(String[] args) {

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

System.***out***.println("Введіть тип фігури, яку ви хочите обсилити (Трикутник(1) / Чотирикутник(2) / Коло(3)):");

int choice = scanner.nextInt();

switch (choice) {

case 1:

System.***out***.println("Введіть координати трикутника (x1 y1 x2 y2 x3 y3):");

double x1 = scanner.nextDouble();

double y1 = scanner.nextDouble();

double x2 = scanner.nextDouble();

double y2 = scanner.nextDouble();

double x3 = scanner.nextDouble();

double y3 = scanner.nextDouble();

Point a = new Point(x1, y1);

Point b = new Point(x2, y2);

Point c = new Point(x3, y3);

Triangle triangle = new Triangle(a, b, c);

System.***out***.println("Трикутник: " + triangle);

System.***out***.println("Площа: " + triangle.area());

System.***out***.println("Центроїд: " + triangle.centroid());

break;

case 2:

System.***out***.println("Введіть координати чотирикутника (x1 y1 x2 y2 x3 y3 x4 y4):");

double x1q = scanner.nextDouble();

double y1q = scanner.nextDouble();

double x2q = scanner.nextDouble();

double y2q = scanner.nextDouble();

double x3q = scanner.nextDouble();

double y3q = scanner.nextDouble();

double x4q = scanner.nextDouble();

double y4q = scanner.nextDouble();

Point Aq = new Point(x1q, y1q);

Point Bq = new Point(x2q, y2q);

Point Cq = new Point(x3q, y3q);

Point Dq = new Point(x4q, y4q);

Quadrilateral quadrilateral = new Quadrilateral(Aq, Bq, Cq, Dq);

System.***out***.println("Чотирикутник: " + quadrilateral);

System.***out***.println("Площа: " + quadrilateral.area());

System.***out***.println("Центроїд: " + quadrilateral.centroid());

break;

case 3:

System.***out***.println("Введіть координати центра кола (x y) та радіус:");

double xc = scanner.nextDouble();

double yc = scanner.nextDouble();

double radius = scanner.nextDouble();

Point center = new Point(xc, yc);

Circle circle = new Circle(center, radius);

System.***out***.println("Коло: " + circle);

System.***out***.println("Площа: " + circle.area());

System.***out***.println("Центроїд: " + circle.centroid());

break;

default:

System.***out***.println("Invalid choice!");

}

scanner.close();

}

}

Пояснення:

* Створюємо об'єкт Scanner для зчитування введених даних з консолі.
* Користувачу пропонується обрати тип фігури.
* Використовуємо оператор switch для визначення типу фігури, яку користувач хоче обчислити.
* У кожному з трьох варіантів фігури користувачеві потрібно ввести відповідні дані для створення фігури. Зчитуємо координати точок або центра та радіуса для обраного варіанту.
* Створюємо відповідний об'єкт фігури з введеними даними.
* Обчислюємо площу та центроїд для фігури та виводимо на экран.

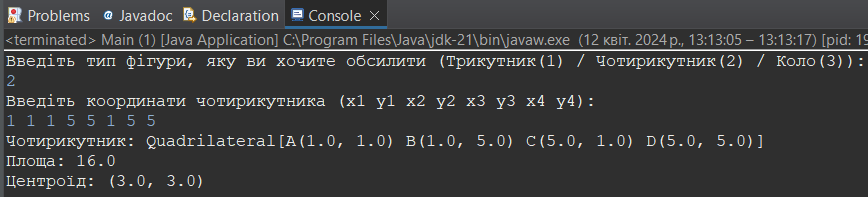


Рис. 1 - результат тестування для фігури “чотирикутник”

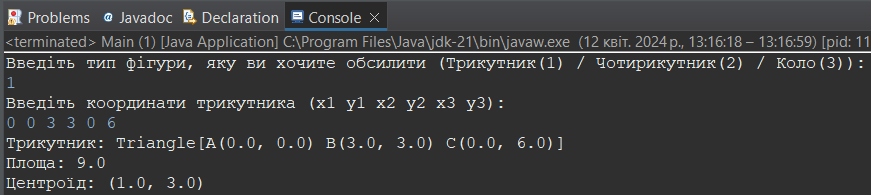


Рис. 2 - результат тестування для фігури “трикутник”

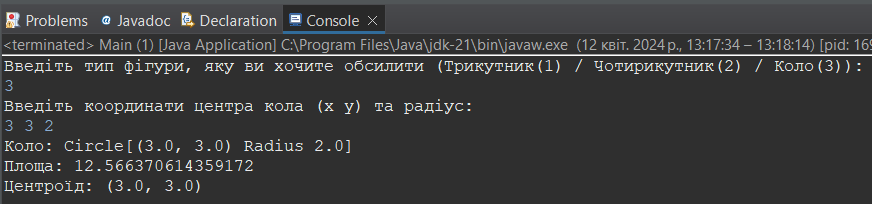


Рис. 3 - результат тестування для фігури “коло”

**Висновок.** Під час виконання даної лабораторної роботи я використовуючи теоретичне підґрунтя абстракті класи та інтерфейси у мові Java, виконала дії що було вказано в завданні до лабораторної роботи.