**1)** public class Cursor {

// Статичні поля для зберігання максимальних значень розмірів курсору

private static final int MAX\_SIZE = 15;

private static final int MIN\_SIZE = 1;

// Поля класу

private int x; // Координата по горизонталі

private int y; // Координата по вертикалі

private boolean isHorizontal; // Вигляд курсору (горизонтальний чи вертикальний)

private int size; // Розмір курсору

// Конструктор за замовчуванням

public Cursor() {

this(0, 0, true, 1); // Виклик іншого конструктора

}

// Конструктор з параметрами

public Cursor(int x, int y, boolean isHorizontal, int size) {

// Перевірка коректності введених параметрів

this.x = x;

this.y = y;

this.isHorizontal = isHorizontal;

this.size = Math.min(Math.max(size, MIN\_SIZE), MAX\_SIZE); // Обмеження розміру курсору

}

// Методи зміни координат курсору

public void moveCursor(int newX, int newY) {

this.x = newX;

this.y = newY;

}

// Метод зміни вигляду курсору

public void changeCursorView(boolean newIsHorizontal) {

this.isHorizontal = newIsHorizontal;

}

// Метод зміни розміру курсору

public void changeCursorSize(int newSize) {

this.size = Math.min(Math.max(newSize, MIN\_SIZE), MAX\_SIZE); // Обмеження розміру курсору

}

// Методи приховання та відновлення курсору

public void hideCursor() {

System.out.println("Cursor is hidden.");

}

public void restoreCursor() {

System.out.println("Cursor is restored.");

}

// Перевантажений метод toString

@Override

public String toString() {

return "Cursor{" +

"x=" + x +

", y=" + y +

", isHorizontal=" + isHorizontal +

", size=" + size +

'}';

}

// Перевантажений метод equals

@Override

public boolean equals(Object obj) {

if (this == obj) return true;

if (obj == null || getClass() != obj.getClass()) return false;

Cursor cursor = (Cursor) obj;

return x == cursor.x && y == cursor.y && isHorizontal == cursor.isHorizontal && size == cursor.size;

}

public static void main(String[] args) {

// Створення об'єкта класу Cursor за допомогою різних конструкторів

Cursor cursor1 = new Cursor();

Cursor cursor2 = new Cursor(5, 7, false, 10);

// Демонстрація роботи методів класу

System.out.println("Cursor 1: " + cursor1);

System.out.println("Cursor 2: " + cursor2);

cursor1.moveCursor(3, 4);

System.out.println("Cursor 1 after moving: " + cursor1);

cursor2.changeCursorView(true);

System.out.println("Cursor 2 after changing view: " + cursor2);

cursor1.changeCursorSize(8);

System.out.println("Cursor 1 after changing size: " + cursor1);

cursor2.hideCursor();

cursor2.restoreCursor();

}

}

**2)** public class Rectangle {

private double length;

private double width;

// Конструктор з параметрами

public Rectangle(double length, double width) {

this.length = length;

this.width = width;

}

// Метод для знаходження периметру прямокутника

public double calculatePerimeter() {

return 2 \* (length + width);

}

// Метод для знаходження площі прямокутника

public double calculateArea() {

return length \* width;

}

// Метод для порівняння з іншим прямокутником

public boolean equals(Rectangle other) {

return this.length == other.length && this.width == other.width;

}

// Метод для виявлення подібності до іншого прямокутника

public boolean isSimilar(Rectangle other) {

return (this.length / this.width) == (other.length / other.width);

}

// Перевантажений метод toString

@Override

public String toString() {

return "Rectangle{length=" + length + ", width=" + width + '}';

}

public static void main(String[] args) {

// Створення об'єктів класу Rectangle

Rectangle rectangle1 = new Rectangle(5, 8);

Rectangle rectangle2 = new Rectangle(7, 10);

// Виведення інформації про прямокутники

System.out.println("Rectangle 1: " + rectangle1);

System.out.println("Rectangle 2: " + rectangle2);

// Виклик методів класу для обчислення периметру та площі

System.out.println("Perimeter of Rectangle 1: " + rectangle1.calculatePerimeter());

System.out.println("Area of Rectangle 2: " + rectangle2.calculateArea());

// Виклик методу порівняння

System.out.println("Are rectangles equal? " + rectangle1.equals(rectangle2));

// Виклик методу виявлення подібності

System.out.println("Are rectangles similar? " + rectangle1.isSimilar(rectangle2));

}

}

**3)** import java.util.LinkedList;

import java.util.Queue;

public class IntegerQueue {

private Queue<Integer> queue;

// Конструктор за замовчуванням

public IntegerQueue() {

this.queue = new LinkedList<>();

}

// Метод для додавання елемента у чергу

public void enqueue(int element) {

queue.add(element);

}

// Метод для вилучення елемента з черги

public int dequeue() {

if (isEmpty()) {

throw new IllegalStateException("Queue is empty");

}

return queue.poll();

}

// Метод для перевірки, чи черга порожня

public boolean isEmpty() {

return queue.isEmpty();

}

// Метод для отримання розміру черги

public int size() {

return queue.size();

}

// Метод для пошуку елемента у черзі

public boolean contains(int element) {

return queue.contains(element);

}

public static void main(String[] args) {

IntegerQueue integerQueue = new IntegerQueue();

// Додавання елементів у чергу

integerQueue.enqueue(10);

integerQueue.enqueue(20);

integerQueue.enqueue(30);

// Виведення розміру черги

System.out.println("Queue size: " + integerQueue.size());

// Виведення та вилучення елементів з черги

while (!integerQueue.isEmpty()) {

int element = integerQueue.dequeue();

System.out.println("Dequeued element: " + element);

}

// Перевірка наявності елемента у черзі

System.out.println("Is 20 in the queue? " + integerQueue.contains(20));

}

}