**Implementation details:**

import java.util.Random;

class NumberGenerator implements Runnable {

    private Random random = new Random();

    @Override

    public void run() {

        try {

            while (true) {

                int randomNumber = random.nextInt(100); // Assuming a range, you can adjust it

                System.out.println("Generated Number: " + randomNumber);

                if (randomNumber % 2 == 0) {

                    // If the number is even, notify the SquareCalculator thread

                    synchronized (SquareCalculator.lock) {

                        SquareCalculator.number = randomNumber;

                        SquareCalculator.lock.notify();

                    }

                } else {

                    // If the number is odd, notify the CubePrinter thread

                    synchronized (CubePrinter.lock) {

                        CubePrinter.number = randomNumber;

                        CubePrinter.lock.notify();

                    }

                }

                Thread.sleep(1000); // Wait for 1 second

            }

        } catch (InterruptedException e) {

            e.printStackTrace();

        }

    }

}

class SquareCalculator implements Runnable {

    static final Object lock = new Object();

    static int number;

    @Override

    public void run() {

        try {

            while (true) {

                synchronized (lock) {

                    lock.wait();

                    int square = number \* number;

                    System.out.println("Square: " + square);

                }

            }

        } catch (InterruptedException e) {

            e.printStackTrace();

        }

    }

}

class CubePrinter implements Runnable {

    static final Object lock = new Object();

    static int number;

    @Override

    public void run() {

        try {

            while (true) {

                synchronized (lock) {

                    lock.wait();

                    int cube = number \* number \* number;

                    System.out.println("Cube: " + cube);

                }

            }

        } catch (InterruptedException e) {

            e.printStackTrace();

        }

    }

}

 class MultiThreadExample {

    public static void main(String[] args) {

        Thread generatorThread = new Thread(new NumberGenerator());

        Thread squareThread = new Thread(new SquareCalculator());

        Thread cubeThread = new Thread(new CubePrinter());

        generatorThread.start();

        squareThread.start();

        cubeThread.start();

    }

}

**Output:**

