1. Write java program to print Table of Five, Seven and Thirteen using Multithreading

     (Use Thread class for the implementation).

CODE:

class fiveTable extends Thread{

    public void run(){

        System.out.println("5 multiplication table:");

        for(int i=1;i<=10;i++){

            System.out.println("5 x " + i + " " + "= " + (5\*i));

        }

        System.out.println();

    }

}

class sevenTable extends Thread{

    public void run(){

        System.out.println("7 multiplication table:");

        for(int i=1;i<=10;i++){

            System.out.println("7 x " + i + " " + "= " + (7\*i));

        }

        System.out.println();

    }

}

class thirteenTable extends Thread{

    public void run(){

        System.out.println("13 multiplication table:");

        for(int i=1;i<=10;i++){

            System.out.println("13 x " + i + " " + "= " + (13\*i));

        }

        System.out.println();

    }

}

public class multiply {

    public static void main(String[] args) {

        Thread fivetable = new fiveTable();

        Thread seventable = new sevenTable();

        Thread thirteentable = new thirteenTable();

        fivetable.start();

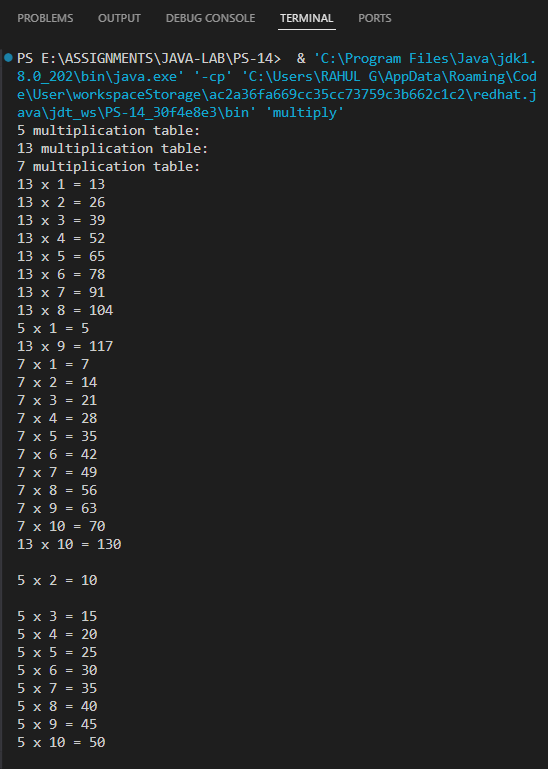
        seventable.start();

        thirteentable.start();

    }

}

Output:



2.  Write a Java program that creates two threads to find and print even and odd numbers

      from 1 to 20.

CODE:

class even extends Thread{

    public void run(){

        for(int j=0;j<=20;j = j+2){

            System.out.println(j);

        }

    }

}

class odd extends Thread{

    public void run(){

        for(int i=1;i<=20;i = i+2){

            System.out.println(i);

        }

    }

}

public class evenOdd {

    public static void main(String[] args) {

        Thread odd\_numbers = new odd();

        Thread even\_numbers = new even();

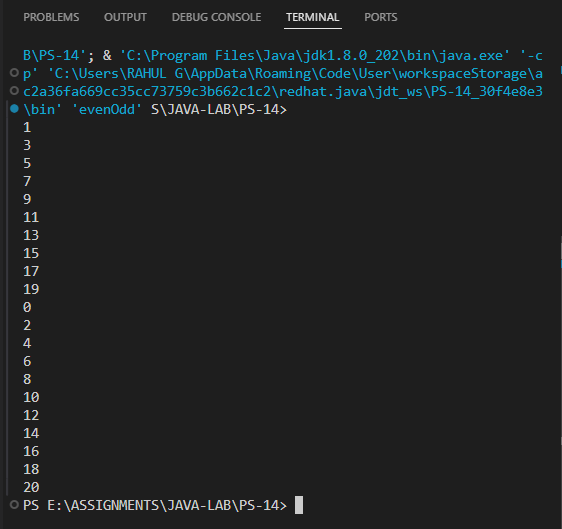
        even\_numbers.start();

        odd\_numbers.start();

    }

}

OUTPUT:



3. Write a Java program to create a thread by implementing a Runnable interface and set

     and print the thread name.

CODE:

class MyRunnable implements Runnable {

    @Override

    public void run() {

        Thread currentThread = Thread.currentThread();

        System.out.println("Thread Name: " + currentThread.getName());

    }

}

public class TheRunnable {

    public static void main(String[] args) {

        MyRunnable myRunnable = new MyRunnable();

        Thread thread = new Thread(myRunnable);

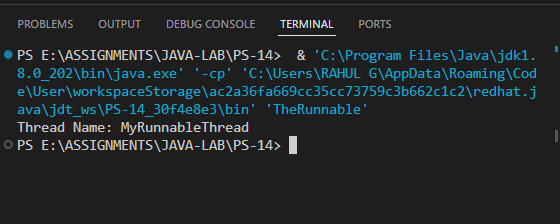
        thread.setName("MyRunnableThread");

        thread.start();

    }

}

OUTPUT:



4. Write a java program to print first 20 prime numbers and 15 Fibonacci numbers by

     creating two child threads and also print the total time taken by each thread for the

     execution.

class PrimeNumbersThread extends Thread {

    @Override

    public void run() {

        long startTime = System.currentTimeMillis();

        int count = 0, num = 2;

        System.out.println("First 20 prime numbers:");

        while (count < 20) {

            if (isPrime(num)) {

                System.out.println(num);

                count++;

            }

            num++;

        }

        long endTime = System.currentTimeMillis();

        System.out.println("Time taken by PrimeNumbersThread: " + (endTime - startTime) + " ms");

    }

    private boolean isPrime(int num) {

        if (num <= 1) return false;

        for (int i = 2; i <= Math.sqrt(num); i++) {

            if (num % i == 0) return false;

        }

        return true;

    }

}

class FibonacciThread extends Thread {

    @Override

    public void run() {

        long startTime = System.currentTimeMillis();

        System.out.println("First 15 Fibonacci numbers:");

        int n1 = 0, n2 = 1;

        System.out.println(n1);

        System.out.println(n2);

        for (int i = 2; i < 15; i++) {

            int n3 = n1 + n2;

            System.out.println(n3);

            n1 = n2;

            n2 = n3;

        }

        long endTime = System.currentTimeMillis();

        System.out.println("Time taken by FibonacciThread: " + (endTime - startTime) + " ms");

    }

}

public class threadsTimeTaken {

    public static void main(String[] args) {

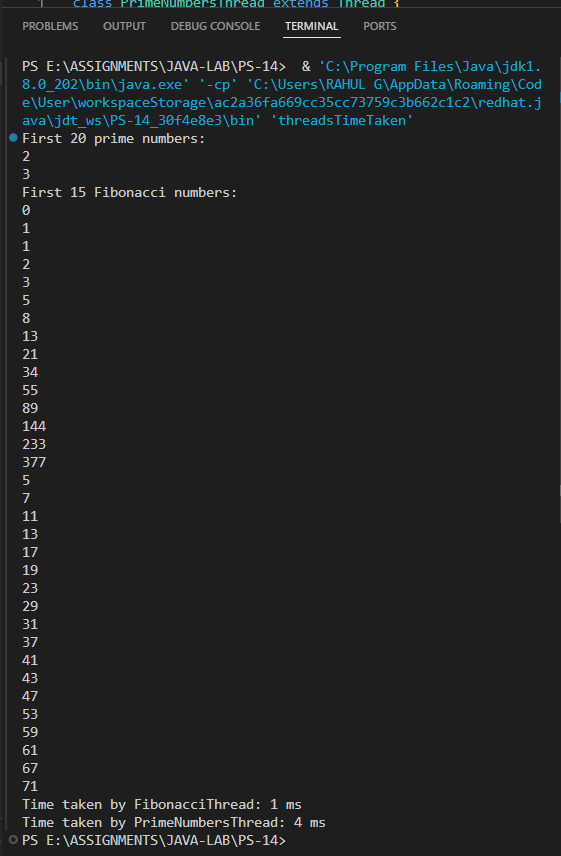
        new PrimeNumbersThread().start();

        new FibonacciThread().start();

    }

}

OUTPUT:



5. Write a java program that implements a multi-thread application that has three threads.

    First thread generates random integer every 1 second and if the value is even, second

    thread computes the square of the number and prints. If the value is odd, the third thread

    will print the value of cube of the number.

CODE:

import java.util.Random;

class SharedData {

    private int number;

    private boolean newNumberAvailable = false;

    public synchronized void setNumber(int number) {

        this.number = number;

        newNumberAvailable = true;

        notifyAll();

    }

    public synchronized int getNumber() {

        while (!newNumberAvailable) {

            try {

                wait();

            } catch (InterruptedException e) {

                Thread.currentThread().interrupt();

            }

        }

        newNumberAvailable = false;

        return number;

    }

}

class NumberGenerator extends Thread {

    private final SharedData sharedData;

    private final Random random = new Random();

    public NumberGenerator(SharedData sharedData) {

        this.sharedData = sharedData;

    }

    @Override

    public void run() {

        while (!Thread.currentThread().isInterrupted()) {

            int number = random.nextInt(100);

            System.out.println("Generated number: " + number);

            sharedData.setNumber(number);

            try {

                Thread.sleep(1000);

            } catch (InterruptedException e) {

                Thread.currentThread().interrupt();

            }

        }

    }

}

class SquareCalculator extends Thread {

    private final SharedData sharedData;

    public SquareCalculator(SharedData sharedData) {

        this.sharedData = sharedData;

    }

    @Override

    public void run() {

        while (!Thread.currentThread().isInterrupted()) {

            int number = sharedData.getNumber();

            if (number % 2 == 0) {

                System.out.println("Square of " + number + ": " + (number \* number));

            }

        }

    }

}

class CubeCalculator extends Thread {

    private final SharedData sharedData;

    public CubeCalculator(SharedData sharedData) {

        this.sharedData = sharedData;

    }

    @Override

    public void run() {

        while (!Thread.currentThread().isInterrupted()) {

            int number = sharedData.getNumber();

            if (number % 2 != 0) {

                System.out.println("Cube of " + number + ": " + (number \* number \* number));

            }

        }

    }

}

public class randomSquareCube {

    public static void main(String[] args) {

        SharedData sharedData = new SharedData();

        new NumberGenerator(sharedData).start();

        new SquareCalculator(sharedData).start();

        new CubeCalculator(sharedData).start();

    }

}

OUTPUT:

