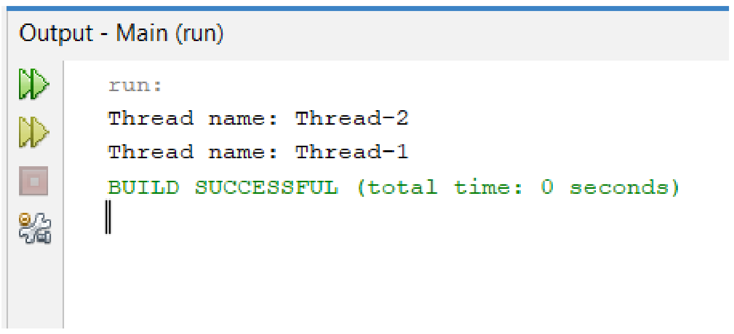
Day-9
Quiz-1

1. Write a Java program to perform a runnable interface, take two threads t1 and t2 and fetch the names of the thread using getName() method.

CODE:

```
class MyRunnable implements Runnable {
    public void run() {
        // Code to be executed by the thread
        System.out.println("Thread name: " + Thread.currentThread().getName());
    }
}

public class ThreadExample {
    public static void main(String[] args) {
        // Creating instances of the MyRunnable class
        MyRunnable myRunnable = new MyRunnable();

        // Creating threads and passing the instances of MyRunnable
        Thread t1 = new Thread(myRunnable, "Thread-1");
        Thread t2 = new Thread(myRunnable, "Thread-2");

        // Starting the threads
        t1.start();
        t2.start();
    }
}
```

OUTPUT:



```
run:
Thread name: Thread-2
Thread name: Thread-1
BUILD SUCCESSFUL (total time: 0 seconds)
```

2.Given an integer N, the task is to write program to print the first N natural numbers in increasing order using two threads.

Input: *N = 10*
Output: *1 2 3 4 5 6 7 8 9 10*

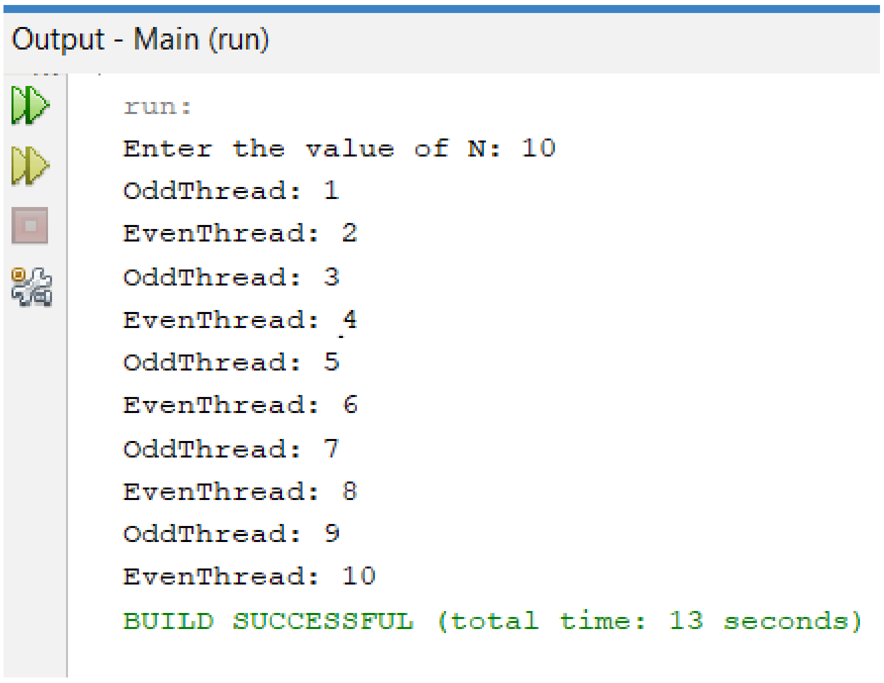
Input: *N = 18*
Output: *1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18*

CODE:

```
package main;



import java.util.Scanner;



class PrintNumbers implements Runnable {

    private static final Object lock = new Object();

    private static int currentNumber = 1;

    private int targetNumber;



    public PrintNumbers(int targetNumber) {

        this.targetNumber = targetNumber;

    }



    @Override

    public void run() {

        synchronized (lock) {

            while (currentNumber <= targetNumber) {

                if (currentNumber % 2 == 0 &&
Thread.currentThread().getName().equals("EvenThread")) {
```

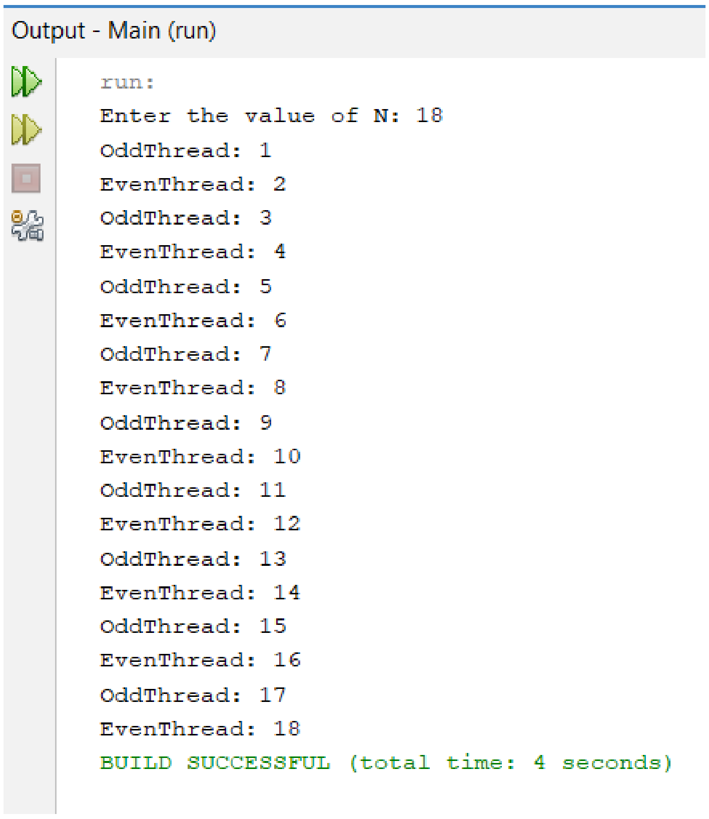
```
                    System.out.println(Thread.currentThread().getName() + ": " + currentNumber);

                    currentNumber++;

                } else if (currentNumber % 2 != 0 &&
Thread.currentThread().getName().equals("OddThread")) {

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

                    currentNumber++;

                } else {

                    try {

                        lock.wait();

                    } catch (InterruptedException e) {

                        e.printStackTrace();

                    }

                }


                lock.notifyAll();

            }

        }

    }

}


public class PrintNumbersExample {

    public static void main(String[] args) {

        Scanner scanner = new Scanner(System.in);

        System.out.print("Enter the value of N: ");

        int N = scanner.nextInt();
        PrintNumbers printNumbers = new PrintNumbers(N);


        Thread evenThread = new Thread(printNumbers, "EvenThread");

        Thread oddThread = new Thread(printNumbers, "OddThread");


        evenThread.start();

        oddThread.start();

    }

}
```

OUTPUT:



Output - Main (run)

```
run:
Enter the value of N: 10
OddThread: 1
EvenThread: 2
OddThread: 3
EvenThread: 4
OddThread: 5
EvenThread: 6
OddThread: 7
EvenThread: 8
OddThread: 9
EvenThread: 10
BUILD SUCCESSFUL (total time: 13 seconds)
```


Output - Main (run)

```
run:
Enter the value of N: 18
OddThread: 1
EvenThread: 2
OddThread: 3
EvenThread: 4
OddThread: 5
EvenThread: 6
OddThread: 7
EvenThread: 8
OddThread: 9
EvenThread: 10
OddThread: 11
EvenThread: 12
OddThread: 13
EvenThread: 14
OddThread: 15
EvenThread: 16
OddThread: 17
EvenThread: 18
BUILD SUCCESSFUL (total time: 4 seconds)
```

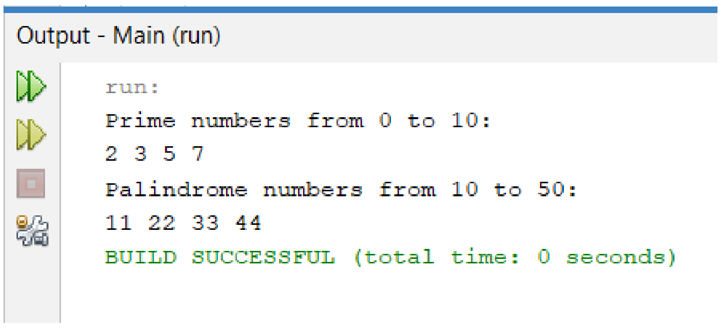
3. Write a two-threaded program, where one thread finds all prime numbers (in 0 to 10) and another thread finds all palindrome numbers (in 10 to 50). Schedule these threads in a sequential manner to get the results.

Palindrome numbers from 10 to 50 : 11 22 33 44
Prime numbers from 0 to 10 : 2 3 5 7

CODE:

```
class PrimeNumberThread extends Thread {
    @Override
    public void run() {
        System.out.println("Prime numbers from 0 to 10:");
        for (int i = 2; i <= 10; i++) {
            if (isPrime(i)) {
                System.out.print(i + " ");
            }
        }
        System.out.println();
    }

    private boolean isPrime(int num) {
        if (num < 2) {
            return false;
        }
        for (int i = 2; i <= Math.sqrt(num); i++) {
            if (num % i == 0) {
                return false;
            }
        }
        return true;
    }
}

class PalindromeNumberThread extends Thread {
    @Override
    public void run() {
        System.out.println("Palindrome numbers from 10 to 50:");
        for (int i = 10; i <= 50; i++) {
            if (isPalindrome(i)) {
                System.out.print(i + " ");
            }
        }
        System.out.println();
    }

    private boolean isPalindrome(int num) {
        int originalNum = num;
        int reversedNum = 0;

        while (num != 0) {
            int digit = num % 10;
            reversedNum = reversedNum * 10 + digit;
            num /= 10;
        }

        return originalNum == reversedNum;
    }
}

public class SequentialThreadExample {
    public static void main(String[] args) {
        PrimeNumberThread primeThread = new PrimeNumberThread();
        PalindromeNumberThread palindromeThread = new PalindromeNumberThread();

        // Schedule threads sequentially
        primeThread.start();
        try {
            primeThread.join(); // Wait for primeThread to finish before starting
palindromeThread
        } catch (InterruptedException e) {
            e.printStackTrace();
        }
        palindromeThread.start();
    }
}
```

OUTPUT:



```
run:
Prime numbers from 0 to 10:
2 3 5 7
Palindrome numbers from 10 to 50:
11 22 33 44
BUILD SUCCESSFUL (total time: 0 seconds)
```