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.

```
class MyRunnable implements Runnable {
    public void run() {

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

public class Main {
    public static void main(String[] args) {

        MyRunnable myRunnable = new MyRunnable();


        Thread t1 = new Thread(myRunnable);
        Thread t2 = new Thread(myRunnable);


        t1.setName("Thread-1");
        t2.setName("Thread-2");


        t1.start();
        t2.start();
    }
}
```

```
Thread name: Thread-1
Thread name: Thread-2
```

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

```
import java.util.Scanner;
class PrintNumbers implements Runnable {
    private int limit;
    private static Object lock = new Object();
    private static int number = 1;

    PrintNumbers(int limit) {
        this.limit = limit;
    }

    @Override
```

```
    public void run() {
        synchronized (lock) {
            while (number <= limit) {
                if ((Thread.currentThread().getName().equals("Odd") && (number % 2 != 0))
                        || (Thread.currentThread().getName().equals("Even") && (number % 2 == 0)))
                {
                    System.out.println(Thread.currentThread().getName() + ": " + number);
                    number++;
                    lock.notifyAll();
                } else {
                    try {
                        lock.wait();
                    } catch (InterruptedException e) {
                        e.printStackTrace();
                    }
                }
            }
        }
    }
}

public class Main{
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        System.out.print("Enter the value of N: ");
        int N = scanner.nextInt();

        Runnable oddRunnable = new PrintNumbers(N);
        Runnable evenRunnable = new PrintNumbers(N);

        Thread oddThread = new Thread(oddRunnable, "Odd");
        Thread evenThread = new Thread(evenRunnable, "Even");

        oddThread.start();
        evenThread.start();

        scanner.close();
    }
}
```

```
Enter the value of N: 10
Odd: 1
Even: 2
Odd: 3
Even: 4
Odd: 5
Even: 6
Odd: 7
Even: 8
Odd: 9
Even: 10
```

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

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

   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 PalindromeNumbersRunnable implements Runnable {
   @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 Main {
    public static void main(String[] args) {
        Thread primeThread = new Thread(new PrimeNumbersRunnable());
        Thread palindromeThread = new Thread(new PalindromeNumbersRunnable());

        primeThread.start();
        try {

            primeThread.join();
        } catch (InterruptedException e) {
            e.printStackTrace();
        }

        palindromeThread.start();
    }
}
```

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