```
1.class MyRunnable implements Runnable {
    public void run() {
        System.out.println("Thread name: " + Thread.currentThread().getName());
    }
}

public class ThreadExample {
    public static void main(String[] args) {
        MyRunnable myRunnable = new MyRunnable();
        Thread t1 = new Thread(myRunnable, "Thread-1");
        Thread t2 = new Thread(myRunnable, "Thread-2");
        t1.start();
        t2.start();
    }
}
2.import java.util.Scanner;

public class PrintNumbers {

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

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

        SharedPrinter printer = new SharedPrinter();

        Thread t1 = new Thread(new NumberPrinter(printer, 1, N / 2), "Thread-1");
        Thread t2 = new Thread(new NumberPrinter(printer, N / 2 + 1, N), "Thread-2");

        t1.start();
        t2.start();

        scanner.close();
    }
    static class SharedPrinter {
        private int number = 1;

        public void print(int num) {
            synchronized (this) {
                while (number <= num) {
                    while (number < num) {
                        try {
                            wait();
                        } catch (InterruptedException e) {
                            e.printStackTrace();
                        }
                    }
                    if (number <= num) {
                        System.out.println(Thread.currentThread().getName() + ": " + number);
                        number++;
                        notifyAll();
                    }
                }
            }
        }
    }
    static class NumberPrinter implements Runnable {
        private final SharedPrinter printer;
        private final int start;
        private final int end;

        public NumberPrinter(SharedPrinter printer, int start, int end) {
            this.printer = printer;
            this.start = start;
            this.end = end;
        }

        @Override
        public void run() {
            for (int i = start; i <= end; i++) {
                printer.print(i);
            }
        }
    }
}
```
3.
```
import java.util.Scanner;

public class NumberFinder {

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

        System.out.print("Enter the start of prime number range (e.g., 0): ");
        int primeStart = scanner.nextInt();

        System.out.print("Enter the end of prime number range (e.g., 10): ");
        int primeEnd = scanner.nextInt();

        System.out.print("Enter the start of palindrome number range (e.g., 10): ");
        int palindromeStart = scanner.nextInt();

        System.out.print("Enter the end of palindrome number range (e.g., 50): ");
        int palindromeEnd = scanner.nextInt();

        PrimeNumberFinder primeFinder = new PrimeNumberFinder(primeStart, primeEnd);
        PalindromeNumberFinder palindromeFinder = new
PalindromeNumberFinder(palindromeStart, palindromeEnd);

        Thread primeThread = new Thread(primeFinder);
        Thread palindromeThread = new Thread(palindromeFinder);

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

        palindromeThread.start();
        try {
            palindromeThread.join();
        } catch (InterruptedException e) {
            e.printStackTrace();
        }

        System.out.println("Prime numbers from " + primeStart + " to " + primeEnd + " : " +
primeFinder.getPrimeNumbers());
        System.out.println("Palindrome numbers from " + palindromeStart + " to " +
palindromeEnd + " : " + palindromeFinder.getPalindromeNumbers());

        scanner.close();
    }

    static class PrimeNumberFinder implements Runnable {
        private final StringBuilder primeNumbers = new StringBuilder();
        private final int start;
        private final int end;

        public PrimeNumberFinder(int start, int end) {
            this.start = start;
            this.end = end;
        }

        @Override
        public void run() {
            for (int i = start; i <= end; i++) {
                boolean isPrime = true;
                for (int j = 2; j <= Math.sqrt(i); j++) {
                    if (i % j == 0) {
                        isPrime = false;
                        break;
                    }
                }
                if (isPrime) {
                    primeNumbers.append(i).append(" ");
                }
            }
        }

        public String getPrimeNumbers() {
            return primeNumbers.toString().trim();
        }
    }

    static class PalindromeNumberFinder implements Runnable {
        private final StringBuilder palindromeNumbers = new StringBuilder();
        private final int start;
        private final int end;

        public PalindromeNumberFinder(int start, int end) {
            this.start = start;
            this.end = end;
        }

        @Override
        public void run() {
            for (int i = start; i <= end; i++) {
                if (isPalindrome(i)) {
                    palindromeNumbers.append(i).append(" ");
                }
            }
        }

        private boolean isPalindrome(int num) {
            int originalNum = num;
            int reverseNum = 0;
            while (num != 0) {
                int remainder = num % 10;
                reverseNum = reverseNum * 10 + remainder;
                num /= 10;
            }
            return originalNum == reverseNum;
        }

        public String getPalindromeNumbers() {
            return palindromeNumbers.toString().trim();
        }
    }
}
```