1,
```
class MyRunnable implements Runnable {
    @Override
    public void run() {
        // Fetching the name of the current thread using getName() method
        String threadName = Thread.currentThread().getName();
        System.out.println("Thread name: " + threadName);
    }
}

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

        // Creating threads and associating them with the MyRunnable instances
        Thread t1 = new Thread(myRunnable, "Thread 1");
        Thread t2 = new Thread(myRunnable, "Thread 2");

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

2,
```
class PrintNumbers implements Runnable {
    private static final Object lock = new Object();
    private static int number = 1;
    private int max;
    private boolean isEvenThread;
```

```
    public PrintNumbers(int max, boolean isEvenThread) {
        this.max = max;
        this.isEvenThread = isEvenThread;
    }

    @Override
    public void run() {
        while (number <= max) {
            synchronized (lock) {
                // Check if the current thread should print the number
                if ((isEvenThread && number % 2 == 0) || (!isEvenThread && number % 2 != 0)) {
                    System.out.println(Thread.currentThread().getName() + ": " + number);
                    number++;
                    lock.notifyAll();
                } else {
                    try {
                        // Wait for the other thread to print its number
                        lock.wait();
                    } catch (InterruptedException e) {
                        e.printStackTrace();
                    }
                }
            }
        }
    }
}

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

```
        int N = 10;

        Runnable printEven = new PrintNumbers(N, true);
        Runnable printOdd = new PrintNumbers(N, false);

        Thread evenThread = new Thread(printEven, "EvenThread");
        Thread oddThread = new Thread(printOdd, "OddThread");

        evenThread.start();
        oddThread.start();
    }
}
3, class PrimeNumbers implements Runnable {
    private int start;
    private int end;

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

    @Override
    public void run() {
        System.out.print("Prime numbers from " + start + " to " + end + " : ");
        for (int i = start; i <= end; 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 PalindromeNumbers implements Runnable {
    private int start;
    private int end;

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

    @Override
    public void run() {
        System.out.print("Palindrome numbers from " + start + " to " + end + " : ");
        for (int i = start; i <= end; i++) {
```

```
            if (isPalindrome(i)) {
                System.out.print(i + " ");
            }
        }
        System.out.println();
    }

    private boolean isPalindrome(int num) {
        int original = num;
        int reversed = 0;

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

        return original == reversed;
    }
}

public class Main {
    public static void main(String[] args) {
        Thread primeThread = new Thread(new PrimeNumbers(0, 10));
        Thread palindromeThread = new Thread(new PalindromeNumbers(10, 50));

        primeThread.start();
        try {
            primeThread.join(); // Wait for the primeThread to finish
```

```
        } catch (InterruptedException e) {
            e.printStackTrace();
        }

        palindromeThread.start();
    }
}
```