Java Threads:

1st method: extend Thread class in java.lang.

Ex. // SimpleThread.java

```java
    class SimpleThread extends Thread {
        private int id;
        private int delay;

        SimpleThread(int id, int delay) {
            this.id = id;
            this.delay = delay;
        }
    }
```
public void run() {
    System.out.println("id + " + "started");
    System.out.flush();
    for (int i = 0; i < 10; i++) {
        try {
            sleep(delay);
        } catch (InterruptedException e) {
            System.out.println("interrupted: " + e);
            System.out.println("id + " + i);
        }
        System.out.println("id + " + "finished")
    }
}
// TwoThreads.java

class TwoThreads {
    public static void main(String[] args) {
        SimpleThread t1 = new SimpleThread(1, 1000);
        SimpleThread t2 = new SimpleThread(2, 1300);
        t1.start();
        t2.start();
    }
}

Can also implement the Runnable interface.
// SimpleThread2.java
class SimpleThread2 implements Runnable {
    private int id;
    private int delay;

    SimpleThread2(int id, int delay) {
        this.id = id;
        this.delay = delay;
    }

    public void run() {
        System.out.println(id + " started");
        System.out.flush();
        for(int i = 0; i < 10; i++) {
            try {
                Thread.currentThread().sleep(delay);
            } catch (Exception e) {
                System.out.println(id + " finished");
            }
            System.out.println(id + " : " + i);
        }
    }
}
// TwoThreads.java

class TwoThreads2 {

    // main

    Thread t1 = new Thread(new SimpleThread2(1, 1000));

    Thread t2 = new Thread(1, 1300));

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

}
Three common ways:

- shared Objects
- messages (Sockets... later)
- remote method invocation (RMI)
class Counter {
    private int count;
    private int limit;
    Counter(int e, int m) {
        count = e;
        limit = m;
    }

    void click() {
        count = (count + 1) % limit;
    }

    int get() {
        return count;
    }
}
// Racer.java

class Racer implements Runnable {
    private int id;
    private Counter counter;

    Racer(int id, Counter counter) {
        this.id = id;
        this.counter = counter;
    }

    public void run() {
        System.out.println(id + " start");
        for (int i = 0; i < 1000000; i++) {
            counter.click();
        }
        System.out.println(id + " finish": + counter.get(1));
    }
}
// TwoRacers.java

class TwoRacers {

    public static void main(String[] args) {
        Counter c = new Counter(0, 1000000);

        Thread t1 = new Thread(new Racer(1, c));

        Thread t2 = new Thread(new Racer(2, c));

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

    // Counter class

    public class Counter{
        private int count;

        public Counter(int start, int limit) {
            this.count = start;
        }

        public synchronized void increment() {
            count++;
        }

        public synchronized void decrement() {
            count--;
        }

        public synchronized int getCount() {
            return count;
        }
    }

    // Racer class

    public class Racer implements Runnable{
        private int racerId;
        private Counter counter;

        public Racer(int racerId, Counter counter) {
            this.racerId = racerId;
            this.counter = counter;
        }

        public void run() {
            int start = 1;
            int end = 1000000;

            while (counter.getCount() < end) {
                synchronized (counter) {
                    counter.increment();
                    System.out.println("Racer "+racerId+" has passed "+
                        "counter: "+counter.getCount());
                }
            }
        }
    }
}