//Program on CyclicBarrier

```
import java.util.concurrent.BrokenBarrierException;
import java.util.concurrent.CyclicBarrier;

public class CyclicBarrierDemo {
        public static void main(String args[]) {
                // creating CyclicBarrier (checkPoint) with
     // 4 parties (Bikers) threads that need to call await()
     final CyclicBarrier checkPoint
       = new CyclicBarrier(4, new Runnable(){
        public void run(){
          //This task will be executed once all biker threads will reach barrier
          System.out.println("\nAll bikers have arrived to checkpoint. Lets refill the
petrol\n");
        }
     });

     //starting each of thread
     Thread biker1 = new Thread(new Biker(checkPoint), "Biker Thread 1");
     Thread biker2 = new Thread(new Biker(checkPoint), "Biker Thread 2");
     Thread biker3 = new Thread(new Biker(checkPoint), "Biker Thread 3");
     Thread biker4 = new Thread(new Biker(checkPoint), "Biker Thread 4");

     biker1.start();
     biker2.start();
     biker3.start();
     biker4.start();
   }
}

class Biker implements Runnable {
        private CyclicBarrier checkPoint;
   public Biker(CyclicBarrier checkPoint) {
     this.checkPoint = checkPoint;
   }

   // Code to be executed by each biker
   public void run() {
     try
     {
        System.out.println(Thread.currentThread().getName() + " has left Manali");

        checkPoint.await();
        System.out.println(Thread.currentThread().getName() + " has left the first
checkpoint / barrier");
```

```
            checkPoint.await();
            System.out.println(Thread.currentThread().getName() + " has left the second
checkpoint / barrier");

            checkPoint.await();
            System.out.println(Thread.currentThread().getName() + " has reached Leh");

        }
        catch (InterruptedException ex) {
            ex.printStackTrace();
        }
        catch (BrokenBarrierException b){
            b.printStackTrace();
        }
    }
}
```

//Program on Exchanger

```
import java.util.concurrent.Exchanger;
public class exchangerEx implements Runnable{

    Exchanger exchanger = null;
    Object    object    = null;

    public exchangerEx(Exchanger exchanger, Object object) {
        this.exchanger = exchanger;
        this.object = object;
    }

    public void run() {
        try {
            Object previous = this.object;

            this.object = this.exchanger.exchange(this.object);

            System.out.println(
                    Thread.currentThread().getName() +
                    " exchanged " + previous + " for " + this.object
            );
        } catch (InterruptedException e) {
            e.printStackTrace();
        }
    }
```

}