```java
import java.util.Iterator;
import java.util.Random;
import java.util.concurrent.LinkedBlockingQueue;

/**
 *
 * @author Ayisha S.R. Sowkathali, Sifaben Vahora
 *
 *         QueueSystem class runs the main
 *
 */
public class QueueSystem {

    /*
     * Main method instantiates 3 queues and 3 checkers. Passengers are added at
     * random.
     */
    public static void main(String[] args) throws InterruptedException {

        int queueASize = 30;
        int queueBSize = 20;
        int totalSize = queueASize + queueBSize;

        // Create 2 queues and 2 checkers
        LinkedBlockingQueue<Passenger> passengerQA = new LinkedBlockingQueue<Passenger>();
        LinkedBlockingQueue<Passenger> passengerQB = new LinkedBlockingQueue<Passenger>();

        Checker chkA = new Checker("Checker A");
        Checker chkB = new Checker("Checker B");

        // insert passengers into queue
        for (int i = 1; i <= totalSize; i++) {
            // generate a random number
            Random rn = new Random();
            int random = rn.nextInt(2) + 1;

            // Using mod to decide which queue the passengers go into
            if ((random % 2 == 0) && (passengerQA.size() <= queueASize)) {
                Passenger passenger = new Passenger("Passenger " + i,
System.currentTimeMillis());
                System.out.println("Adding " + passenger.getPassengerId() + " to queue
A");
                passengerQA.add(passenger);
            } else {
                if ((random % 2 == 1) && (passengerQB.size() <= queueBSize)) {
                    Passenger passenger = new Passenger("Passenger " + i,
System.currentTimeMillis());
                    System.out.println("Adding " + passenger.getPassengerId() + " to queue
B");
                    passengerQB.add(passenger);
                }
            }
        }
```

```
50
51          // Create Queue C and checker C
52          LinkedBlockingQueue<Passenger> passengerQC = new LinkedBlockingQueue<Passenger>();
53          Checker chkC = new Checker("Checker C");
54
55          // Create 2 threads, one for each queue
56          Runnable runnable1 = new Runnable() {
57              @Override
58              public void run() {
59                  Iterator<Passenger> iteratorA = passengerQA.iterator();
60                  while (iteratorA.hasNext()) {
61                      Passenger passenger = iteratorA.next();
62                      if (chkA.processPassenger(passenger, passengerQA)) {
63                          passengerQC.add(passenger);
64                          System.out.println("Queue C size: " + passengerQC.size());
65                      }
66                  }
67              }
68          };
69          Thread t1 = new Thread(runnable1);
70          t1.start();
71
72          Runnable runnable2 = new Runnable() {
73              @Override
74              public void run() {
75                  Iterator<Passenger> iteratorB = passengerQB.iterator();
76
77                  while (iteratorB.hasNext()) {
78                      Passenger passenger = iteratorB.next();
79                      if (chkB.processPassenger(passenger, passengerQB)) {
80                          passengerQC.add(passenger);
81                          System.out.println("Queue C size: " + passengerQC.size());
82                      }
83                  }
84              }
85          };
86          Thread t2 = new Thread(runnable2);
87          t2.start();
88
89          Runnable runnable3 = new Runnable() {
90              @Override
91              public void run() {
92                  Iterator<Passenger> iteratorC = passengerQC.iterator();
93
94                  while (iteratorC.hasNext()) {
95                      Passenger passenger = iteratorC.next();
96                      if (chkC.processPassenger(passenger, passengerQC)) {
97                          System.out.println("Queue C processed : " +
    passenger.getPassengerId());
98                          passengerQC.remove(passenger);
99                          System.out.println("Queue C removed : " +
    passenger.getPassengerId());
100                     }
```

```
                }
            }
        };
        Thread t3 = new Thread(runnable3);
        t3.start();
    }
}

```