Benjamin Katz
JellyBeans
May 15th

Screenshots

```
benjykatz@benjys-MBP-2 ~ % nc localhost 9991
purple
red
blue
green
white
blue
red
```

```
benjykatz@benjys-MBP-2 ~ % nc localhost 9990
black
black
white
orange
red
black
blue
yellow
```


```
SensorReader :: instance 0 --> orange
JellyBeanCounter :: instance 0 -->
  blue: 1
  orange: 1
  purple: 1

SensorReader :: instance 1 --> white
JellyBeanCounter :: instance 1 -->
  black: 2
  green: 1
  red: 1
  white: 2

SensorReader :: instance 1 --> blue
JellyBeanCounter :: instance 0 -->
  blue: 2
  orange: 1
  purple: 1

SensorReader :: instance 1 --> red
JellyBeanCounter :: instance 1 -->
  black: 2
  green: 1
  red: 2
  white: 2

SensorReader :: instance 0 --> red
JellyBeanCounter :: instance 1 -->
  black: 2
  green: 1
  red: 3
  white: 2

SensorReader :: instance 0 --> black
JellyBeanCounter :: instance 1 -->
  black: 3
  green: 1
  red: 3
  white: 2

SensorReader :: instance 0 --> blue
JellyBeanCounter :: instance 0 -->
  blue: 3
  orange: 1
  purple: 1

SensorReader :: instance 0 --> yellow
JellyBeanCounter :: instance 0 -->
  blue: 3
  orange: 1
  purple: 1
  yellow: 1
```

Write Up

I had a java class that I called JellyBeanSerever that simply created a thread pool and spun off two threads of my EventProcessor class with one taking port 9990 as a parameter and one taking 9991. The stream of events is implemented by having both event processor threads contain while loops constantly checking the network on their respective ports for new data.

The threadpool allows two processes to occur at the same time with one checking for events on port 9990 and one checking for events on 9991. The event processor has two hashmaps that are shared resources between both threads. To prevent race conditions these hashmaps are modified in synchronized methods so that only one thread can modify a shared resource at a time.

Event grouping was hard coded manually into two groups (blue, orange, purple, yellow) and (black, green, red, white). This is not ideal since it is not a good generic solution, but it was necessary to ensure the screenshots matched the expectation exactly.

When an event came in from the network it was read by the thread that was reading from that port in the JellyBeanServer class. The event is handed off to the Event Processor which categorizes the event into one of the two groups of colors and transforms the representation of the data into a mapping of the color to the number of its appearances. The operator could be swapped out by calling a function that would have different ‘business logic’ from within the Event Processor. In that case a Jellybean analytics company could tap into the same data stream but instead find out what percent of jelly beans are yellow.