1 FlatBufferSafeletExecuter

```
1  package scjlevel2examples.flatbuffer;
2
3  import javax.safetycritical.Safelet;
4  import javax.safetycritical.SafeletExecuter;
5
6  //Application entry point, runs the Safelet
7  public class FlatBufferSafeletExecuter extends SafeletExecuter
8  {
9
10   public FlatBufferSafeletExecuter(Safelet arg0)
11   {
12     super(arg0);
13   }
14
15   public static void main (String [] args)
16   {
17     System.out.println("FlatBufferSafeletExecuter ");
18     //Run the safelet which starts the whole application
19     FlatBufferSafeletExecuter.run(new FlatBuffer());
20   }
21 }
```

2 FlatBuffer

```
1  package scjlevel2examples.flatbuffer;
2
3  import javax.realtime.PriorityParameters;
4  import javax.safetycritical.*;
5  import javax.safetycritical.annotate.Level;
6
7  public class FlatBuffer implements Safelet
8  {
9
10       public Level getLevel()
11   {
12     return Level.LEVEL_2;
13   }
14
15   public MissionSequencer getSequencer()
16   {
17     System.out.println("FlatBuffer");
18     //Create and return the main mission sequencer
19     return new FlatBufferMissionSequencer(new PriorityParameters(5),
20         new StorageConfigurationParameters(1048576, 1048576, 1048576));
21   }
22
23   public void setup()
24   {
25   }
26
27   public void teardown()
28   {
29   }
30 }
```

3 FlatBufferMissionSequencer

```
1  package scjlevel2examples.flatbuffer;
2
3  import javax.realtime.PriorityParameters;
4  import javax.safetycritical.Mission;
5  import javax.safetycritical.MissionSequencer;
6  import javax.safetycritical.StorageConfigurationParameters;
7
8
9  public class FlatBufferMissionSequencer extends MissionSequencer
10 {
11   private boolean returnedMission;
12
13   public FlatBufferMissionSequencer(PriorityParameters priorityParameters,
14       StorageConfigurationParameters storageConfigurationParameters)
15   {
16     super(priorityParameters, storageConfigurationParameters);
17     returnedMission =false;
18   }
19
20
21   protected Mission getNextMission()
22   {
23     System.out.println("FlatBufferMissionSequencer");
24
25     //As this sequencer only delivers one mission,
26     //if it has not been returned yet then return it,
27     //else return null which will terminate the sequencer
28
29     if(!returnedMission)
30     {
31       System.out.println("FlatBufferMissionSequencer returns mission");
32       returnedMission = true;
33       return new FlatBufferMission();
34     }
35     else
36     {
37       return null;
38     }
39   }
40
41
42
43 }
```

4 FlatBufferMission

```
1  package scjlevel2examples.flatbuffer;
2
3  import javax.realtime.PeriodicParameters;
4  import javax.realtime.PriorityParameters;
5  import javax.realtime.RelativeTime;
6  import javax.safetycritical.Mission;
7  import javax.safetycritical.PriorityScheduler;
8  import javax.safetycritical.StorageConfigurationParameters;
9
10
11 public class FlatBufferMission extends Mission
12 {
13   private volatile int[] buffer;
14   private Writer writer;
15   private Reader reader;
16   private boolean terminatingPending;
17
18
19   public FlatBufferMission()
20   {
21     buffer = new int[1];
22       buffer[0]= 0;
23       terminatingPending = false;
24
25     System.out.println("FlatBufferMission");
26   }
27
28
29   protected void initialize()
30   {
31     reader = new Reader(new PriorityParameters(5), new
           StorageConfigurationParameters(1048576, 1048576, 1048576), this, writer);
32
33     writer = new Writer(new PriorityParameters(5) , new
           StorageConfigurationParameters(1048576, 1048576, 1048576), this,  reader);
34
35       System.out.println("FlatBufferMission init");
36   }
37
38   public boolean bufferEmpty()
39   {
40     return buffer[0] == 0;
41   }
42
43   public synchronized void write(int update)
44   {
45     buffer[0] = update;
46   }
47
48   public synchronized int read()
49   {
50     int out = buffer[0];
51     this.buffer[0] = 0;
52
53     return out;
54   }
55
56   public boolean terminationPending()
57   {
58     return terminatingPending;
59   }
60
61   public long missionMemorySize()
62   {
63     return 1048576;
64   }
65 }
```

5 Writer

```
1  package scjlevel2examples.flatbuffer;
2
3  import javax.realtime.PeriodicParameters;
4  import javax.realtime.PriorityParameters;
5  import javax.safetycritical.ManagedThread;
6  import javax.safetycritical.StorageConfigurationParameters;
7
8  import java.io.*;
9
10
11 public class Writer extends ManagedThread
12 {
13   private final FlatBufferMission fbMission;
14   private final Reader reader;
15   private int i;
16
17   public Writer(PriorityParameters priority, StorageConfigurationParameters storage
         , FlatBufferMission fbMission, Reader reader)
18   {
19     super(priority,storage);
20
21     this.fbMission = fbMission;
22     this.reader = reader;
23   }
24
25   public synchronized void notifyWriter()
26   {
27     notify();
28   }
29
30
31   public synchronized void run()
32   {
33     System.out.println("Writer!");
34
35     while(! fbMission.terminationPending())
36     {
37       try
38       {
39
40         while(!fbMission.bufferEmpty())
41         {
42           wait();
43         }
44
45
46         fbMission.write(i);
47         i ++;
48
49         reader.notifyReader();
50       }
51       catch(InterruptedException ie)
52       {
53
54       }
55     }
56
57   }
58 }
```

6 Reader

```
1  package scjlevel2examples.flatbuffer;
2
3  import javax.realtime.PeriodicParameters;
4  import javax.realtime.PriorityParameters;
5  import javax.safetycritical.ManagedThread;
6  import javax.safetycritical.StorageConfigurationParameters;
7
8
9
10 public class Reader extends ManagedThread
11 {
12   private final Writer writer;
13   private final FlatBufferMission fbMission;
14
15   public Reader(PriorityParameters priority, StorageConfigurationParameters storage
         , FlatBufferMission fbMission, Writer writer)
16   {
17     super(priority,  storage);
18
19     this.fbMission = fbMission;
20     this.writer = writer;
21   }
22
23   public synchronized void notifyReader()
24   {
25     notify();
26   }
27
28
29   public synchronized void run()
30   {
31     System.out.println("Reader!");
32
33     while(! fbMission.terminationPending())
34     {
35       try
36       {
37         while(fbMission.bufferEmpty())
38         {
39           wait();
40         }
41
42         System.out.println("I Read: " + fbMission.read());
43
44         writer.notify();
45       }
46       catch(InterruptedException ie)
47       {
48
49       }
50
51     }
52   }
53 }
```