1 FlatBufferSafeletLauncher

```
1  package scjlevel2examples.flatbuffer;
2
3  import javax.safetycritical.Launcher;
4  import javax.safetycritical.Mission;
5  import javax.safetycritical.Safelet;
6
7  //Application entry point, runs the Safelet
8  public class FlatBufferSafeletLauncher extends Launcher
9  {
10
11   public FlatBufferSafeletLauncher(Safelet<Mission> safelet)
12   {
13     super(safelet, 2);
14   }
15
16   public static void main(String[] args)
17   {
18     System.out.println("FlatBufferSafeletExecuter ");
19     // Run the safelet which starts the whole application
20     new FlatBufferSafeletLauncher(new FlatBuffer()).run();
21   }
22 }
```

2 FlatBuffer

```
1  package scjlevel2examples.flatbuffer;
2
3  import javax.realtime.PriorityParameters;
4  import javax.safetycritical.*;
5  import javax.safetycritical.annotate.Level;
6  import javax.scj.util.Const;
7
8  public class FlatBuffer implements Safelet<Mission>
9  {
10
11       public Level getLevel()
12   {
13     return Level.LEVEL_2;
14   }
15
16   public MissionSequencer<Mission> getSequencer()
17   {
18     System.out.println("FlatBuffer");
19     //Create and return the main mission sequencer
20     StorageParameters storageParameters = new StorageParameters(150 * 1000, new
           long[] { Const.HANDLER_STACK_SIZE },
21           Const.PRIVATE_MEM_SIZE-25*1000, Const.IMMORTAL_MEM_SIZE-50*1000, Const.
               MISSION_MEM_SIZE-100*1000);
22
23     return new FlatBufferMissionSequencer(new PriorityParameters(5),
           storageParameters);
24   }
25
26
27   @Override
28   public long immortalMemorySize()
29   {
30     return Const.IMMORTAL_MEM_SIZE;
31   }
32
33   @Override
34   public void initializeApplication()
35   {
36
37   }
38 }
```

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.StorageParameters;
7
8  public class FlatBufferMissionSequencer extends MissionSequencer<Mission>
9  {
10   private boolean returnedMission;
11
12   public FlatBufferMissionSequencer(PriorityParameters priorityParameters,
13       StorageParameters storageParameters)
14   {
15     super(priorityParameters, storageParameters);
16     returnedMission = false;
17   }
18
19   protected Mission getNextMission()
20   {
21     System.out.println("FlatBufferMissionSequencer");
22
23     // As this sequencer only delivers one mission,
24     // if it has not been returned yet then return it,
25     // else return null which will terminate the sequencer
26
27     if (!returnedMission)
28     {
29       System.out.println("FlatBufferMissionSequencer returns mission");
30       returnedMission = true;
31       return new FlatBufferMission();
32     }
33     else
34     {
35       return null;
36     }
37   }
38
39 }
```

4 FlatBufferMission

```
1  package scjlevel2examples.flatbuffer;
2
3  import javax.realtime.PriorityParameters;
4  import javax.safetycritical.Mission;
5  import javax.safetycritical.StorageParameters;
6  import javax.scj.util.Const;
7
8  public class FlatBufferMission extends Mission
9  {
10   private volatile int[] buffer;
11   private Writer writer;
12   private Reader reader;
13
14   public FlatBufferMission()
15   {
16     buffer = new int[1];
17     buffer[0] = 0;
18
19     System.out.println("FlatBufferMission");
20   }
21
22   protected void initialize()
23   {
24     StorageParameters storageParameters = new StorageParameters(1048576,
25         new long[] { Const.HANDLER_STACK_SIZE }, 1048576, 1048576,
26         Const.MISSION_MEM_SIZE - 100 * 1000);
27
28     reader = new Reader(new PriorityParameters(5), storageParameters, this,
29         writer);
30
31     writer = new Writer(new PriorityParameters(5), storageParameters, this,
32         reader);
33
34     System.out.println("FlatBufferMission init");
35   }
36
37   public boolean bufferEmpty()
38   {
39     return buffer[0] == 0;
40   }
41
42   public synchronized void write(int update)
43   {
44     buffer[0] = update;
45   }
46
47   public synchronized int read()
48   {
49     int out = buffer[0];
50     this.buffer[0] = 0;
51
52     return out;
53   }
54
55   public long missionMemorySize()
56   {
57     return 1048576;
58   }
59 }
```

5 Writer

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

6 Reader

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