1 EchoApplication

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

2 EchoMissionSequencer

```
1  package scjlevel2examples.echo;
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 EchoMissionSequencer extends MissionSequencer
10 {
11   private boolean returnedMission;
12
13   public EchoMissionSequencer(PriorityParameters priorityParameters,
14       StorageConfigurationParameters storageConfigurationParameters)
15   {
16     super(priorityParameters, storageConfigurationParameters);
17     returnedMission =false;
18   }
19
20   protected Mission getNextMission()
21   {
22     //As this sequencer only delivers one mission,
23     //if it has not been returned yet then return it,
24     //else return null which will terminate the sequencer
25
26     if(!returnedMission)
27     {
28       returnedMission = true;
29       return new EchoMission();
30     }
31     else
32     {
33       return null;
34     }
35   }
36 }
```

3 EchoMission

```
1  package scjlevel2examples.echo;
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 EchoMission extends Mission
12 {
13   private volatile String buffer;
14
15   protected void initialize()
16   {
17     //start the two submission sequencers, note a reference to this object is
           passed to both so that they can access the buffer
18     EchoInputMissionSequencer echoInputMissionSequencer =
19       new EchoInputMissionSequencer (new PriorityParameters(5),
20                     new StorageConfigurationParameters(1048576, 1048576, 1048576),
21                     this);
22
23     EchoOutputMissionSequencer echoOutputMissionSequencer =
24       new EchoOutputMissionSequencer (new PriorityParameters(5),
25                      new StorageConfigurationParameters(1048576, 1048576, 1048576),
26                      this);
27
28     ehcoInputMissionSequencer.register();
29     echoOutputMissionSequencer.register();
30
31     buffer = "These are words in the buffer"; //though they are never read
32   }
33
34   public void put(final String words)
35   {
36     ManagedMemory.enterPrivateMemory(100, new Runnable()
37     {
38       public void run()
39       {
40         buffer = words;
41       }
42
43     });
44   }
45
46   public String get()
47   {
48     return buffer;
49   }
50
51   public long missionMemorySize()
52   {
53     return 100000;
54   }
55
56 }
```

4 EchoInputMissionSequencer

```
1  package scjlevel2examples.echo;
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 EchoInputMissionSequencer extends MissionSequencer
10 {
11   private boolean returnedMission;
12   private final EchoMission echoMission;
13
14   public EchoInputMissionSequencer(PriorityParameters priorityParameters,
15       StorageConfigurationParameters storageConfigurationParameters, EchoMission
            echoMission)
16   {
17     super(priorityParameters, storageConfigurationParameters);
18     returnedMission = false;
19     this.echoMission = echoMission;
20   }
21
22   protected Mission getNextMission()
23   {
24     //As this sequencer only delivers one mission,
25     //if it has not been returned yet then return it,
26     //else return null which will terminate the sequencer
27
28     if(!returnedMission)
29     {
30       returnedMission = true;
31       return new EchoInputMission(echoMission);
32     }
33     else
34     {
35       return null;
36     }
37   }
38 }
```

5 EchoInputMission

```
1  package scjlevel2examples.echo;
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 EchoInputMission extends Mission
12 {
13   private final EchoMission echoMission;
14
15   public EchoInputMission(EchoMission echoMission)
16   {
17     super();
18     this.echoMission= echoMission;
19   }
20
21   protected void initialize()
22   {
23     //Start this mission's handler
24     EchoInputter echoInputter = new EchoInputter(new PriorityParameters(10),
25         new PeriodicParameters(new RelativeTime(100, 0)),
26         new StorageConfigurationParameters(1000, 1000, 1000),
27         1000,
28         echoMission);
29
30     echoInputter.register();
31   }
32
33   public long missionMemorySize()
34   {
35     return 100000;
36   }
37
38 }
```

6 EchoInputter

```
1  package scjlevel2examples.echo;
2
3  import javax.realtime.PeriodicParameters;
4  import javax.realtime.PriorityParameters;
5  import javax.safetycritical.PeriodicEventHandler;
6  import javax.safetycritical.StorageConfigurationParameters;
7
8  public class EchoInputter extends PeriodicEventHandler
9  {
10   private final EchoMission echoMission;
11   private int i;
12   private String[] words = {"Fight","From","The","Inside",
13         "Attack","From","The","Rear",
14         "Fight","From","The","Inside",
15         "You","Can't","Win","With","Your","Hands","Tied"};
16   private final int arraySize = words.length;
17
18   public EchoInputter(PriorityParameters priority, PeriodicParameters periodic,
19       StorageConfigurationParameters storage, long size, EchoMission echoMission)
20   {
21     super(priority, periodic, storage, size);
22
23     this.echoMission = echoMission;
24     i =0;
25   }
26
27   public void handleEvent()
28   {
29     //Put the next word from the array into the buffer in the main mission (
           EchoMission)
30     echoMission.put(words[i]);
31
32     //if the end of the array has been reached, reset the index; else increment the
           index
33     if(i<arraySize)
34     {
35       i = i+1;
36     }
37     else
38     {
39       i = 0;
40     }
41
42   }
43 }
```

7 EchoOutputMissionSequencer

```
1  package scjlevel2examples.echo;
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 EchoOutputMissionSequencer extends MissionSequencer
10 {
11   private boolean returnedMission;
12   private final EchoMission echoMission;
13
14   public EchoOutputMissionSequencer(PriorityParameters priorityParameters,
15       StorageConfigurationParameters storageConfigurationParameters, EchoMission
            echoMission)
16   {
17     super(priorityParameters, storageConfigurationParameters);
18     returnedMission = false;
19     this.echoMission = echoMission;
20   }
21
22   protected Mission getNextMission()
23   {
24     //As this sequencer only delivers one mission,
25     //if it has not been returned yet then return it,
26     //else return null which will terminate the sequencer
27     if(!returnedMission)
28     {
29       returnedMission = true;
30       return new EchoOutputMission(echoMission);
31     }
32     else
33     {
34       return null;
35     }
36   }
37 }
```

8 EchoOutputMission

```
1  package scjlevel2examples.echo;
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 EchoOutputMission extends Mission
12 {
13   private final EchoMission echoMission;
14
15   public EchoOutputMission(EchoMission echoMission)
16   {
17     super();
18     this.echoMission= echoMission;
19   }
20
21   protected void initialize()
22   {
23     //Start this mission's handler
24     EchoOutputter echoOutputter = new EchoOutputter(new PriorityParameters(10),
25         new PeriodicParameters(new RelativeTime(100, 0)),
26         new StorageConfigurationParameters(1000, 1000, 1000),
27         1000,
28         echoMission);
29
30     echoOutputter.register();
31   }
32
33   public long missionMemorySize()
34   {
35     return 100000;
36   }
37
38 }
```

9 EchoOutputter

```
1  package scjlevel2examples.echo;
2
3  import javax.realtime.PeriodicParameters;
4  import javax.realtime.PriorityParameters;
5  import javax.safetycritical.PeriodicEventHandler;
6  import javax.safetycritical.StorageConfigurationParameters;
7  import java.io.PrintStream;
8
9  public class EchoOutputter extends PeriodicEventHandler
10 {
11   private final EchoMission echoMission;
12   private PrintStream ps;
13
14   public EchoOutputter(PriorityParameters priority, PeriodicParameters periodic,
15       StorageConfigurationParameters storage, long size, EchoMission echoMission)
16   {
17     super(priority, periodic, storage, size);
18
19     this.echoMission = echoMission;
20     ps = new PrintStream(System.out, true);
21   }
22
23   public void handleEvent()
24   {
25     //Print the contents of the buffer
26     ps.println(echoMission.get());
27
28   }
29 }
```