

1 EchoSafeletExecuter

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

2 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
10     @Override
11     public Level getLevel()
12     {
13         return Level.LEVEL2;
14     }
15
16     @Override
17     public MissionSequencer getSequencer()
18     {
19         System.out.println("EchoApplication");
20         //Create and return the main mission sequencer
21         return new EchoMissionSequencer(new PriorityParameters(5),
22             new StorageConfigurationParameters(1048576, 1048576, 1048576));
23     }
24
25     @Override
26     public void setup()
27     {
28     }
29
30     @Override
31     public void teardown()
32     {
33     }
34
35 }
36 }
```

3 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     @Override
21     protected Mission getNextMission()
22     {
23         System.out.println("EchoMissionSequencer");
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("EchoMissionSequencer returns mission");
32             returnedMission = true;
33             return new EchoMission();
34         }
35         else
36         {
37             return null;
38         }
39     }
40
41
42
43 }
```

4 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     @Override
16     protected void initialize()
17     {
18         //start the two submission sequencers, note a reference to this object is
19         passed to both so that they can access the buffer
20         EchoInputMissionSequencer echoInputMissionSequencer = new
21             EchoInputMissionSequencer (new PriorityParameters(5),
22                                         new StorageConfigurationParameters(1048576, 1048576,
23                                         1048576), this );
24
25         EchoOutputMissionSequencer echoOutputMissionSequencer = new
26             EchoOutputMissionSequencer (new PriorityParameters(5),
27                                         new StorageConfigurationParameters(1048576, 1048576,
28                                         1048576), this);
29         buffer = "These are words in the buffer"; //though they are never read
30
31         System.out.println("EchoMission");
32     }
33
34     public void put(String words)
35     {
36         buffer = words;
37     }
38
39     public String get()
40     {
41         return buffer;
42     }
43
44     @Override
45     public long missionMemorySize()
46     {
47         return 100000;
48     }
49 }
```

5 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
16         echoMission)
17     {
18         super(priorityParameters , storageConfigurationParameters);
19         returnedMission = false;
20         this.echoMission = echoMission;
21         System.out.println("EchoInputMissionSequencer constructor");
22     }
23
24     @Override
25     protected Mission getNextMission()
26     {
27         System.out.println("EchoInputMissionSequencer getNextMission");
28
29         //As this sequencer only delivers one mission ,
30         //if it has not been returned yet then return it ,
31         //else return null which will terminate the sequencer
32
33         if(!returnedMission)
34         {
35             System.out.println("EchoInputMissionSequencer returns mission");
36             returnedMission = true;
37             return new EchoInputMission(echoMission);
38         }
39         else
40         {
41             return null;
42         }
43     }
44 }
```

6 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     @Override
22     protected void initialize()
23     {
24         //Start this mission's handler
25         EchoInputter echoInputter = new EchoInputter(new PriorityParameters(10),
26             new PeriodicParameters(new RelativeTime(100, 0)),
27             new StorageConfigurationParameters(1000, 1000, 1000),
28             1000,
29             echoMission);
30
31     }
32
33     @Override
34     public long missionMemorySize()
35     {
36         return 100000;
37     }
38
39 }
```

7 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 import java.io.*;
9
10
11 public class EchoInputter extends PeriodicEventHandler
12 {
13     private final EchoMission echoMission;
14     private int i;
15     private String[] words = {"Fight","From","The","Inside",
16                             "Attack","From","The","Rear",
17                             "Fight","From","The","Inside",
18                             "You","Can't","Win","With","Your","Hands","Tied"};
19
20     public EchoInputter(PriorityParameters priority, PeriodicParameters periodic,
21                        StorageConfigurationParameters storage, long size, EchoMission echoMission)
22     {
23         super(priority, periodic, storage, size);
24
25         this.echoMission = echoMission;
26         i = 0;
27     }
28
29     @Override
30     public void handleEvent()
31     {
32         //Put the next word from the array into the buffer in the main mission (
33             EchoMission)
34         echoMission.put(words[i]);
35
36         //if the end of the array has been reached, reset the index; else increment the
37             index
38         if (i < 18)
39         {
40             i = i + 1;
41         }
42         else
43         {
44             i = 0;
45         }
46     }
47 }
```

8 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
16         echoMission)
17     {
18         super(priorityParameters, storageConfigurationParameters);
19         returnedMission = false;
20         this.echoMission = echoMission;
21         System.out.println("EchoOutputMissionSequencer constructor");
22     }
23
24     @Override
25     protected Mission getNextMission()
26     {
27         System.out.println(" EchoOutputMissionSequencer getNextMission");
28
29         //As this sequencer only delivers one mission,
30         //if it has not been returned yet then return it,
31         //else return null which will terminate the sequencer
32         if(!returnedMission)
33         {
34             System.out.println("EchoOutputMissionSequencer returns mission");
35             returnedMission = true;
36             return new EchoOutputMission(echoMission);
37         }
38         else
39         {
40             return null;
41         }
42     }
43 }
```


9 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     @Override
22     protected void initialize()
23     {
24         //Start this mission's handler
25         EchoOutputter echoOutputter = new EchoOutputter(new PriorityParameters(10),
26             new PeriodicParameters(new RelativeTime(100, 0)),
27             new StorageConfigurationParameters(1000, 1000, 1000),
28             1000,
29             echoMission);
30
31     }
32
33     @Override
34     public long missionMemorySize()
35     {
36         return 100000;
37     }
38
39 }
```

10 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
8
9 public class EchoOutputter extends PeriodicEventHandler
10 {
11     private final EchoMission echoMission;
12
13     public EchoOutputter(PriorityParameters priority, PeriodicParameters periodic,
14         StorageConfigurationParameters storage, long size, EchoMission echoMission)
15     {
16         super(priority, periodic, storage, size);
17         this.echoMission = echoMission;
18     }
19
20     @Override
21     public void handleEvent()
22     {
23         //Print the contents of the buffer
24         System.out.println(echoMission.get());
25     }
26 }
27
28
29 }
```