1 TestSafelet

```
1  /** Simple Nested Sequencer
2   *
3   *
4   *    @author Matt Luckcuck <ml881@york.ac.uk>
5   */
6  package scjlevel2examples.simpleNestedSequencer;
7
8  import javax.realtime.PriorityParameters;
9  import javax.safetycritical.Mission;
10 import javax.safetycritical.MissionSequencer;
11 import javax.safetycritical.Safelet;
12 import javax.safetycritical.StorageParameters;
13 import javax.scj.util.Const;
14
15 import devices.Console;
16
17 public class TestSafelet implements Safelet<Mission>
18 {
19
20   // public static StorageParameters storageParametersSchedulable;
21   public static StorageParameters storageParameters_topLevelSequencer;
22   public static StorageParameters storageParameters_nestedSequencer;
23   public static StorageParameters storageParameters_Schedulable;
24
25   @Override
26   public MissionSequencer<Mission> getSequencer()
27   {
28     Console.println("TestSafelet: getSequencer");
29
30     storageParameters_topLevelSequencer = new StorageParameters(
31         1000000,
32         new long[] { Const.HANDLER_STACK_SIZE },
33         100000,
34         10000 ,
35         100000);
36
37     return new MainMissionSequencer(new PriorityParameters(5),
38         storageParameters_topLevelSequencer);
39   }
40
41   @Override
42   public long immortalMemorySize()
43   {
44     return 100000;
45   }
46
47   @Override
48   public void initializeApplication()
49   {
50     Console.println("TestSafelet: Init");
51     // storageParameters = new StorageParameters(150 * 1000, new long[] {
52     // Const.HANDLER_STACK_SIZE },
53     // Const.PRIVATE_MEM_SIZE-25*1000, Const.IMMORTAL_MEM_SIZE-50*1000,
54     // Const.MISSION_MEM_SIZE-100*1000);
55     //
56     // storageParameters_Schedulable = new
57     // StorageParameters(Const.PRIVATE_MEM_SIZE-30*1000, new long[] {
58     // Const.HANDLER_STACK_SIZE },
59     // Const.PRIVATE_MEM_SIZE-30*1000, Const.IMMORTAL_MEM_SIZE-50*1000,
60     // Const.MISSION_MEM_SIZE-100*1000);
61
62 //     storageParameters_nestedSequencer = new StorageParameters(
63 //         (Const.BACKING_STORE_SIZE_DEFAULT / 2) + 1000000,
64 //         new long[] { Const.HANDLER_STACK_SIZE },
65 //         Const.PRIVATE_MEM_SIZE_DEFAULT, 10000 * 2,
66 //         Const.MISSION_MEM_SIZE_DEFAULT);
67
68
```

```
69      storageParameters_nestedSequencer = new StorageParameters(
70          1000000,
71      new long[] { Const.HANDLER_STACK_SIZE },
72      100000,
73      10000 ,
74      0);
75
76      storageParameters_Schedulable = new StorageParameters(
77          1000000,
78      new long[] { Const.HANDLER_STACK_SIZE },
79      10000,
80      10000 ,
81      0);
82
83      // storageParametersSchedulable = new StorageParameters(
84      // Const.BACKING_STORE_SIZE ,
85      // new long[] { Const.HANDLER_STACK_SIZE },
86      // Const.PRIVATE_MEM_SIZE_DEFAULT,
87      // Const.IMMORTAL_MEM_SIZE_DEFAULT,
88      // Const.MISSION_MEM_SIZE_DEFAULT );
89
90      Console.println("TestSafelet: Begin");
91    }
92
93  }
```

2 MainMissionSequencer

```
1  /** Simple Nested Sequencer
2   *
3   *
4   *    @author Matt Luckcuck <ml881@york.ac.uk>
5   */
6  package scjlevel2examples.simpleNestedSequencer;
7
8  import javax.realtime.PriorityParameters;
9  import javax.safetycritical.Mission;
10 import javax.safetycritical.MissionSequencer;
11 import javax.safetycritical.StorageParameters;
12
13 import devices.Console;
14
15 public class MainMissionSequencer extends MissionSequencer<Mission>
16 {
17
18   /**
19    * Has this single mission been returned?
20    */
21   private boolean returnedMission;
22
23   /**
24    * Class Constructor
25    *
26    * @param pp
27    *             the PriorityParameters for the sequencer
28    * @param sp
29    *             the StorageParameters for the sequencer
30    */
31   public MainMissionSequencer(PriorityParameters pp, StorageParameters sp)
32   {
33     super(pp, sp);
34     returnedMission = false;
35   }
36
37   /**
38    * Returns the new mission
39    */
40   @Override
41   protected Mission getNextMission()
42   {
43     Console.println("MainMissioNSequencer: getNextMission");
44     // This returns the main mission once only
45     if (returnedMission)
46     {
47       Console.println("MainMissioNSequencer: returing null");
48       return null;
49
50     } else
51     {
52       Console.println("MainMissioNSequencer: returning MainMission");
53       returnedMission = true;
54       return new MainMission();
55     }
56   }
57
58 }
```

3 MainMission

```
1  /** Simple Nested Sequencer
2   *
3   *
4   *    @author Matt Luckcuck <ml881@york.ac.uk>
5   */
6  package scjlevel2examples.simpleNestedSequencer;
7
8  import javax.realtime.PriorityParameters;
9  import javax.safetycritical.Mission;
10 import javax.scj.util.Const;
11
12 import devices.Console;
13
14 public class MainMission extends Mission
15 {
16   @Override
17   protected void initialize()
18   {
19     Console.println("Main Mission: Init ");
20
21     NestedMissionSequencer sPModeChanger = new NestedMissionSequencer(
22         new PriorityParameters(5),
23         TestSafelet.storageParameters_nestedSequencer);
24
25     Console.println("Main Mission: Nested Sequencers Init");
26
27     sPModeChanger.register();
28
29     Console.println("Main Mission: Nested Sequencer register");
30
31     Console.println("Main Mission: Begin ");
32   }
33
34   /**
35    * Returns the required size of this Mission's private memory
36    */
37   @Override
38   public long missionMemorySize()
39   {
40     return 10000;
41   }
42
43 }
```

3.1 Schedulables of MainMission

3.2 NestedMissionSequencer

```
1  /** Simple Nested Sequencer
2   *
3   *
4   *    @author Matt Luckcuck <ml881@york.ac.uk>
5   */
6  package scjlevel2examples.simpleNestedSequencer;
7
8  import javax.realtime.PriorityParameters;
9  import javax.safetycritical.Mission;
10 import javax.safetycritical.MissionSequencer;
11 import javax.safetycritical.StorageParameters;
12
13 import devices.Console;
14
15 public class NestedMissionSequencer extends MissionSequencer<Mission>
16 {
17
18   private boolean returnedMission = false;
19
20   public NestedMissionSequencer(PriorityParameters priority,
21       StorageParameters storage)
22   {
23     super(priority, storage);
24     Console.println("Nested Mission Sequencer: Construct ");
25
26   }
27
28   @Override
29   protected Mission getNextMission()
30   {
31     Console.println("Mode Changer: getNextMission ");
32
33     if (returnedMission)
34     {
35       return null;
36     } else
37     {
38       returnedMission = true;
39       return new NestedMission();
40     }
41
42   }
43
44 }
```

4 NestedMission

```
1  /** Simple Nested Sequencer
2   *
3   *
4   *    @author Matt Luckcuck <ml881@york.ac.uk>
5   */
6  package scjlevel2examples.simpleNestedSequencer;
7
8  import javax.realtime.AperiodicParameters;
9  import javax.realtime.HighResolutionTime;
10 import javax.realtime.PriorityParameters;
11 import javax.realtime.RelativeTime;
12 import javax.safetycritical.Mission;
13
14 import javax.scj.util.Const;
15
16 import devices.Console;
17
18 public class NestedMission extends Mission
19 {
20
21   @Override
22   protected void initialize()
23   {
24     Console.println("Launch Mission: Init ");
25
26     // Initially false because the conditions haven't been checked yet
27     NestedOneShotEventHandler nestedOneShot =  new NestedOneShotEventHandler(new
           PriorityParameters(5),
28         new RelativeTime(5, 0),
29         new AperiodicParameters(),
30         TestSafelet.storageParameters_Schedulable);
31
32     nestedOneShot.register();
33
34     Console.println("Launch Mission: Begin ");
35   }
36
37   /**
38    * Returns the size of the mission's memory
39    */
40   @Override
41   public long missionMemorySize()
42   {
43     return 10000;
44   }
45
46 }
```

4.1 Schedulables of NestedMission

4.2 NestedOneShotEventHandler

```
1  package scjlevel2examples.simpleNestedSequencer;
2
3  import javax.safetycritical.OneShotEventHandler;
4  import javax.safetycritical.StorageParameters;
5  import javax.realtime.AperiodicParameters;
6  import javax.realtime.PriorityParameters;
7  import javax.realtime.HighResolutionTime;
8
9  import devices.Console;
10
11 public class NestedOneShotEventHandler extends OneShotEventHandler
12 {
13
14
15   public NestedOneShotEventHandler(PriorityParameters priority, HighResolutionTime
         time, AperiodicParameters release, StorageParameters storage)
16   {
17     super(priority, time, release, storage);
18   }
19
20   @Override
21   public void handleAsyncEvent()
22   {
23     Console.println("Nested One-Shot: Release");
24   }
25 }
```