1 EchoSafeletExecuter

```java
package scjlevel2examples.echo;

import javax.safetycritical.Safelet;
import javax.safetycritical.SafeletExecuter;

//Application entry point, runs the Safelet
public class EchoSafeletExecuter extends SafeletExecuter
{

  public EchoSafeletExecuter(Safelet arg0)
  {
    super(arg0);
  }

  public static void main (String [] args)
  {
    System.out.println("EchoSafeletExecuter ");
    //Run the safelet which starts the whole application
    EchoSafeletExecuter.run(new EchoApplication());
  }
}
```

2 EchoApplication

```java
package scjlevel2examples.echo;

import javax.realtime.PriorityParameters;
import javax.safetycritical.*;
import javax.safetycritical.annotate.Level;

public class EchoApplication implements Safelet
{

  @Override
  public Level getLevel()
  {
    return Level.LEVEL_2;
  }

  @Override
  public MissionSequencer getSequencer()
  {
    System.out.println("EchoApplication");
    //Create and return the main mission sequencer
    return new EchoMissionSequencer(new PriorityParameters(5),
        new StorageConfigurationParameters(1048576, 1048576, 1048576));
  }

  @Override
  public void setup()
  {
  }

  @Override
  public void teardown()
  {
  }


}
```

3 EchoMissionSequencer

```java
package scjlevel2examples.echo;

import javax.realtime.PriorityParameters;
import javax.safetycritical.Mission;
import javax.safetycritical.MissionSequencer;
import javax.safetycritical.StorageConfigurationParameters;


public class EchoMissionSequencer extends MissionSequencer
{
  private boolean returnedMission;

  public EchoMissionSequencer(PriorityParameters priorityParameters,
      StorageConfigurationParameters storageConfigurationParameters)
  {
    super(priorityParameters, storageConfigurationParameters);
    returnedMission =false;
  }

  @Override
  protected Mission getNextMission()
  {
    System.out.println("EchoMissionSequencer");

    //As this sequencer only delivers one mission,
    //if it has not been returned yet then return it,
    //else return null which will terminate the sequencer

    if(!returnedMission)
    {
      System.out.println("EchoMissionSequencer returns mission");
      returnedMission = true;
      return new EchoMission();
    }
    else
    {
      return null;
    }
  }



}
```

4 EchoMission

```
package scjlevel2examples.echo;

import javax.realtime.PeriodicParameters;
import javax.realtime.PriorityParameters;
import javax.realtime.RelativeTime;
import javax.safetycritical.Mission;
import javax.safetycritical.PriorityScheduler;
import javax.safetycritical.StorageConfigurationParameters;


public class EchoMission extends Mission
{
  private volatile String buffer;

  @Override
  protected void initialize()
  {
    //start the two submission sequencers, note a reference to this object is
        passed to both so that they can access the buffer
    EchoInputMissionSequencer echoInputMissionSequencer = new
        EchoInputMissionSequencer (new PriorityParameters(5),
                        new StorageConfigurationParameters(1048576, 1048576,
                            1048576), this );

    EchoOutputMissionSequencer echoOutputMissionSequencer = new
        EchoOutputMissionSequencer (new PriorityParameters(5),
                          new StorageConfigurationParameters(1048576, 1048576,
                              1048576), this);
    buffer = "These are words in the buffer"; //though they are never read

    System.out.println("EchoMission");
  }

  public void put(String words)
  {
    buffer = words;
  }

  public String get()
  {
    return buffer;
  }

  @Override
  public long missionMemorySize()
  {
    return 100000;
  }

}
```

5 EchoInputMissionSequencer

```java
package scjlevel2examples.echo;

import javax.realtime.PriorityParameters;
import javax.safetycritical.Mission;
import javax.safetycritical.MissionSequencer;
import javax.safetycritical.StorageConfigurationParameters;


public class EchoInputMissionSequencer extends MissionSequencer
{
  private boolean returnedMission;
  private final EchoMission echoMission;

  public EchoInputMissionSequencer(PriorityParameters priorityParameters,
      StorageConfigurationParameters storageConfigurationParameters, EchoMission
          echoMission)
  {
    super(priorityParameters, storageConfigurationParameters);
    returnedMission = false;
    this.echoMission = echoMission;
    System.out.println("EchoInputMissionSequencer constructor");
  }

  @Override
  protected Mission getNextMission()
  {
    System.out.println("EchoInputMissionSequencer getNextMission");

    //As this sequencer only delivers one mission,
    //if it has not been returned yet then return it,
    //else return null which will terminate the sequencer

    if(!returnedMission)
    {
      System.out.println("EchoInputMissionSequencer returns mission");
      returnedMission = true;
      return new EchoInputMission(echoMission);
    }
    else
    {
      return null;
    }
  }
}
```

6 EchoInputMission

```
package scjlevel2examples.echo;

import javax.realtime.PeriodicParameters;
import javax.realtime.PriorityParameters;
import javax.realtime.RelativeTime;
import javax.safetycritical.Mission;
import javax.safetycritical.PriorityScheduler;
import javax.safetycritical.StorageConfigurationParameters;


public class EchoInputMission extends Mission
{
  private final EchoMission echoMission;

  public EchoInputMission(EchoMission echoMission)
  {
    super();
    this.echoMission= echoMission;
  }

  @Override
  protected void initialize()
  {
    //Start this mission's handler
    EchoInputter echoInputter = new EchoInputter(new PriorityParameters(10),
        new PeriodicParameters(new RelativeTime(100, 0)),
        new StorageConfigurationParameters(1000, 1000, 1000),
        1000,
        echoMission);

  }

  @Override
  public long missionMemorySize()
  {
    return 100000;
  }

}
```

7 EchoInputter

```java
package scjlevel2examples.echo;

import javax.realtime.PeriodicParameters;
import javax.realtime.PriorityParameters;
import javax.safetycritical.PeriodicEventHandler;
import javax.safetycritical.StorageConfigurationParameters;

import java.io.*;


public class EchoInputter extends PeriodicEventHandler
{
  private final EchoMission echoMission;
  private int i;
  private String[] words = {"Fight","From","The","Inside",
        "Attack","From","The","Rear",
        "Fight","From","The","Inside",
        "You","Can't","Win","With","Your","Hands","Tied"};

  public EchoInputter(PriorityParameters priority, PeriodicParameters periodic,
      StorageConfigurationParameters storage, long size, EchoMission echoMission)
  {
    super(priority, periodic, storage, size);

    this.echoMission = echoMission;
    i =0;
  }

  @Override
  public void handleEvent()
  {
    //Put the next word from the array into the buffer in the main mission (
        EchoMission)
    echoMission.put(words[i]);

    //if the end of the array has been reached, reset the index; else increment the
        index
    if(i<18)
    {
      i = i+1;
    }
    else
    {
      i = 0;
    }

  }
}
```

8 EchoOutputMissionSequencer

```java
package scjlevel2examples.echo;

import javax.realtime.PriorityParameters;
import javax.safetycritical.Mission;
import javax.safetycritical.MissionSequencer;
import javax.safetycritical.StorageConfigurationParameters;


public class EchoOutputMissionSequencer extends MissionSequencer
{
  private boolean returnedMission;
  private final EchoMission echoMission;

  public EchoOutputMissionSequencer(PriorityParameters priorityParameters,
      StorageConfigurationParameters storageConfigurationParameters, EchoMission
          echoMission)
  {
    super(priorityParameters, storageConfigurationParameters);
    returnedMission = false;
    this.echoMission = echoMission;
    System.out.println("EchoOutputMissionSequencer constructor");
  }

  @Override
  protected Mission getNextMission()
  {
    System.out.println(" EchoOutputMissionSequencer getNextMission");

    //As this sequencer only delivers one mission,
    //if it has not been returned yet then return it,
    //else return null which will terminate the sequencer
    if(!returnedMission)
    {
      System.out.println("EchoOutputMissionSequencer returns mission");
      returnedMission = true;
      return new EchoOutputMission(echoMission);
    }
    else
    {
      return null;
    }
  }

}
```

9 EchoOutputMission

```java
package scjlevel2examples.echo;

import javax.realtime.PeriodicParameters;
import javax.realtime.PriorityParameters;
import javax.realtime.RelativeTime;
import javax.safetycritical.Mission;
import javax.safetycritical.PriorityScheduler;
import javax.safetycritical.StorageConfigurationParameters;


public class EchoOutputMission extends Mission
{
  private final EchoMission echoMission;

  public EchoOutputMission(EchoMission echoMission)
  {
    super();
    this.echoMission= echoMission;
  }

  @Override
  protected void initialize()
  {
    //Start this mission's handler
    EchoOutputter echoOutputter = new EchoOutputter(new PriorityParameters(10),
        new PeriodicParameters(new RelativeTime(100, 0)),
        new StorageConfigurationParameters(1000, 1000, 1000),
        1000,
        echoMission);

  }

  @Override
  public long missionMemorySize()
  {
    return 100000;
  }

}
```

10 EchoOutputter

```java
package scjlevel2examples.echo;

import javax.realtime.PeriodicParameters;
import javax.realtime.PriorityParameters;
import javax.safetycritical.PeriodicEventHandler;
import javax.safetycritical.StorageConfigurationParameters;


public class EchoOutputter extends PeriodicEventHandler
{
  private final EchoMission echoMission;

  public EchoOutputter(PriorityParameters priority, PeriodicParameters periodic,
      StorageConfigurationParameters storage, long size, EchoMission echoMission)
  {
    super(priority, periodic, storage, size);

    this.echoMission = echoMission;
  }

  @Override
  public void handleEvent()
  {
    //Print the contents of the buffer
    System.out.println(echoMission.get());

  }

}
```