Start Building the Simulator Class

As we have just seen with the PIA class we begin to build a class by creating a unit test. In this case we create a unit test for a message that the simulator will send to the GUI. Let's start with the boiler.

What we want to do here is start up the simulator in its own Thread then change the PIA register. We then wait a fraction of second and then make sure a message was sent to the GUI.

```
package simulator.r5.unittest;

import unittest.framework.*;
import simulator.r5.*;

class TestBoilerOn extends Test implements SimulationInterface
   {private int messageSent;
   private Thread simulation;

   public void setUp()
      {messageSent = 0;
      PIA.register = 0x0000;
      PIA.setInputs(0x003F);
      startSimulator();}

   public void runTest()
      {PIA.write(0x1000);
      pauseOneHalfSecond();
      should(messageSent == 1, "Got boilerOn " + messageSent + " instead of
once");}

   public void tearDown()
      {stopSimulator();}

   public void boilerOn()
      {messageSent++;}

   private void startSimulator()
      {simulation = new Thread(new Simulator(this));
      simulation.run();}

   private void pauseOneHalfSecond()
      {try
         {Thread.sleep(500);}
      catch (Exception exception)
         {};}

   private void stopSimulator()
      {simulation.stop();
      simulation = null;};}
```

With the test framework we are using we will also need to add this test to the test suite we are building.

```
package simulator.r5.unittest;

import unittest.framework.*;

public class SimulatorTests extends TestSuite
{public SimulatorTests()
{tests = new Test[3];
tests[0] = new TestBoilerOn();
tests[1] = new TestReadFromPIA();
tests[2] = new TestWriteToPIA();};}
```

With Java we need to create stubs for all the messages we will send before we can compile.

```
package simulator.r5;

public class Simulator implements Runnable
   {public Simulator (SimulationInterface aGUI)
      {}

   public void run()
      {};}

public interface SimulationInterface
   {public void boilerOn();}
```

Now let's compile and run this new unit test to make sure that if fails. It fails as expected so we can now create some code to replace the stubs we just created.

```
package simulator.r6;

public class Simulator extends Thread
   {SimulationInterface gui;

   public Simulator (SimulationInterface aGUI)
      {super();
      gui = aGUI;}

   public void run()
      {for (int each = 0; each < 50; each++)
         {if ((PIA.register & 0x1000) > 0) gui.boilerOn();
         try
            {sleep(100);}
         catch (InterruptedException exception)
            {};};};}
```

Now we can run the unit tests again to see if we have the required functionality. The test still fails. Why? Let's read what the test says, we got more than one message. We forgot to make sure the message gets sent only the first time the register is changed.