Test the PIA Class Into Existence

The first bit of programming we can do is for the PIA class. We know that we need to simulate the hardware and we know what that needs to look like. There will be 3 methods, one for programming the input and output bits, one to read, and one to write to the PIA. So let's create the PIA first.

The way that we create code in Extreme Programming (XP) is to start with a test. Fortunately we have already created a unit testing framework. So lets create some code for our first test. What we want here is to be able to read the PIA and have any bits programmed to output always be zero. Input bits are unaffected when being read.

```
package simulator.r1.unittest;

import unittest.framework.*;
import simulator.r1.*;

public class TestReadFromPIA extends Test
   {public void runTest()
      {PIA.register = 0x0F0F;
      PIA.setInputs(0x00FF);
      should(PIA.read() == 0x000F, "Outputs should always be zero");};}
```

To be able to compile and run this test we need to create a stub class for our PIA. We will just create the methods and not bother writing code.

```
package simulator.r1;

public class PIA
   {public static int register;

   public static int read()
      {return 0x0000;}

   public static void setInputs(int aShort)
      {};}
```

Next we can create a TestSuite for our first test.

```
package simulator.r1.unittest;

import unittest.framework.*;

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

Now we can run this test. It fails of course. But we like to run it to make sure. Occasionally the test will pass, which means that our code already does what we need or our test doesn't test what we need. Lets go back to the PIA class and write some code to make the test work. We add some code and get the following.

```
package simulator.r2;

class PIA
   {public static int register = 0;
   public static int inputBits = 0;

   public static int read()
      {return register & inputBits;}

   public static void setInputs(int aBitMask)
      {inputBits = aBitMask;};}
```

Let's try the unit test now. It runs as expected. Our code is still nice and simple and clean so we don't need to refactor. So let's add a second test. We need to test writing to the PIA.