Tutorial Exercises for:

xUnit Test Patterns and Smells

Presented by Gerard Meszaros
xunitTutorial@gerardmeszaros.com

and

Geoff Hardy
geoff@clrstream.com

Exercise CS1 - Result Verification

Situation

You have just inherited maintenance of the Flight Management System. The good news is that there are automated unit tests. The bad news is that most of the tests look something like these tests.

Instructions:

Examine the code in the handout and determine what code smells you are seeing.

Discussion Questions:

- Which Code Smells are we having?
- What are the underlying root causes?
- Which Patterns can we apply to alleviate them?

Test Code

```
public void testGetFlightsByOriginAirports () throws Exception {
   // create 3 flights from one airport to two anonymous airports and one flight beyond:
   //  b<-a->c->d
   FlightDto expectedFlight1 = createNewFlightBetweenTwoNewAirports(); // a ->b
   FlightDto expectedFlight2 = createNewFlightToNewAirportFrom(expectedFlight1.origAirport); // a ->c
   FlightDto expectedFlight3 = createNewFlightToNewAirportFrom(expectedFlight2.destAirport); // c ->d

   // Test 1: no flights from the destination of Flight 1:
   //    Exercise SUT:
   List flightsFromDest1 = facade.getFlightsByOriginAirport(expectedFlight1.destAirport);
   //    Verify outcome:
   assertTrue("Should not be any flights from dest of flight1", 0==flightsFromDest1.size());

   // Test 2: one flight from the destination of Flight 2:
   //    Exercise SUT:
   List flightsFromDest2 = facade.getFlightsByOriginAirport(expectedFlight2.destAirport);
   //    Verify outcome: # of flights
   if (1==flightsFromDest2.size()) {
      //    Verify attributes of only FlightDto:
      FlightDto onlyFlight = (FlightDto)flightsFromDest2.get(0);
      assertEquals("flight #", expectedFlight1.flightNumber, onlyFlight.flightNumber);
      assertEquals("origAirportId", expectedFlight1.origAirportId, onlyFlight.origAirportId);
      assertEquals("destAirportId", expectedFlight1.destAirportId, onlyFlight.destAirportId);
      assertEquals("origCity", expectedFlight1.origCity, onlyFlight.origCity);
      assertEquals("destCity", expectedFlight1.destCity, onlyFlight.destCity);
      assertEquals("equipType", expectedFlight1.equipmentType, onlyFlight.equipmentType);
   } else {
      fail("should be one flight from airport");
   }

   // Test 3: Two flights from the common origination airport:
   //    Exercise SUT:
   List flightsFromOrig = facade.getFlightsByOriginAirport(expectedFlight1.origAirport);
   //    Verify Outcome:
   //      1st, verify correct number of flights
   if (2==flightsFromDest2.size()) {
      //      2nd, verify 1st flight is the one we expected
      //         (let’s hope they come back in the right order!)
      FlightDto firstFlight = (FlightDto)flightsFromDest2.get(0);
      assertEquals("flight #", expectedFlight1.flightNumber, firstFlight.flightNumber);
      assertEquals("origAirportId", expectedFlight1.origAirportId, firstFlight.origAirportId);
      assertEquals("destAirportId", expectedFlight1.destAirportId, firstFlight.destAirportId);
      assertEquals("origCity", expectedFlight1.origCity, firstFlight.origCity);
      assertEquals("destCity", expectedFlight1.destCity, firstFlight.destCity);
      assertEquals("equipType", expectedFlight1.equipmentType, firstFlight.equipmentType);

      //      3rd, verify 2nd flight is one we expected:
      FlightDto secondFlight = (FlightDto)flightsFromDest2.get(1);
      assertEquals("flight #", expectedFlight2.flightNumber, secondFlight.flightNumber);
      assertEquals("origAirportId", expectedFlight2.origAirportId, secondFlight.origAirportId);
      assertEquals("destAirportId", expectedFlight2.destAirportId, secondFlight.destAirportId);
      assertEquals("origCity", expectedFlight2.origCity, secondFlight.origCity);
      assertEquals("destCity", expectedFlight2.destCity, secondFlight.destCity);
      assertEquals("equipType", expectedFlight2.equipmentType, secondFlight.equipmentType);
   } else {
      fail("should be two flights from airport");
   }
}
```

Exercise CS2: Fixture Setup

Situation

You have just inherited maintenance of the Flight Management System. The good news is that there are automated unit tests. The bad news is that most of the tests look something like these tests.

Instructions:

Examine the code in the handout and determine what code smells you are seeing.

Discussion Questions:

- Which Code Smells are we having?
- What are the underlying root causes?
- Which Patterns can we apply to alleviate them?

Test Code

```
public void testGetFlightsByOriginAirports_TwoOutboundFlights() throws Exception {
     // Set up 2 flights from one airport
     //    first, the origin airport
    BigDecimal calgaryAirportId = facade.createAirport(
               CALGARY_AIRPORT_CODE, CALGARY_AIRPORT_NAME, CALGARY_CITY);
    //    next, the two destination airports
    BigDecimal sanFranAirportId = facade.createAirport(
               SAN_FRAN_AIRPORT_CODE, SAN_FRAN_AIRPORT_NAME, SAN_FRAN_CITY);
    BigDecimal vancouverAirportId = facade.createAirport(
               VANCOUVER_AIRPORT_CODE, VANCOUVER_AIRPORT_NAME, VANCOUVER_CITY);

    //    now, the first flight DTO
    FlightDto expectedFlight1 = new FlightDto();
    expectedFlight1.setOriginAirportId(calgaryAirportId);
    expectedFlight1.setOriginAirportId(CALGARY_CITY);
    expectedFlight1.setDestinationAirportId(sanFranAirportId);
    expectedFlight1.setDestinationCity(SAN_FRAN_CITY);

    // Here’s where we actually create the first flight:
    expectedFlight1.setFlightNumber(facade.createFlight(calgaryAirportId,  sanFranAirportId));

    // And the second flight DTO:
    FlightDto expectedFlight2 = new FlightDto();
    expectedFlight2.setOriginAirportId(calgaryAirportId);
    expectedFlight2.setOriginAirportId(CALGARY_CITY);
    expectedFlight2.setDestinationAirportId(vancouverAirportId);
    expectedFlight2.setDestinationCity(VANCOUVER_CITY);

    // Here’s where we actually create the second flight:
    expectedFlight2.setFlightNumber(facade.createFlight(calgaryAirportId, vancouverAirportId));

    // Exercise the SUT:
    List flightsFromCalgary = facade.getFlightsByOriginAirport(calgaryAirportId);
    assertEquals("Number of flights originating in Calgary", 2, flightsFromCalgary.size());

    // Verify that flights expectedFlight1 and expectedFlight2 are in the
    // list:
    //  etc.
}
```

Exercise BS 2

Instructions:

Symptoms:

Earlier today, you ran all the tests after making some code changes and the tests ran green. You then went to lunch. When you came back you re-ran the tests “just to make sure” before committing your changes. Now, several tests are failing or erroring.

Discussion Questions:

Based on the these symptoms, which Behaviour Smells are we having?
What questions do we need to ask to find out why they are occurring?
What are the underlying root causes?
What can we do about them?

You may also find it useful to ask yourselves these questions:

1. What kind of fixture are theses tests using?
2. Which fixture setup pattern is being used?
3. Why is this causing the failures?

Supporting Material:

The following is the test runner and console output from the test runs. For convenience, the developers have decided to use a logging tool to document what is happening in the various parts of the test in the console output.

Suggested Approach

You might find it helpful to draw a sketch of the Airports and Flights as you read the console output.

Console Output (Earlier today->Green)

```
Testcase Object: testGetFlightsByOriginAirport_OneOutboundFlight
   setUp
      createAirport(YYC)
      createAirport(LAX)
      createAirport(LON)
      createFlight(YYC-LAX)
   method testGetFlightsByOriginAirport_OneOutboundFlight
      getFlightsByOriginAirport(YYC)
   tearDown
.........................................................................................................................................................
Testcase Object: testGetFlightsByDestAirport_OneInboundFlight
   setUp
   method testGetFlightsByDestAirport_OneInboundFlight
      getFlightsByDestAirport(LAX)
   teardown
.........................................................................................................................................................
Testcase Object: testGetFlightsByOriginAirport_TwoOutboundFlights
   setUp
   method testGetFlightsByOriginAirport_TwoOutboundFlights
      createAirport(DIA)
      createFlight(YYC-DIA)
      getFlightsByOriginAirport(YYC)
   tearDown
```

XUnit TestRunner Output(Now à 2 Failures)

```
FlightManagementFacadeTest. testGetFlightsByOriginAirport_OneOutboundFlight():
junit.framework.AssertionFailedError: Flights at origin number of flights:
expected:<1> but was:<2>
        at junit.framework.Assert.fail(Assert.java:47)
        at junit.framework.Assert.failNotEquals(Assert.java:282)
        at junit.framework.Assert.assertEquals(Assert.java:64)
        at junit.framework.Assert.assertEquals(Assert.java:201)
FlightManagementFacadeTest. testGetFlightsByOriginAirport_TwoOutboundFlights():
com.clrstream.flightmgnt.FlightManagementError: Airport already exists: 'DIA'
        at com.clrstream.flightmgnt.FlightManagementFacadeTest.testGetFlightsByOriginAirport_TwoOutboundFlights():
```

Console Output (Now à Failure)

```
Testcase Object: testGetFlightsByOriginAirport_OneOutboundFlight
   setUp
   method testGetFlightsByOriginAirport_OneOutboundFlight
      getFlightsByOriginAirport(YYC)
   teardown
..............................................................................................
Testcase Object: testGetFlightsByDestAirport_OneInboundFlight
   setUp
   method testGetFlightsByDestAirport_OneInboundFlight
      getFlightsByDestAirport(LAX)
   tearDown
..............................................................................................
Testcase Object: testGetFlightsByOriginAirport_TwoOutboundFlights
   setUp
   method testGetFlightsByOriginAirport_TwoOutboundFlights
      createAirport(DIA)
   tearDown
```

Exercise BS 3

Instructions:

Symptoms:

Last week, all the tests ran clean. Since then, 10 new tests have been added (green) but several existing tests are failing.

Discussion Questions:

Based on the these symptoms, which Behaviour Smells are we having?
What questions do we need to ask to find out why they are occurring?
What are the underlying root causes?
What can we do about them?

Supporting Material:

The following is the test runner and console output from the test runs. For convenience, the developers have decided to use a logging tool to document the console output with what is happening in the various parts of the test. Extra lines have been added to delineate one test from the other.

Suggested Approach
You might find it helpful to draw a sketch of the Airports and Flights as you read the console output.

XUnit TestRunner Output

```
FlightManagementFacadeTest.testGetFlightsByOriginAirport_TwoOutboundFlights():
junit.framework.AssertionFailedError: # of flights at origin 2OF:
expected:<2> but was:<4>
        at junit.framework.Assert.fail(Assert.java:47)
        at junit.framework.Assert.failNotEquals(Assert.java:282)
        at junit.framework.Assert.assertEquals(Assert.java:64)
        at junit.framework.Assert.assertEquals(Assert.java:201)
```

Console Output (Historical, only 1 test shown)

```
Testcase Object: testGetFlightsByOriginAirport_TwoOutboundFlights
   setUp
      setupStandardAirportsAndFlights
         createAirport(NOF)
         createAirport(1OF)
         createAirport(2OF)
         createAirport(MIF)
         createFlight(1OF-MIF)
         createFlight(2OF-MIF)
         createFlight(2OF-1OF)
   running testGetFlightsByOriginAirport_TwoOutboundFlights
      getFlightsByOriginAirport(2OF)
   tearDown
      removeStandardAirportsAndFlights
         removeFlight(IOF-MIF)
         removeFlight(2OF-MIF)
         removeFlight(2OF-1OF)
         removeAirport(NOF)
         removeAirport(1OF)
         removeAirport(2OF)
         removeAirport(MIF)
```

Console Output (Current, only 1 test shown)

```
Testcase Object: testGetFlightsByOriginAirport_TwoOutboundFlights
   setUp
      setupStandardAirportsAndFlights
         createAirport(NOF)
         createAirport(1OF)
         createAirport(2OF)
         createAirport(MIF)
         createFlight(IOF-MIF)
         createFlight(2OF-MIF)
         createFlight(2OF-1OF)
      addExtraFlights
         createFlight(2OF-MIF)
         createFlight(2OF-1OF)
   running testGetFlightsByOriginAirport_TwoOutboundFlights
      getFlightsByOriginAirport(2OF)
   tearDown
      removeExtraFlights
         removeAirport(2OF-MIF)
         removeAirport(2OF-1OF)
      removeStandardAirportsAndFlights
         removeFlight(IOF-MIF)
         removeFlight(2OF-MIF)
         removeFlight(2OF-1OF)
         removeAirport(NOF)
         removeAirport(1OF)
         removeAirport(2OF)
         removeAirport(MIF)
```

Exercise BS 4

Symptoms:

You are the first one into the office this morning. You check the builds logs from the overnight build and discover a test failure. When you run the test on your machine it passes. Now what?

Instructions:

Given the following test code and the corresponding TestRunner output, how can you change the test to provide more diagnostic output?

Discussion Questions:

Based on these symptoms, which Behaviour Smells are we having?
What questions do we need to ask to find out why they are occurring?
What are the underlying root causes?
What can we do about them?

Sample Code:

```
[TestFixture]
public class FlightSchedulerTest
{
   [Test]
   public void TestFlightSchedulerWith2ScheduledFlights()
   {
      Flight flight1 = new Flight(“YYC”, “LON”);
      Flight flight2 = new Flight(“LAX”, “YYC”);
      FlightScheduler sut = new FlightScheduler();
      sut.Schedule(flight1);
      sut.Schedule (flight2);
      Assert.AreEqual( flight2, sut.NextFlight(flight1));
      Assert.AreEqual( flight2, sut.top());
    }
}
```

Console Output:

```
TestCase FlightSchedulerTest. testFlightSchedulerWith2ScheduledFlights' failed:
        expected:<flight(LAX->YYC)”>
         but was:NIL
```

Exercise Solutions

Solutions for the exercises are available at:

http://xunitPatternsTutorialSolutions.gerardmeszaros.com