Agile Software Development

Produced by

Eamonn de Leastar (edeleastar@wit.ie)

Waterford Institute *of* Technology
INSTITIÚID TEICNEOLAÍOCHTA PHORT LÁIRGE

First Tests

Assertions

- To check if code is behaving as you expect, use an *assertion*, a simple method call that verifies that something is true.

- E.g the method assertTrue checks that the given boolean condition is true

```
public void assertTrue(boolean condition)
{
  if (!condition)
  {
    abort();
  }
}
```

Using Asserts

- You could use this assert to check all sorts of things, including whether numbers are equal to each other.

```
int a = 2;
//...
assertTrue (a == 2);
```

- To check that two integers are equal, a method that takes two integer parameters might be more useful.

```
public void assertEquals (int a, int b)
{
  assertTrue(a == b);
}
```

- We can now write the first test a little more expressively:

```
int a = 2;

assertEquals (2, a);
```

Planning Tests

- Method to test: A static method designed to find the largest number in a list of numbers.

```
public static int largest (int[] list)
{
...
}
```

- The following tests would seem to make sense:
  - [7, 8, 9] → 9
  - [8, 9, 7] → 9
  - [9, 7, 8] → 9

(supplied test data → expected result)

More Test Data + First Implementation

- Already have this data:

[7, 8, 9] → 9

[8, 9, 7] → 9

[9, 7, 8] → 9

- What about this set:

[7, 9, 8, 9] → 9

[1] → 1

[-9, -8, -7] → -7

```
public static int largest (int[] list)
{
  int index, max = Integer.MAX_VALUE;

  for (index = 0; index < list.length - 1; index++)
  {
    if (list[index] > max)
    {
      max = list[index];
    }
  }
  return max;
}
```

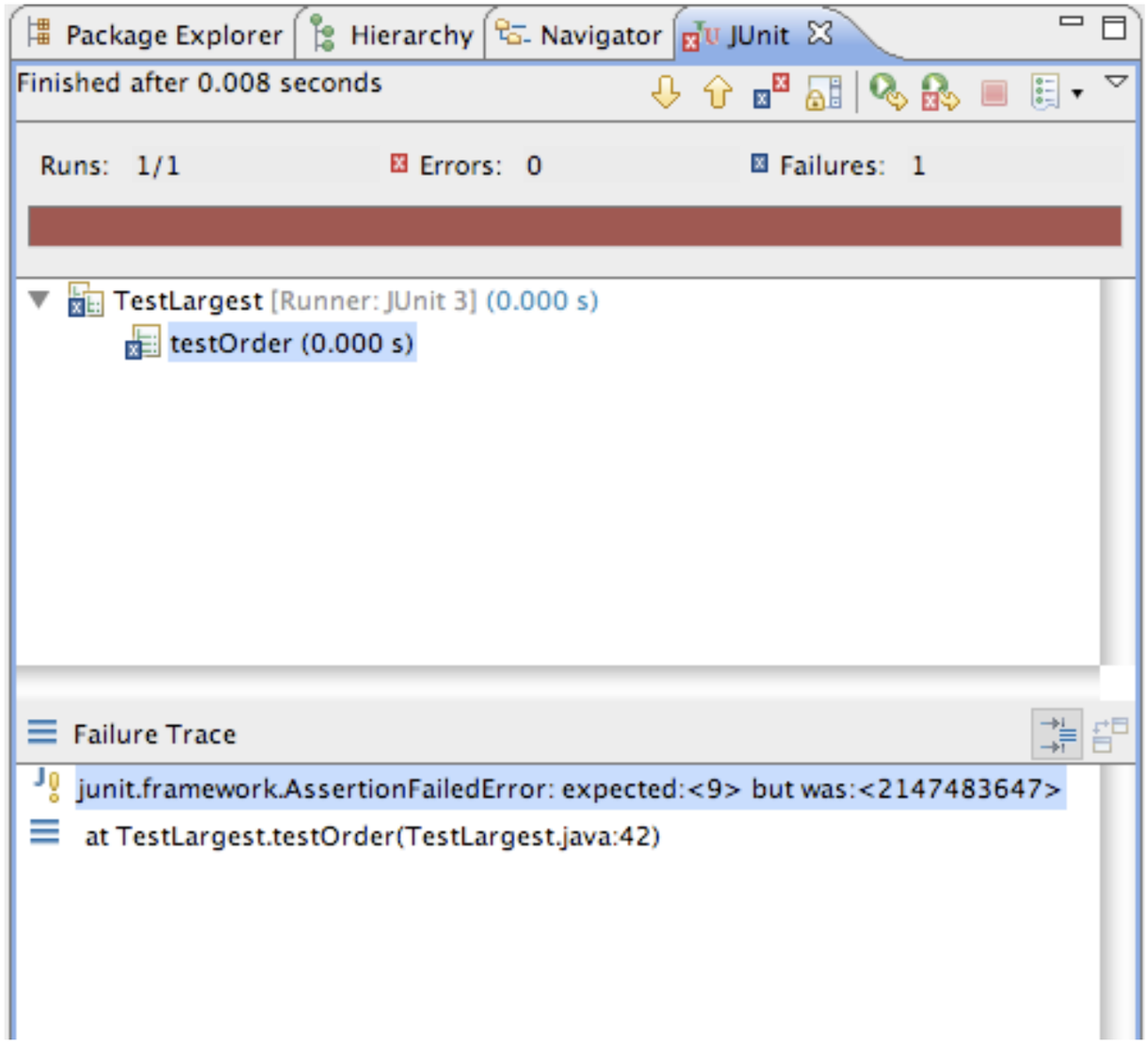
Writing the Test

- This is a TestCase called TestLargest.
- It has one Unit Test - to verify the behaviour of the largest method.

```
import junit.framework.TestCase;

public class TestLargest extends TestCase
{
  public TestLargest (String name)
  {
    super(name);
  }

  public void testOrder ()
  {
    int[] arr = new int[3];
    arr[0] = 8;
    arr[1] = 9;
    arr[2] = 7;
    assertEquals(9, Largest.largest(arr));
  }
}
```

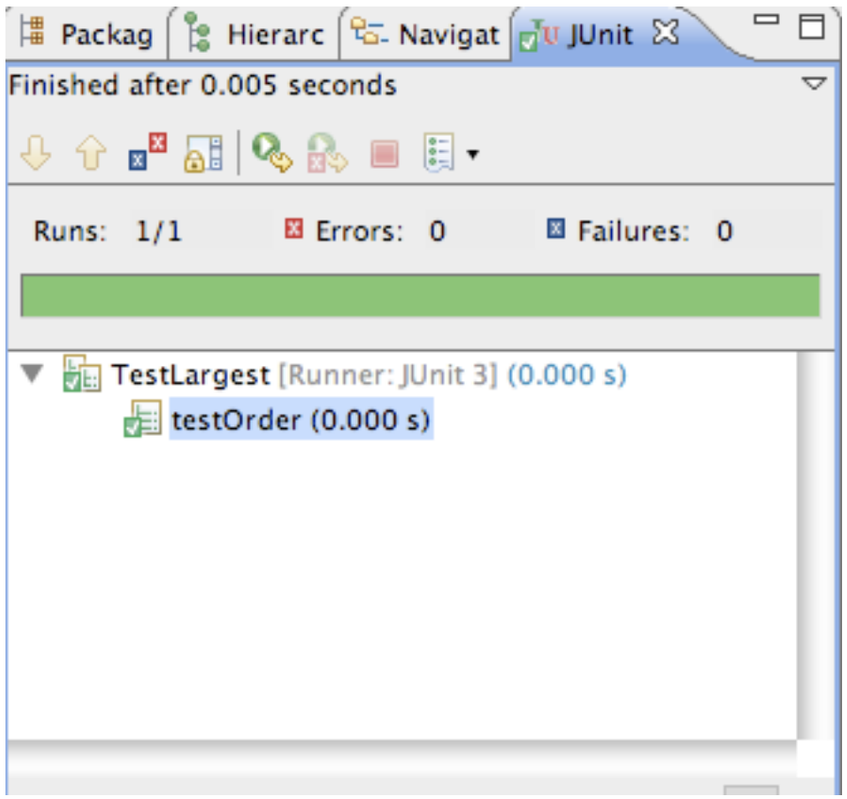
Running the Test

- Why did it return such a huge number instead of our 9
- Where could that very large number have come from?



Bug

- First line should initialize max to zero, not MAX_VALUE.



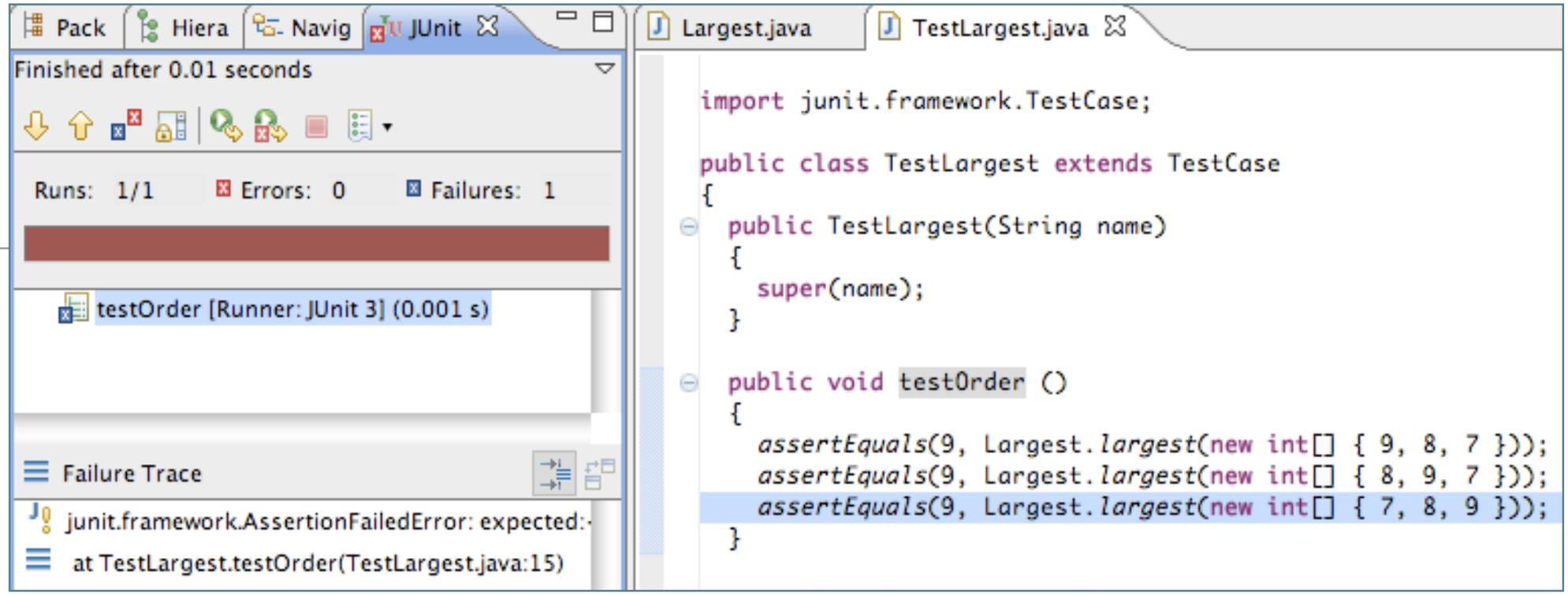

```
public static int largest (int[] list)
{
  //int index, max = Integer.MAX_VALUE;
  int index, max = 0;

  for (index = 0; index < list.length - 1; index++)
  {
    if (list[index] > max)
    {
      max = list[index];
    }
  }
  return max;
}
```

Further Tests

- What happens when the largest number appears in different places in the list - first or last, and somewhere in the middle?
  - Bugs most often show up at the “edges”.
  - In this case, edges occur when the largest number is at the start or end of the array that we pass in.
- Aggregate into a single unit test:

```
public void testOrder ()
{
  assertEquals(9, Largest.largest(new int[] { 9, 8, 7 }));
  assertEquals(9, Largest.largest(new int[] { 8, 9, 7 }));
  assertEquals(9, Largest.largest(new int[] { 7, 8, 9 }));
}
```

Failure + Fix

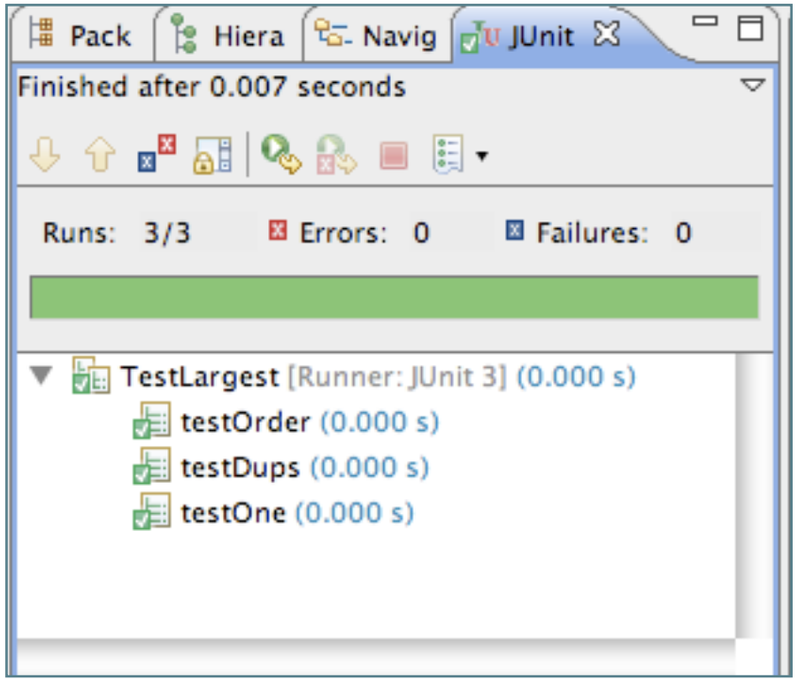



```
public static int largest (int[] list)
{
  int index, max = 0;
  //for (index = 0; index < list.length - 1; index++)
  for (index = 0; index < list.length; index++)
  {
    if (list[index] > max)
    {
      max = list[index];
    }
  }
  return max;
}
```

Further Boundary Conditions

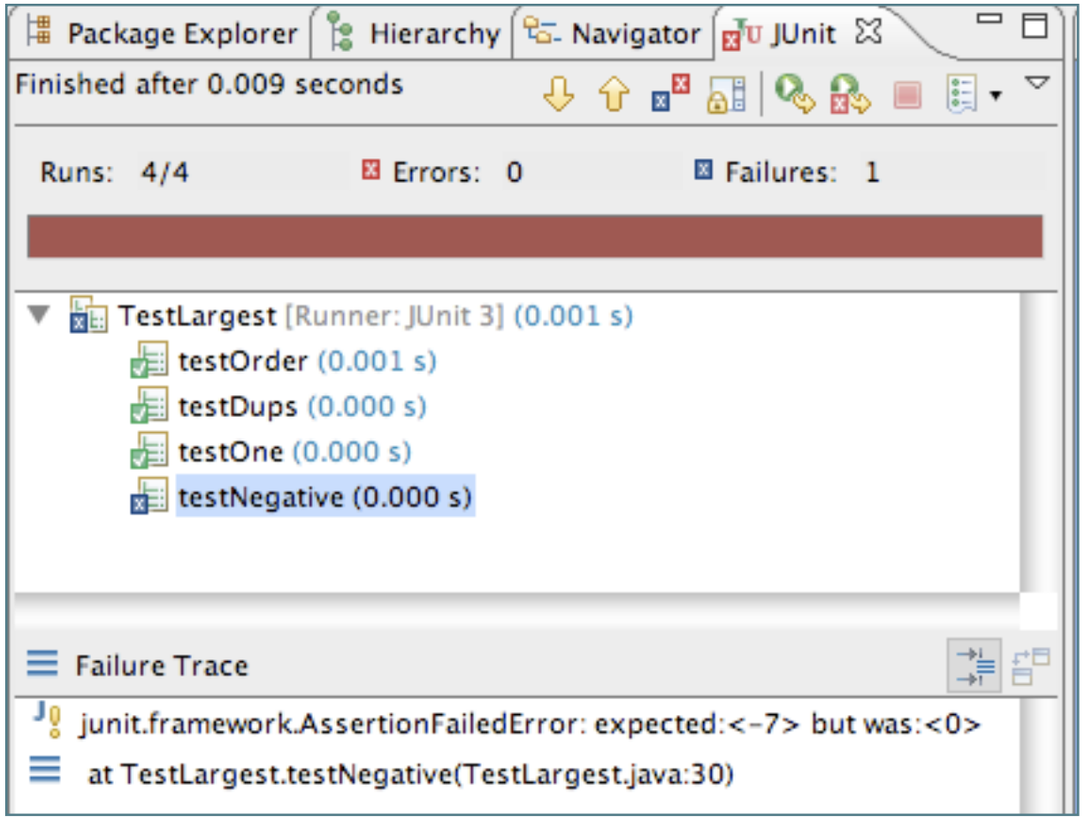
```
public void testDups ()
{
  assertEquals(9, Largest.largest(new int[] { 9, 7, 9, 8 }));
}

public void testOne ()
{
  assertEquals(1, Largest.largest(new int[] { 1 }));
}
```

- Now exercising multiple tests



Failure on testNegative

```
public void testNegative ()
{
    int[] negList = new int[] { -9, -8, -7 };
    assertEquals(-7, Largest.largest(negList));
}
```



fix testNegative

- Choosing 0 to initialize max was a bad idea;
- Should have been MIN VALUE, so as to be less than all negative numbers as well.

```
public static int largest (int[] list)
{
    //int index, max = 0;
    int index, max = Integer.MIN_VALUE;

    for (index = 0; index < list.length; index++)
    {
        if (list[index] > max)
        {
            max = list[index];
        }
     }
     return max;
}
```

Expected Errors?

- If the array is empty, this is considered an error, and an exception should be thrown.

```
public static int largest (int[] list)
{
  int index, max = Integer.MIN_VALUE;

  if (list.length == 0)
  {
    throw new RuntimeException("Empty list");
  }
  for (index = 0; index < list.length; index++)
  {
    if (list[index] > max)
    {
      max = list[index];
    }
  }
  return max;
}
```

```
public void testEmpty ()
{
   try
   {
      Largest.largest(new int[] {});
      fail("Should have thrown an exception");
   }
   catch (RuntimeException e)
   {
      assertTrue(true);
   }
}
```