Testing on Steriods

EECS 4315

www.cse.yorku.ca/course/4315/

Unit Testing

A unit test is designed to test a single unit of code, for example, a method.

Such a test should be automated as much as possible; ideally, it should require no human interaction in order to run, should assess its own results, and notify the programmer only when it fails.

A class that contains unit tests is known as a test case.

The code to be tested is known as the unit under test.

JUnit

JUnit is a Java unit testing framework written by Kent Beck and Erich Gamma.

JUnit is available at `www.junit.org`.

Java Annotations

Annotations provide data about code that is not part of the code itself. Therefore, it is also called metadata.

In its simplest form, an annotation looks like

```
@Deprecated
```

(The annotation type `Deprecated` is part of `java.lang` and, therefore, need not be imported.)

An annotation can include elements and their values:

```
@Test(timeout=1000)
```

(The annotation type `Test` is part of `org.junit` and, therefore, needs to be imported.)

A Test Case

```
import org.junit.Assert;
import org.junit.Test;

public class ...
{
  @Test
  public void ...()
  {
        ...
  }

  @Test
  public void ...()
  {
        ...
  }
}
```

Names of Test Methods

It is good practice to use descriptive names for the test methods. This makes tests more readable when they are looked at later.

Assertions in Test Methods

Each test method should contain (at least) one assertion: an invocation of a method of the **`Assert`** class of the **`org.unit`** package.

Do not confuse these assertions with Java's **`assert`** statement.

Body of Unit Test Method

1. Create some objects.
2. Invoke methods on them.
3. Check the results using a method of the `Assert` class.

Test Case

For each attribute, method and constructor (from simplest to most complex)

1. Study its API.
2. Create unit tests.

Test the attribute **`MIN_VALUE`**

Question

What can we test about the attribute **`MIN_VALUE`**?

Test the attribute `MIN_VALUE`

Question

What can we test about the attribute `MIN_VALUE`?

Answer

Its value.

Test the attribute MIN_VALUE

Question

Can the test

```
@Test
public void testMinValue()
{
  Assert.assertEquals(new Byte(Byte.MIN_VALUE),
                      new Byte((byte) -128));
}
```

be simplified?

Test the attribute MIN_VALUE

Question

Can the test

```
@Test
public void testMinValue()
{
  Assert.assertEquals("-128",
                      Byte.MIN_VALUE + "");
}
```

be simplified?

Test the constructor

Question

What can we test about the constructor?

Test the constructor

Question

What can we test about the constructor?

Answer

That the object is not null.

Test the constructor

Question

What does the following test check?

```
@Test
public void testConstructor()
{
  Byte b = new Byte(0);
  Assert.assertTrue(b.getClass() == Byte.class);
}
```

Test the constructor

Question

What does the following test check?

```
@Test
public void testConstructor()
{
  Byte b = new Byte(0);
  Assert.assertTrue(b.getClass() == Byte.class);
}
```

Answer

The constructor returns an object of type `Byte`.

Test the constructor

Question

Does the constructor conform to the API if it returns an instance of a subclass of the class `Byte`?

Test the constructor

Question

Does the constructor conform to the API if it returns an instance of a subclass of the class **`Byte`**?

Answer

Yes?

Test the constructor

Question

What does the following test check?

```
@Test(expected=NumberFormatException.class)
public void testConstructor()
{
  new Byte(new java.lang.Byte("asfgsdgf"));
}
```

Test the constructor

Question

What does the following test check?

```
@Test(expected=NumberFormatException.class)
public void testConstructor()
{
  new Byte(new java.lang.Byte("asfgsdgf"));
}
```

Answer

The constructor of the class `java.lang`, not the constructor of the class `quiz.Byte`.

Test the constructor

Question

What does the following test check?

```
@Test(expected = NumberFormatException.class)
public void testConstructor()
{
  new Byte(Byte.MIN_VALUE - 1 + "");
}
```

Test the constructor

Question

What does the following test check?

```
@Test(expected = NumberFormatException.class)
public void testConstructor()
{
  new Byte(Byte.MIN_VALUE - 1 + "");
}
```

Answer

Nothing. It fails to compile.

Test the `equals` method

Question

What can we test about the `equals` method?

Test the `equals` method

Question

What can we test about the `equals` method?

Answer

- Whether two `Byte` objects are the same.
- Whether a `Byte` object is equal to itself.
- Whether a `Byte` object is equal to `null`.
- Whether a `Byte` object is equal to an object of another type.

Test the `equals` method

Question

For which `Byte` object(s) do we check equality to itself.

Test the `equals` method

Question

For which `Byte` object(s) do we check equality to itself.

Answer

All (there are only 256 different ones).

Test the `equals` method

Question

Is the following test correct?

```
@Test
public void testEquals()
{
  for (int a = Byte.MIN_VALUE; a <= Byte.MAX_VALUE; a++)
  {
    for (int b = Byte.MIN_VALUE; b <= Byte.MAX_VALUE; b+
    {
          Assert.assertEquals(new Byte((byte) a),
                              new Byte((byte) b));
        }
  }
}
```

Test the `hashCode` method

Question

What can we test about the `hashCode` method?

Test the `hashCode` method

Question

What can we test about the `hashCode` method?

Answer

The value it returns.

Test the hashCode method

Question

What does the following test check?

```
@Test
public void testHashCode()
{
  Assert.assertEquals(new Byte((byte) 1).hashCode()
                      new Byte((byte) 1).hashCode()
}
```

Test the hashCode method

Question

What does the following test check?

```
@Test
public void testHashCode()
{
  Assert.assertEquals(new Byte((byte) 1).hashCode()
                      new Byte((byte) 1).hashCode()
}
```

Answer

Not much (API: The hash code is the value of this object, represented as an int.)

Test the `isEven` method

Question

What can we test about the `isEven` method?

Test the `isEven` method

Question

What can we test about the `isEven` method?

Answer

The value it returns.

Test the `toString` method

Question

What can we test about the `toString` method?

Test the `toString` method

Question

What can we test about the `toString` method?

Answer

The value it returns.

Test the Byte Class

Question

If we run the JUnit test case `ByteTest` and all tests pass, can we conclude that the class `Byte` correctly implements the API?

Test the Byte Class

Question

If we run the JUnit test case `ByteTest` and all tests pass, can we conclude that the class `Byte` correctly implements the API?

Answer

No.

Test the Byte Class

Question

If we run the JUnit test case `ByteTest` and all tests pass, can we conclude that the class `Byte` correctly implements the API?

Answer

No.

Question

Why not?

Test the Byte Class

Question

If we run the JUnit test case `ByteTest` and all tests pass, can we conclude that the class `Byte` correctly implements the API?

Answer

No.

Question

Why not?

Answer

Run the JUnit test case `ByteTest` several times.

Test the Byte Class

Question

How is it possible that the JUnit test case `ByteTest` passes all tests pass in some runs and fails the method `testMinValue` in other runs?

Test the Byte Class

Question

How is it possible that the JUnit test case **`ByteTest`** passes all tests pass in some runs and fails the method **`testMinValue`** in other runs?

Answer

Let's have a look at the code of **`MIN_VALUE`**.

Test the Byte Class

Question

How is it possible that the JUnit test case **`ByteTest`** passes all tests pass in some runs and fails the method **`testMinValue`** in other runs?

Answer

Let's have a look at the code of **`MIN_VALUE`**.

Answer

Because the code of **`MIN_VALUE`** uses randomization.

Randomization

Question

Why are we interested in randomization in our code?

Randomization

Question

Why are we interested in randomization in our code?

Answer

The source code of most computer and video games contains some sort of randomization. This provides games with the ability to surprise players, which is a key factor to their long-term appeal.

Katie Salen and Eric Zimmerman. *Rules of Play: Game Design Fundamentals*. The MIT Press. 2004.

Randomization

Question

Why are we interested in randomization in our code?

Randomization

Question

Why are we interested in randomization in our code?

Answer

Randomization may reduce the expected running time or memory usage.

Randomization

Question

Why are we interested in randomization in our code?

Answer

Randomization may reduce the expected running time or memory usage.

Question

Which algorithms exploit randomization this way?

Randomization

Question

Why are we interested in randomization in our code?

Answer

Randomization may reduce the expected running time or memory usage.

Question

Which algorithms exploit randomization this way?

Answer

- Randomized quicksort.
- Skiplist.
- ...

Randomization

Question

Why are we interested in randomization in our code?

Randomization

Question

Why are we interested in randomization in our code?

Answer

Randomization may allow us to solve problems.

Randomization

Question

Why are we interested in randomization in our code?

Answer

Randomization may allow us to solve problems.

Question

For which problems is randomization exploited this way?

Randomization

Question

Why are we interested in randomization in our code?

Answer

Randomization may allow us to solve problems.

Question

For which problems is randomization exploited this way?

Answer

- Consensus problem (in an asynchronous distributed system in which processes may fail).
- . . .

Nondeterminism

Nondeterministic code is code that, even for the same input, can exhibit different behaviors on different runs, as opposed to deterministic code.

Randomization gives rise to nondeterminism.

Nondeterminism

Nondeterministic code is code that, even for the same input, can exhibit different behaviors on different runs, as opposed to deterministic code.

Randomization gives rise to nondeterminism.

Question

Besides randomization, are there other programming concept that give rise to nondeterminism?

Nondeterminism

Nondeterministic code is code that, even for the same input, can exhibit different behaviors on different runs, as opposed to deterministic code.

Randomization gives rise to nondeterminism.

Question

Besides randomization, are there other programming concept that give rise to nondeterminism?

Answer

Concurrency.

Quiz 1

- When: Monday January 16 during the lab
- Topic: testing