

Forum:[Hack Forums](#)

Thread: [\[quick\] need some help writing unit tests for school](#)

gatermann - November 27, 2016, 6:40 pm

It's a pretty easy exercise, just need some help since I skipped class and left this till last minute.
It's writing unit tests for any implementations of an abstract class and interface.
Thanks

Goodies - November 27, 2016, 8:16 pm

Not much of a java guy to be honest, but does something like this work?

I made this abstract class:

PHP Code

```
<?php
public abstract class Maths {
    abstract double sqrt(double value, int sigfigs);
    abstract double abs(double value);
}
```

And this interface:

PHP Code

```
<?php
public interface Rounder {
    double round(double value, int places);
}
```

And this class called "Stuff" to use both of them.

PHP Code

```
<?php
public class Stuff extends Maths implements Rounder {
    public double round(double value, int places) {
        // truncates a java double into `places` places after the decimal
        long pow = (long)Math.pow(10, places);
        return (long)Math.round(value * pow) / (double)pow;
    }
}
```

@Override

```
double abs(double value) {
    // return the absolute value of a number
    return (value >= 0 ? value : -value);
}
```

@Override

```
double sqrt(final double value, final int sigfigs) {
```

```
// a simple implementation of the babylonian method of determining the square root of a number
    double prev = -1, curr = sigfigs * 100;
    while(round(prev, sigfigs) != round(curr, sigfigs)) {
        prev = curr;
```

```

        curr = (prev + (value / prev)) / 2;
    }
    return round(curr, sigfigs);
}
}

```

And here's an example unit test you can write:

PHP Code

```
<?php
```

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

```

public class Tester {
    @Test
    public void checkRounderRound(){
        final double
            input1 = 123.456, target1 = 123.46, epsilon1 = 0.0,
            input2 = 999999.999999, target2 = 1000000.0, epsilon2 = 0.0,
            input3 = 73737.3737, target3 = 73737.37, epsilon3 = 0.0;

        final int
            sigfigs1 = 2,
            sigfigs2 = 5,
            sigfigs3 = 2;

        Rounder rounder = new Stuff(); // works fine because it implements rounder
        Assert.assertEquals(rounder.round(input1, sigfigs1), target1, epsilon1);
        Assert.assertEquals(rounder.round(input2, sigfigs2), target2, epsilon2);
        Assert.assertEquals(rounder.round(input3, sigfigs3), target3, epsilon3);
    }

    @Test
    public void checkMathsSqrt(){
        final double
            input1 = 100.0, target1 = 10.0, epsilon1 = 0.0,
            input2 = 125348, target2 = 354.04519485512014, epsilon2 = 0.0,
            input3 = 1, target3 = 1, epsilon3 = 0.0;

        final int
            sigfigs1 = 5,
            sigfigs2 = 15,
            sigfigs3 = 5;

        Maths maths = new Stuff(); // works fine because it extends Maths
        Assert.assertEquals(maths.sqrt(input1, sigfigs1), target1, epsilon1);
        Assert.assertEquals(maths.sqrt(input2, sigfigs2), target2, epsilon2);
        Assert.assertEquals(maths.sqrt(input3, sigfigs3), target3, epsilon3);
    }

    @Test
    public void checkMathsAbs(){
        final double
            input1 = 100, target1 = 100, epsilon1 = 0.0,

```

```
input2 = -12345, target2 = 12345, epsilon2 = 0.0,  
input3 = 0, target3 = 0, epsilon3 = 0.0;
```

```
Maths maths = new Stuff(); // works fine because it extends Maths  
Assert.assertEquals(maths.abs(input1), target1, epsilon1);  
Assert.assertEquals(maths.abs(input2), target2, epsilon2);  
Assert.assertEquals(maths.abs(input3), target3, epsilon3);  
}  
}
```

gatermann - November 27, 2016, 9:28 pm

(Nov 27, 2016 09:16 PM)Goodies Wrote: [\[CITE\]](#) Not much of a java guy to be honest, but does something like this work?

I made this abstract class:

PHP Code

```
<?php  
public abstract class Maths {  
    abstract double sqrt(double value, int sigfigs);  
    abstract double abs(double value);  
}
```

And this interface:

PHP Code

```
<?php  
public interface Rounder {  
    double round(double value, int places);  
}
```

And this class called "Stuff" to use both of them.

PHP Code

```
<?php  
public class Stuff extends Maths implements Rounder {  
    public double round(double value, int places) {  
        // truncates a java double into `places` places after the decimal  
        long pow = (long)Math.pow(10, places);  
        return (long)Math.round(value * pow) / (double)pow;  
    }  
  
    @Override  
    double abs(double value) {  
        // return the absolute value of a number  
        return (value >= 0 ? value : -value);  
    }  
  
    @Override  
    double sqrt(final double value, final int sigfigs) {  
  
// a simple implementation of the babylonian method of determining the square root of a number  
        double prev = -1, curr = sigfigs * 100;  
        while(round(prev, sigfigs) != round(curr, sigfigs)) {  
            prev = curr;
```

```

        curr = (prev + (value / prev)) / 2;
    }
    return round(curr, sigfigs);
}
}

```

And here's an example unit test you can write:

PHP Code

```
<?php
```

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

```

public class Tester {
    @Test
    public void checkRounderRound(){
        final double
            input1 = 123.456, target1 = 123.46, epsilon1 = 0.0,
            input2 = 999999.999999, target2 = 1000000.0, epsilon2 = 0.0,
            input3 = 73737.3737, target3 = 73737.37, epsilon3 = 0.0;

        final int
            sigfigs1 = 2,
            sigfigs2 = 5,
            sigfigs3 = 2;

        Rounder rounder = new Stuff(); // works fine because it implements rounder
        Assert.assertEquals(rounder.round(input1, sigfigs1), target1, epsilon1);
        Assert.assertEquals(rounder.round(input2, sigfigs2), target2, epsilon2);
        Assert.assertEquals(rounder.round(input3, sigfigs3), target3, epsilon3);
    }

    @Test
    public void checkMathsSqrt(){
        final double
            input1 = 100.0, target1 = 10.0, epsilon1 = 0.0,
            input2 = 125348, target2 = 354.04519485512014, epsilon2 = 0.0,
            input3 = 1, target3 = 1, epsilon3 = 0.0;

        final int
            sigfigs1 = 5,
            sigfigs2 = 15,
            sigfigs3 = 5;

        Maths maths = new Stuff(); // works fine because it extends Maths
        Assert.assertEquals(maths.sqrt(input1, sigfigs1), target1, epsilon1);
        Assert.assertEquals(maths.sqrt(input2, sigfigs2), target2, epsilon2);
        Assert.assertEquals(maths.sqrt(input3, sigfigs3), target3, epsilon3);
    }

    @Test
    public void checkMathsAbs(){
        final double
            input1 = 100, target1 = 100, epsilon1 = 0.0,

```

```
input2 = -12345, target2 = 12345, epsilon2 = 0.0,  
input3 = 0, target3 = 0, epsilon3 = 0.0;
```

```
Maths maths = new Stuff(); // works fine because it extends Maths  
Assert.assertEquals(maths.abs(input1), target1, epsilon1);  
Assert.assertEquals(maths.abs(input2), target2, epsilon2);  
Assert.assertEquals(maths.abs(input3), target3, epsilon3);  
}  
}
```

Omg thanks this was a good enough example to give me an understanding <3
I'm done thanks to you :)