Java Code Geeks
JAVA 2 JAVA DEVELOPERS RESOURCE CENTER

ABOUT VINOD KUMAR KASHYAP

Vinod is Sun Certified and love to work in Java and related technologies. Having more than 10 years of experience, he had developed software's including technologies like Java, Hibernate, Struts, Spring, HTML 5, jQuery, CSS, Web Services, MongoDB, AngularJS. He is also a JUG Leader of Chandigarh Java User Group.

JUnit Test Constructor Example

Posted by: Vinod Kumar Kashyap in junit April 12th, 2017

In JUnit Test Constructor Example, we shall show you the process of testing the constructor of a class. It is as simple as we are testing other methods. Since the constructor is called before any other methods of the class we need to see the relevance of testing it.

Constructors are used for initializing the class and then do some processing on it. We will examine the different ways of testing the constructor of a class.

1. Introduction

JUnit provides lot of methods to test our cases. We can test the constructor of a class using the same techniques we have used in our previous examples. Sometimes we need to initialize the objects and we do them in a constructor. In JUnit we can also do same using the

```
@Before
```

method. But what will be the difference between them?

They are following the same scenario as both all called before any test case is executed.

Tip
Constructor and @Before both are same. But constructor has a drawback that it will not give proper error message, whereas @Before method will give proper error message in case of failing.

2. Technologies Used

We will be using following technologies in out below example.

- **Java 1.8** – language to code
- **JUnit 4.12** – testing framework
- **Maven** – build tool
- **Eclipse** – IDE for code

3. Project Setup

Let’s start creating our project.

Tip
You may skip project creation and jump directly to the **beginning of the example** below.
Open Eclipse. Click **File -> New -> Maven Project**. You will be taken to below screen. Fill in the details and click on **Next** button.

Figure 1: JUnit Test Constructor Example Setup 1

On this screen, fill in all the details and click on the **Finish** button.

Figure 2: JUnit Test Constructor Example Setup 2

This will create a blank Maven project. You are now ready to start coding the example.

4. JUnit Test Constructor Example

Let's start creating classes our example.

First of all, open

```
pom.xml
```

file and copy and paste below line into it.

pom.xml

```
<dependencies>
    <dependency>
        <groupId>junit</groupId>
        <artifactId>junit</artifactId>
        <version>4.12</version>
    </dependency>
</dependencies>

<properties>
    <maven.compiler.source>1.8</maven.compiler.source>
    <maven.compiler.target>1.8</maven.compiler.target>
</properties>
```

Here, we have ask Maven to pull **JUnit 4.12** dependency and use **Java 1.8** for compiling our example.

4.1 Java Classes

Start by create a model class that will help in testing our example. Here we are creating a simple class called

```
Calculator
```

.

Calculator.java

```
    private int numA;
    private int numB;

    Calculator(int numA,int numB){
        this.numA = numA;
        this.numB = numB;
        if(numA<0 || numB<0){
            throw new IllegalArgumentException();
        }
    }

    public int sum(){
        return this.numA + this.numB;
    }
}
```

As we can see, this class 2 variables

```
numA
```

and

```
numB
```

. One constructor which is also throwing

```
IllegalArgumentException()
```

.

We will test both scenarios where we will pass positive as well as negative values. We will also see **how we can handle the exceptions** in test class so that test cases don't fail.

Now, we will create a test class that will test our constructor.

CalculatorTest.java

```
01 package junittestconstructor;
02 
03 import static org.hamcrest.CoreMatchers.is;
04 import static org.junit.Assert.assertThat;
05 
06 import org.junit.Test;
07 
08 public class CalculatorTest {
09 
10     private Calculator c;
11 
12     @Test
13     public void constructorTest(){
14         c = new Calculator(4, 5);
15         assertThat(9, is(c.sum()));
16     }
17 
18     @Test(expected = IllegalArgumentException.class)
19     public void constructorExceptionTest(){
20         c = new Calculator(-4, 5);
21     }
22 
23 }
```

Here we can see that at **line no 14** we are testing with the positive values and at **line no 20** we are testing negative scenario.

See how we have handled the exception. If we do not handle it, this method will fail. To comply standards and to test our case to pass, we need to catch the exception. We have done by passing the arguments to the

```
@Test
```

annotation. See **line no 18**.

Output
Here is the output seen in Eclipse JUnit window.