



[ANDROID](#) | 
 [CORE JAVA](#) | 
 [DESKTOP JAVA](#) | 
 [ENTERPRISE JAVA](#) | 
 [JAVA BASICS](#) | 
 [JVM LANGUAGES](#) | 
 [SOFTWARE DEVELOPMENT](#) | 
 [DEVOPS](#)

△ Home » Core Java » junit » JUnit Testcase Example

## 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 Testcase Example

□ Posted by: Vinod Kumar Kashyap □ in junit □ April 6th, 2017

In this example, we shall show users how to use JUnit Testcase. JUnit Testcase example will follow you to the scenarios of how we can use and test our methods with the help of the JUnit Testcase.

As a regular reader, you are already familiar with the JUnit and its usage. If you are new to it you are recommend to visit JUnit Series on Java Code Geeks.

## Want to be a JUnit Master ?

Subscribe to our newsletter and download the JUnit Programming Cookbook [right now!](#)

In order to help you master unit testing with JUnit, we have compiled a kick-ass guide with all the major JUnit features and use cases! Besides studying them online you may download the eBook in PDF format!

### Email address:

[Sign up](#)

Before this article, I have shown many different uses of JUnit and its many different properties. But in this example, we will show something different which we haven't done in previous articles.

## 1. Introduction

JUnit provides a class known as

. This class will help us to run all our test cases. In previous articles, we have used the

annotation on all our test cases. But, here we will not use any annotation and see how we can test the methods with the help of the

## CARRER OPPORTUNITIES



Get the latest jobs to your inbox





Send me jobs!

Job Search



```
TestCase
```

class and then proceed further for testing. Let's start creating a project.

## 2. Technologies Used

Following technologies are used in this example.

- **Java 8:** Language for example
- **JUnit 4.12:** Testing Framework
- **Maven:** Dependency and build tool
- **Eclipse:** IDE for coding

## 3. Project Setup

### Tip

You may skip project creation and jump directly to the **beginning of the example** below.

We will be using the Eclipse and Maven for our project. Click **File -> New -> Maven Project** On the first screen simply check the check mark corresponding to the "Create a simple project" and click on **Next** button.

### NEWSLETTER

**182,134** insiders are already enjoying weekly updates and complimentary whitepapers!

**Join them now** to gain **EXCLUSIVE ACCESS** to the latest news in the Java world as well as insights about Android, Scala, Groovy and other related technologies.

### Email address:

☒ Receive Java & Developer job alerts in your Area from our partners over at ZipRecruiter

[Sign up](#)

### JOIN US



With **1,240,600** unique visitors and **500** authors placed among the top related sites at Google, we are constantly being looked out for and encourage you. So If you have unique and interesting content then you can check out our **JCG** partners program. You can be a **guest writer** for Java Code Geek and showcase your writing skills!

Figure 1: JUnit Testcase Example Setup 1

Fill in all the details and click on **Finish** button.

## JUnit Testcase Example Setup 2

Now our blank project is ready.

## 4. JUnit Testcase Example

We will be doing 2 things here. First configure the JUnit by adding dependency in the

```
pom.xml
```

and secondly, creating java classes for testing.

### 4.1 xml configuration

Firstly, put the following code in the

```
pom.xml
```

*pom.xml*

```
01 <dependencies>
02   <dependency>
03     <groupId>junit</groupId>
04     <artifactId>junit</artifactId>
05     <version>4.12</version>
06   </dependency>
07 </dependencies>
08
09 <properties>
10   <maven.compiler.source>1.8</maven.compiler.source>
11   <maven.compiler.target>1.8</maven.compiler.target>
12 </properties>
```

Here we are simply telling maven to pull dependency of

```
JUnit 4.12
```

and use

for complete.

## 4.2 Java classes

Next step is to create a model class which will help in testing.

ListTest.java

```
01 package junittestcase;
02
03 import java.util.ArrayList;
04 import java.util.List;
05
06 public class ListTest {
07
08     private List lstFruits = new ArrayList();
09
10     public void add(String fruit) {
11         lstFruits.add(fruit);
12     }
13
14     public void remove(String fruit) {
15         lstFruits.remove(fruit);
16     }
17
18     public int size() {
19         return lstFruits.size();
20     }
21
22     public void removeAll() {
23         lstFruits.clear();
24     }
25 }
```

Now we are ready to write our main class that will actually run our test cases. We need to follow some points so that we run our test cases:

- We need to extend the

TestCase

class of JUnit

- setUp()

and

tearDown()

methods, which are optional and runs before and after every test case.

- Every test case should be public and name should begin with **test**. They should not take any argument.
- Test case should not return any value.

MyTestCase.java

```
01 package junittestcase;
02
03 import junit.framework.TestCase;
04
05 public class MyTestCase extends TestCase {
06
07     protected ListTest lstTest = new ListTest();
08
09     protected void setUp() {
10         lstTest.add("Apple");
11         lstTest.add("Orange");
12         lstTest.add("Grapes");
13     }
14
15     public void testSize() {
16         assertEquals("Checking size of List", 3, lstTest.size());
17     }
18
19     public void testAdd() {
20         lstTest.add("Banana");
21         assertEquals("Adding 1 more fruit to list", 4, lstTest.size());
22     }
23
24     public void testRemove() {
25         lstTest.remove("Orange");
26         assertEquals("Removing 1 fruit from list", 2, lstTest.size());
27     }
28
29     protected void tearDown() {
30         lstTest.removeAll();
31     }
32 }
```

Let's analyze all the details line by line of this class.

At line no 5, we have extended the

class.

Line no 9, specifies a

```
setUp()
```

method that will run before any test method.

Line 15,19,24 are our test cases.

Line no 29 specifies the

```
tearDown()
```

method that will be called after each test case.

It is to be noted that,

```
setUp()
```

and

```
tearDown()
```

are not mandatory at all. We have used here only to illustrate the usage of them.

Output

Figure 3: JUnit Testcase Example Output

## 5. Conclusion

We have learned a way to test our test cases with the help of the

```
TestCase
```

class of JUnit. Now a days we are using annotations instead of the

```
TestCase
```

class. But through JUnit Testcase example we have learned a new way to test.

## 6. Download the Eclipse Project

This is a JUnit Testcase Example.

### Download

You can download the full source code of this example here: **JUnitTestCaseExample**