



[ANDROID](#) | 
 [JAVA](#) | 
 [JVM LANGUAGES](#) | 
 [SOFTWARE DEVELOPMENT](#) | 
 [AGILE](#) | 
 [CAREER](#) | 
 [COMMUNICATIONS](#) | 
 [DEVOPS](#) | 
 [META JCG](#)

△ Home » Java » Core Java » Parameterized JUnit tests with JUnitParams

## ABOUT RAFAL BOROWIEC



Software developer, Team Leader, Agile practitioner, occasional blogger, lecturer. Open Source enthusiast, quality oriented and open-minded.



## Parameterized JUnit tests with JUnitParams

□ Posted by: Rafal Borowiec □ in Core Java □ December 22nd, 2013

Parameterized unit tests are used to test the same code under different conditions. Thanks to parameterized unit tests we can set up a test method that retrieves data from some data source. This data source can be a collection of test data objects, external file or maybe even a database. The general idea is to make it easy to test different conditions with the same unit test method, which will limit the source code we need to write and makes the test code more robust. We can call these tests data-driven unit tests.

The best way to achieve data-driven unit tests in JUnit is to use a JUnit's custom runner –

Parameterized



or JUnitParams'

JUnitParamsRunner

. Using JUnit's approach may work in many cases, but the latter seems to be more easy to use and more powerfull.

## Basic example

In our example, a poker dice, we need to calculate the score of a full house. As in card poker, a Full House is a roll where you have both a 3 of a kind, and a pair. For the sake of simplicity, the score is a sum of all dice in a roll. So let's see the code:

```

01 class FullHouse implements Scoreable {
02
03     @Override
04     public Score getScore(Collection dice) {
05         Score pairScore = Scorables.pair().getScore(dice);
06         Score threeOfAKindScore = Scorables.threeOfAKind().getScore(pairScore.getRemainder());
07         if (bothAreGreaterThanZero(pairScore.getValue(), threeOfAKindScore.getValue())) {
08             return new Score(pairScore.getValue() + threeOfAKindScore.getValue()); // no remainder
09         }
10         return new Score(0, dice);
11     }
12
13     private boolean bothAreGreaterThanZero(int value1, int value2) {
14         return value1 > 0 && value2 > 0;
15     }
16 }

```

I would like to be sure that the roll is properly scored (of course I have unit tests for both Pair and ThreeOfAKind already). So I would like to test the following conditions:

- Score is 11 for: 1, 1, 3, 3, 3
- Score is 8 for: 2, 2, 2, 1, 1
- Score is 0 for: 2, 3, 4, 1, 1

## CAREER OPPORTUNITIES



Get the latest jobs to your ir

Enter your name

Enter your email

developer

Enter city, state or zip

Send me jobs!

Job Search



```

01 @RunWith(Parameterized.class)
02 public class FullHouseTest {
03
04     private Collection rolled;
05     private int score;
06
07     public FullHouseTest(Collection rolled, int score) {
08         this.rolled = rolled;
09         this.score = score;
10     }
11
12     @Test
13     public void fullHouse() {
14         assertThat(new FullHouse().getScore(rolled).getValue()).isEqualTo(score);
15     }
16
17     @Parameterized.Parameters
18     public static Iterable data() {
19         return Arrays.asList(
20             new Object[][] {
21                 {roll(1, 1, 3, 3, 3), score(11)},
22                 {roll(2, 2, 2, 1, 1), score(8)},
23                 {roll(2, 3, 4, 1, 1), score(0)},
24                 {roll(5, 5, 5, 5, 5), score(25)}
25             }
26         );
27     }
28
29     private static int score(int score) {
30         return score;
31     }
32 }

```

And the other one, with JUnitParams:

```

01 @RunWith(JUnitParamsRunner.class)
02 public class FullHouseTest {
03
04     @Test
05     @Parameters
06     public void fullHouse(Collection rolled, int score) {
07         assertThat(new FullHouse().getScore(rolled).getValue()).isEqualTo(score);
08     }
09
10     public Object[] parametersForFullHouse() {
11         return $(
12             $(roll(1, 1, 3, 3, 3), score(11)),
13             $(roll(2, 2, 2, 1, 1), score(8)),
14             $(roll(2, 3, 4, 1, 1), score(0)),
15             $(roll(5, 5, 5, 5, 5), score(25))
16         );
17     }
18
19     private static int score(int score) {
20         return score;
21     }
22 }

```

At first glance, both look very similar. And that's true. So what are the differences between JUnit

Parameterized

(1) and JUnitParams (2)? The most important one is the way of passing the parameters, so in fact the architecture of the solution. In (1) parameters are passed in constructor whereas in (2) parameters are passed directly to test method. Should I care? Yes. One of the reason is, in (2), we can have multiple parameterized tests methods with different data for each of the method, like in the below example:

```

01 @RunWith(JUnitParamsRunner.class)
02 public class NumberOfAKindTest {
03
04     @Test
05     @Parameters
06     public void pair(Collection rolled, int[] expected, int score) {
07         NumberOfAKind sut = new NumberOfAKind(2);
08         doTest(rolled, expected, score, sut);
09     }
10
11     @Test
12     @Parameters
13     public void threeOfAKind(Collection rolled, int[] expected, int score) {
14         NumberOfAKind sut = new NumberOfAKind(3);
15         doTest(rolled, expected, score, sut);
16     }
17
18     public Object[] parametersForPair() {
19         return $(
20             $(roll(1, 1, 1, 2, 3), hand(1, 1), score(2)),
21             $(roll(2, 1, 1, 1, 1), hand(1, 1), score(2)),
22             $(roll(2, 3, 4, 1, 1), hand(1, 1), score(2)),
23             $(roll(2, 3, 5, 5, 5), hand(5, 5), score(10)),
24             $(roll(2, 1, 5, 4, 3), null, score(0))
25         );
26     }
27
28     public Object[] parametersForThreeOfAKind() {

```

**179,260** insiders are already enjoying weekly updates and complimentary whitepapers!

**Join them now** to gain **EX**

**access** to the latest news in the Java world, as well as insights about Android, Scala, Groovy and other related technologies.

**Email address:**

Your 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 related sites and constantly being looked out for, we encourage you to be a **guest writer** for Java Code Geek your writing skills!

```

33         $(roll(2, 3, 5, 5, 5), hand(5, 5, 5), score(15)),
34         $(roll(2, 5, 5, 5, 6), hand(5, 5, 5), score(15)),
35         $(roll(2, 2, 5, 5, 3), null, score(0))
36     );
37 }
38
39 private static int[] hand(int... dice) {
40     return dice;
41 }
42
43 private static int score(int score) {
44     return score;
45 }
46
47 }

```

In simpler examples, parameters can be defined as a String array directly in @Parameters annotation via value method. We can also extract the data to an external class and have our tests more clean and readable. The complete test for

NumberOfAKind

reads as following:

```

01 @RunWith(JUnitParamsRunner.class)
02 public class NumberOfAKindTest {
03
04     @Test
05     @Parameters(source = NumberOfAKindProvider.class, method = "providePair")
06     public void pair(Collection rolled, int[] expected, int score) {
07         NumberOfAKind sut = new NumberOfAKind(2);
08         doTest(rolled, expected, score, sut);
09     }
10
11     @Test
12     @Parameters(source = NumberOfAKindProvider.class, method = "provideThreeOfAKind")
13     public void threeOfAKind(Collection rolled, int[] expected, int score) {
14         NumberOfAKind sut = new NumberOfAKind(3);
15         doTest(rolled, expected, score, sut);
16     }
17
18     @Test
19     @Parameters(source = NumberOfAKindProvider.class, method = "provideFourOfAKind")
20     public void fourOfAKind(Collection rolled, int[] expected, int score) {
21         NumberOfAKind sut = new NumberOfAKind(4);
22         doTest(rolled, expected, score, sut);
23     }
24
25     @Test
26     @Parameters(source = NumberOfAKindProvider.class, method = "provideFiveOfAKind")
27     public void fiveOfAKind(Collection rolled, int[] expected, int score) {
28         NumberOfAKind sut = new NumberOfAKind(5);
29         doTest(rolled, expected, score, sut);
30     }
31
32     private void doTest(Collection rolled, int[] expected, int score, NumberOfAKind sut) {
33         Collection consecutiveDice = sut.getConsecutiveDice(rolled);
34
35         assertDiceContainsValues(consecutiveDice, expected);
36         assertThat(sut.getScore(rolled).getValue().isEqualTo(score);
37     }
38
39     private void assertDiceContainsValues(Collection dice, int[] expected) {
40         Collection values = toInts(dice);
41         if (expected == null) {
42             assertThat(values).isEmpty();
43             return;
44         }
45         for (int i = 0; i < expected.length; i++) {
46             assertThat(values).hasSize(expected.length).contains(expected[i]);
47         }
48     }
49
50     private Collection toInts(Collection dice) {
51         return Collections2.transform(dice, new Function() {
52             @Override
53             public Integer apply(Dice input) {
54                 return input.getValue();
55             }
56         });
57     }
58
59 }

```

Each method specifies a provider class and the provider's method name. Look at the provider below:

```

01 public class NumberOfAKindProvider {
02
03     public static Object[] providePair() {
04         return $(
05             $(roll(1, 1, 1, 2, 3), hand(1, 1), score(2)),
06             $(roll(2, 1, 1, 1, 1), hand(1, 1), score(2)),
07             $(roll(2, 3, 4, 1, 1), hand(1, 1), score(2)),
08             $(roll(2, 3, 5, 5, 5), hand(5, 5), score(10)),
09             $(roll(2, 1, 5, 4, 3), null, score(0))
10         );
11     }
12 }

```

```
17         $(roll(2, 3, 1, 1, 1), hand(1, 1, 1), score(3)),
18         $(roll(2, 3, 5, 5, 5), hand(5, 5, 5), score(15)),
19         $(roll(2, 5, 5, 5, 6), hand(5, 5, 5), score(15)),
20         $(roll(2, 2, 5, 5, 3), null, score(0))
21     );
22 }
23
24 public static Object[] provideFourOfAKind() {
25     return $(
26         $(roll(1, 1, 1, 1, 3), hand(1, 1, 1, 1), score(4)),
27         $(roll(2, 1, 1, 1, 1), hand(1, 1, 1, 1), score(4)),
28         $(roll(2, 5, 5, 5, 5), hand(5, 5, 5, 5), score(20)),
29         $(roll(2, 3, 4, 5, 5), null, score(0))
30     );
31 }
32
33 public static Object[] provideFiveOfAKind() {
34     return $(
35         $(roll(1, 1, 1, 1, 1), hand(1, 1, 1, 1, 1), score(5)),
36         $(roll(6, 6, 6, 6, 6), hand(6, 6, 6, 6, 6), score(30)),
37         $(roll(6, 6, 6, 6), null, score(0)),
38         $(roll(2, 3, 4, 6, 6), null, score(0))
39     );
40 }
41
42 private static int[] hand(int... dice) {
43     return dice;
44 }
45
46 private static int score(int score) {
47     return score;
48 }
49 }
```

## Summary

To me

JUnitParams

is a better solution for writing good data-driven unit tests. But what is presented above, is not everything the library has to offer to a developer. There are more features. Params can be passed as a CSV string, we can mix both parameterized and non-parameterized tests just to mention a few.

Please visit the project's website to find out more about this library: <https://code.google.com/p/junitparams>

**Reference:** Parameterized JUnit tests with JUnitParams from our JCG partner Rafal Borowiec at the Codeleak.pl blog.

Tagged with:

JUNIT

TESTING

## Do you want to know how to develop your skillset to become a **Java Rockstar**?

Subscribe to our newsletter to start Rocking right now!

To get you started we give you our best selling eBooks for **FREE!**

1. JPA Mini Book
2. JVM Troubleshooting Guide
3. JUnit Tutorial for Unit Testing
4. Java Annotations Tutorial
5. Java Interview Questions
6. Spring Interview Questions
7. Android UI Design

and many more ....

**Email address:**

Your email address

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

Sign up

## 2 COMMENTS



Denilson

December 26th, 2013 at 6:15 pm

Have you tried DDTUnit? At first the separate XML for the data may be intimidating, but the Java code is much simpler.

Reply



piotrek

November 22nd, 2014 at 3:22 pm

try [zohhak.googlecode.com](http://zohhak.googlecode.com). It lets you write parameters in your java code in a really clean manner, without any providers or external xml files.

Reply

## LEAVE A REPLY

Your email address will not be published. Required fields are marked \*

Name \*

Email \*

Website



Sign me up for the newsletter!

Receive Email Notifications?

no, do not subscribe



instantly

Or, you can subscribe without commenting.

Post Comment

## KNOWLEDGE BASE

[Courses](#)

[Examples](#)

[Resources](#)

[Tutorials](#)

[Whitepapers](#)

## PARTNERS

[Mkyong](#)

## HALL OF FAME

["Android Full Application Tutorial" series](#)

[11 Online Learning websites that you should check out](#)

[Advantages and Disadvantages of Cloud Computing – Cloud computing pros and cons](#)

[Android Google Maps Tutorial](#)

[Android JSON Parsing with Gson Tutorial](#)

## ABOUT JAVA CODE GEEKS

JCGs (Java Code Geeks) is an independent online community focused on creating the ultimate Java to Java developers resource center; targeted at the technical and technical team lead (senior developer), project manager and junior developer. JCGs serve the Java, SOA, Agile and Telecom communities with daily news, web domain experts, articles, tutorials, reviews, announcements, code snippets and source projects.

## DISCLAIMER

All trademarks and registered trademarks appearing on Java Code Geeks are the property of their respective owners. Java is a trademark or registered trademark of Oracle Corporation in the United States and other countries. Examples Java CI is not connected to Oracle Corporation and is not sponsored by Oracle Corporation.

[.NET Code Geeks](#)

[Java Code Geeks](#)

[System Code Geeks](#)

[Web Code Geeks](#)

[Android Quick Preferences Tutorial](#)

[Difference between Comparator and Comparable in Java](#)

[GWT 2 Spring 3 JPA 2 Hibernate 3.5 Tutorial](#)

[Java Best Practices – Vector vs ArrayList vs HashSet](#)