

UNIVERSIDADE FEDERAL DE SERGIPE

VINÍCIUS DIAS VALENÇA

Teste DE SOFTWARE 2

ATIVIDADE 1

Problema do Stack Overflow

São Cristóvão

2024

1. Motivação

How do you assert that a certain exception is thrown in JUnit tests?

<https://stackoverflow.com/questions/156503/how-do-you-assert-that-a-certain-exception-is-thrown-in-junit-tests>

The screenshot shows the Stack Overflow website interface. At the top, there's a navigation bar with the Stack Overflow logo, 'Products', 'OverflowAI', and a search bar. Below this is a blue banner for 'The 2024 Developer Survey results are live!'. The main content area features the question title 'How do you assert that a certain exception is thrown in JUnit tests?' with a '2311' votes badge. A green banner for 'SO para quem fala português!' is visible. The question body contains a code snippet for a JUnit test. To the right, there are sections for 'Featured on Meta' and 'Hot Meta Posts'. At the bottom right, there's a '2024 Developer Survey' badge.

Stack Overflow

Products OverflowAI Search...

The 2024 Developer Survey results are live! See the results

Home Questions Tags Saves Users Companies LABS Jobs Discussions COLLECTIVES Teams

How do you assert that a certain exception is thrown in JUnit tests? Ask Question

Asked 15 years, 10 months ago Modified 3 months ago Viewed 2.0m times

SO para quem fala português! stackoverflow em Português

How can I use JUnit idiomatically to test that some code throws an exception?

While I can certainly do something like this:

```
@Test
public void testFooThrowsIndexOutOfBoundsException() {
    boolean thrown = false;

    try {
        foo.doStuff();
    } catch (IndexOutOfBoundsException e) {
        thrown = true;
    }

    assertTrue(thrown);
}
```

I recall that there is an annotation or an Assert.xyz or something that is far less kludgy and far more in-the-spirit of JUnit for these sorts of situations.

Featured on Meta

- Announcing a change to the data-dump process
- We've made changes to our Terms of Service & Privacy Policy - July 2024

Hot Meta Posts

22 What is the tick next to the "0 answers"?

160 Please don't call ruby "RB" in the developer survey

2024 Developer Survey

See the results

```
@Test
public void testFooThrowsIndexOutOfBoundsException() {

    boolean thrown = false;

    try {
        foo.doStuff();
    } catch (IndexOutOfBoundsException e) {
        thrown = true;
    }

    assertTrue(thrown);
}
```

Ao escrever testes unitários, um cenário comum é garantir que uma exceção específica seja lançada quando certas condições são atendidas. Por exemplo, se um método deve lançar uma exceção quando recebe parâmetros inválidos, queremos testar e garantir esse comportamento.

Problemas com a Abordagem Inicial

1. A necessidade de um bloco `try-catch` e a variável `thrown` adicionam código desnecessário, tornando o teste mais verboso e difícil de ler.
2. O teste depende da variável `thrown` ser corretamente definida no bloco `catch`. Qualquer alteração acidental pode fazer o teste falhar silenciosamente.
3. Se o teste falhar, a mensagem de erro padrão pode não ser muito clara sobre o que deu errado.

2. Reproduzindo o problema

```
4
5
6 package com.example;
7 import java.util.List;
8
9 public class ClassToTest {
10     public void checkNames(List<String> names) {
11         for (String name : names) {
12             if (name.length() <= 2) {
13                 throw new IllegalArgumentException("O nome precisa ter pelo menos 3 letras: " + name);
14             }
15         }
16     }
17 }
```

```
1 package com.example;
2 import org.junit.jupiter.api.Test;
3
4 import java.util.Arrays;
5 import java.util.List;
6
7 import static org.junit.jupiter.api.Assertions.assertEquals;
8 import static org.junit.jupiter.api.Assertions.assertThrows;
9
10 public class TestClass {
11
12     @Test
13     public void testCheckNamesThrowsExceptionForShortNames() {
14         ClassToTest foo = new ClassToTest();
15         List<String> names = Arrays.asList(...a:"Joe", "Al", "Sam");
16
17         IllegalArgumentException exception = assertThrows(IllegalArgumentException.class, () -> foo.checkNames(names));
18         assertEquals(expected:"O nome precisa ter pelo menos 3 letras: Al", exception.getMessage());
19     }
20
21     @Test
22     public void testCheckNamesDoesNotThrowExceptionForValidNames() {
23         ClassToTest foo = new ClassToTest();
24         List<String> names = Arrays.asList(...a:"Joe", "Sam", "Ann");
25
26         foo.checkNames(names);
27     }
28 }
29
30 }
```

testCheckNamesThrowsExceptionForShortNames:

Cria uma instância de `ClassToTest` e uma lista de nomes onde um dos nomes é muito curto.

Usa `assertThrows` para verificar que `IllegalArgumentException` é lançada.

Verifica se a mensagem da exceção corresponde à esperada.

testCheckNamesDoesNotThrowExceptionForValidNames:

Cria uma instância de `ClassToTest` e uma lista de nomes válidos.

Chama `checkNames` e verifica implicitamente que nenhuma exceção é lançada.

```
1  <project xmlns="http://maven.apache.org/POM/4.0.0"
2      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
3      xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">
4      <modelVersion>4.0.0</modelVersion>
5
6      <groupId>com.example</groupId>
7      <artifactId>atividade1-project</artifactId>
8      <version>1.0-SNAPSHOT</version>
9
10     <dependencies>
11         <dependency>+
12             <groupId>org.junit.jupiter</groupId>
13             <artifactId>junit-jupiter-api</artifactId>
14             <version>5.8.2</version>
15             <scope>test</scope>
16         </dependency>
17         <dependency>
18             <groupId>org.junit.jupiter</groupId>
19             <artifactId>junit-jupiter-engine</artifactId>
20             <version>5.8.2</version>
21             <scope>test</scope>
22         </dependency>
23     </dependencies>
24
25     <build>
26         <plugins>
27             <plugin>
28                 <groupId>org.apache.maven.plugins</groupId>
29                 <artifactId>maven-surefire-plugin</artifactId>
30                 <version>3.0.0-M5</version>
31             </plugin>
32         </plugins>
33     </build>
34 </project>
```

3. Outras soluções:



in junit, there are four ways to test exception.

248

junit5.x



- for junit5.x, you can use `assertThrows` as following



```
@Test
public void testFooThrowsIndexOutOfBoundsException() {
    Throwable exception = assertThrows(IndexOutOfBoundsException.class, () -> foo)
    assertEquals("expected messages", exception.getMessage());
}
```

Utilizar uma exceção específica, como `IllegalArgumentException`, comunica claramente a intenção do código e o tipo de erro que está sendo tratado. Isso torna o código mais legível e fácil de entender para outros desenvolvedores que possam trabalhar nele no futuro.

`Throwable` é a superclasse de todas as exceções e erros em Java. Utilizá-la para capturar exceções pode levar a capturar exceções não intencionais, como `NullPointerException`, `IOException`, ou mesmo `OutOfMemoryError`, que são casos que normalmente não deveriam ser tratados da mesma forma.