



Universidade Federal de Sergipe
Centro de Ciências Exatas e Tecnologia
Departamento de Computação

Atividade 1 – Testes Unitários e Stack Overflow

Teste de Software - COMP0444 - T01
Professor: Glauco de Figueiredo Carneiro
Aluno: João Rosa Conceição

Tutorial: Escolha da pergunta, Explicação e resolução do teste

Github: https://github.com/jrosac/Teste_Software_2024_Conceicao_Joao

Vídeo: https://drive.google.com/drive/u/3/folders/1wDTTI0kw2gtoGg3q7rxh_5V1HUg1L6rf

Etapa 1: Escolha da pergunta cadastrada

Pergunta escolhida:

<https://stackoverflow.com/questions/1096650/why-doesnt-junit-provide-assertnotequals-methods>

Descrição da pergunta:

Por que o JUnit 4 fornece o método **assertEquals(foo, bar)** mas não o **assertNotEquals(foo, bar)**? A biblioteca inclui métodos como **assertNotSame** (correspondente ao **assertSame**) e **assertFalse** (correspondente ao **assertTrue**), então parece estranho que eles não incluíram o **assertNotEquals**.

Etapa 2: Detalhamento, Resposta, realização dos testes e motivo das outras respostas não serem escolhidas

Detalhamento :

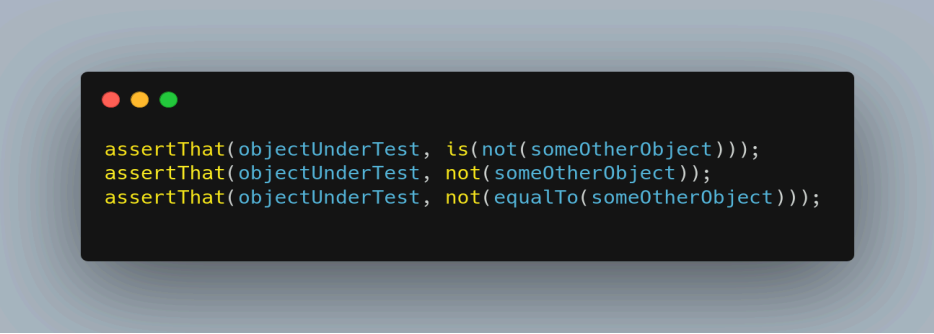
JUnit é uma das bibliotecas de teste unitário mais amplamente utilizadas em Java. No entanto, uma curiosidade que muitos desenvolvedores encontram ao trabalhar com JUnit 4 é a ausência do método **assertNotEquals**. Enquanto a biblioteca

oferece métodos como **assertEquals**, **assertSame**, **assertNotSame**, **assertTrue** e **assertFalse**, não há uma implementação nativa para **assertNotEquals**.

Isso pode parecer inconsistente, considerando que **assertEquals** possui um antagônico lógico, assim como **assertTrue** e **assertFalse** e **assertSame** e **assertNotSame**. A razão para essa ausência específica não é documentada explicitamente pelos criadores do JUnit.

Resposta aceita:


Sugiro que você use o estilo mais novo de asserções **assertThat()**, que pode facilmente descrever todos os tipos de negações e construir automaticamente uma descrição do que você esperava e do que obteve se a asserção falhar:



```
assertThat(objectUnderTest, is(not(someOtherObject)));
assertThat(objectUnderTest, not(someOtherObject));
assertThat(objectUnderTest, not(equalTo(someOtherObject)));
```

Todas as três opções são equivalentes; escolha a que achar mais legível.

Para usar os nomes simples dos métodos (e permitir que essa sintaxe funcione), você precisa dessas **importações**:



```
import static org.junit.Assert.*;
import static org.hamcrest.CoreMatchers.*;
```

Realização dos testes:

Serão feitos testes com base na pergunta do usuário, ou seja, primeiramente será utilizado o método **assertNotEquals**, o que de acordo com ele seria o mais intuitivo. Após esse teste, será realizado um teste com o método **assertThat**, que foi o utilizado pela resposta selecionada.

Os testes serão feitos utilizando esta classe **Pessoa** como exemplo:

```
public class Pessoa {
    private String nome;
    private int idade;

    public Pessoa(String nome, int idade) {
        this.nome = nome;
        this.idade = idade;
    }

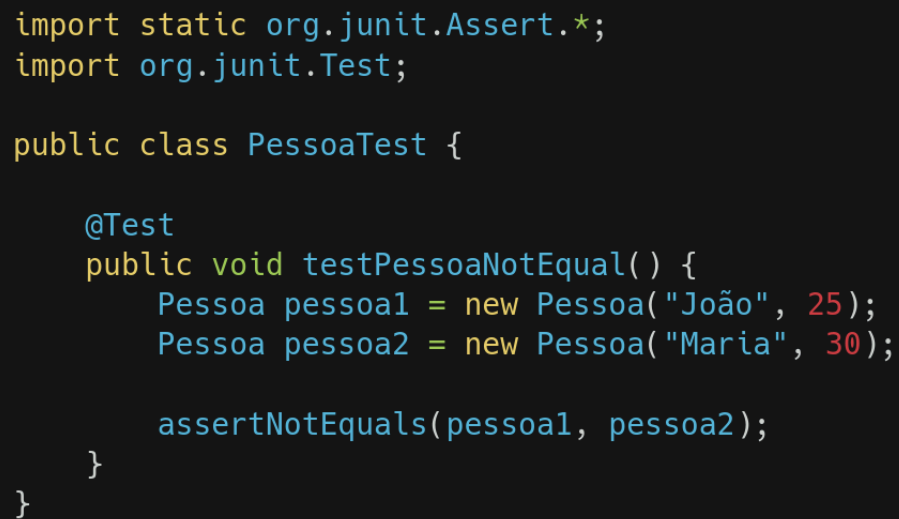
    public String getNome() {
        return nome;
    }

    public int getIdade() {
        return idade;
    }

    @Override
    public boolean equals(Object obj) {
        if (this == obj) return true;
        if (obj == null || getClass() != obj.getClass()) return false;
        Pessoa pessoa = (Pessoa) obj;
        return idade == pessoa.idade && nome.equals(pessoa.nome);
    }

    @Override
    public int hashCode() {
        return Objects.hash(nome, idade);
    }
}
```

1. Teste com **assertNotEquals**:



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

public class PessoaTest {

    @Test
    public void testPessoaNotEqual() {
        Pessoa pessoa1 = new Pessoa("João", 25);
        Pessoa pessoa2 = new Pessoa("Maria", 30);

        assertNotEquals(pessoa1, pessoa2);
    }
}
```

2. Teste com **assertThat**:

```
import static org.hamcrest.CoreMatchers.*;
import static org.junit.Assert.*;
import org.junit.Test;

public class PessoaTest {

    @Test
    public void testPessoaNotEqual() {
        Pessoa pessoa1 = new Pessoa("João", 25);
        Pessoa pessoa2 = new Pessoa("Maria", 30);

        assertThat(pessoa1, is(not(equalTo(pessoa2))));
    }
}
```

Porque as outras respostas não foram aceitas:

1º Resposta:

There is an `assertNotEquals` in JUnit 4.11:

<https://github.com/junit-team/junit/blob/master/doc/ReleaseNotes4.11.md#improvements-to-assert-and-assume>

```
import static org.junit.Assert.assertNotEquals;
```

Justificativa:

- **Atualização Direta:** A resposta oferece uma informação direta e factual sobre a introdução de **assertNotEquals** em uma versão posterior do JUnit, que é útil, mas não explora outras abordagens ou boas práticas.
- **Limitação de Versão:** Essa resposta é útil apenas para aqueles que podem atualizar para a versão 4.11 ou superior, o que pode não ser possível para todos os usuários devido a restrições do ambiente de desenvolvimento.

2º Resposta:

I wonder same. The API of Assert is not very symmetric; for testing whether objects are the same, it provides `assertSame` and `assertNotSame`.

Of course, it is not too long to write:

```
assertFalse(foo.equals(bar));
```

With such an assertion, the only informative part of the output is unfortunately the name of the test method, so descriptive message should be formed separately:

```
String msg = "Expected <" + foo + "> to be unequal to <" + bar + ">";
```

```
assertFalse(msg, foo.equals(bar));
```

That is of course so tedious, that it is better to roll your own `assertNotEqual`. Luckily in future it will maybe be part of the JUnit: [JUnit issue 22](#)

Justificativa:

- **Falta de Simetria:** O autor observa a falta de simetria na API do JUnit, mas não oferece uma solução prática além de `assertFalse(foo.equals(bar))`, que é menos expressivo.
- **Descritividade Limitada:** A resposta aponta corretamente a limitação da abordagem `assertFalse` em fornecer mensagens descritivas, mas a solução sugerida para formar uma mensagem é considerada tediosa e não é automática.

3º Resposta:

I'm coming to this party pretty late but I have found that the form:

```
static void assertTrue(java.lang.String message, boolean condition)
```

can be made to work for most 'not equals' cases.

```
int status = doSomething() ; // expected to return 123
```

```
assertTrue("doSomething() returned unexpected status", status != 123 ) ;
```

Justificativa:

- **Simplicidade:**A resposta não verificada sugere o uso do método assertTrue para verificar condições de "não igual". Embora essa abordagem funcione, ela não é tão direta quanto ter um método específico para isso.
- **Menos Flexibilidade:**Usar assertTrue requer que o desenvolvedor manualmente descreva a condição e a mensagem, o que pode ser mais propenso a erros e menos intuitivo, especialmente para testes mais complexos.
- **Falta de Atualização:**A resposta não menciona abordagens modernas ou alternativas que podem ser mais adequadas no contexto atual de desenvolvimento, o que pode ser visto como uma limitação.
- **Exemplo Único:**O exemplo fornecido é funcional, mas não explora a variedade de casos ou apresenta a mesma clareza e adaptabilidade que a solução assertThat.