Experimento de Refatoração de Test Smells

Nome: Gabriel Maia Gondim - 478943

Projeto: https://github.com/gabrielmaia2/commons-lang

Eu estou atualmente trabalhando na refatoração dos test smells seguintes:

5 test smells do tipo duplicate assertion no arquivo commons-lang\src\test\java\org\apache\commons\lang3\CharRangeTest.java.

Código original:

```java
@Test
public void testContains_Char() {
    CharRange range = CharRange.is('c');
    assertFalse(range.contains('b'));
    assertTrue(range.contains('c'));
    assertFalse(range.contains('d'));
    assertFalse(range.contains('e'));

    range = CharRange.isIn('c', 'd');
    assertFalse(range.contains('b'));
    assertTrue(range.contains('c'));
    assertTrue(range.contains('d'));
    assertFalse(range.contains('e'));

    range = CharRange.isIn('d', 'c');
    assertFalse(range.contains('b'));
    assertTrue(range.contains('c'));
    assertTrue(range.contains('d'));
    assertFalse(range.contains('e'));

    range = CharRange.isNotIn('c', 'd');
    assertTrue(range.contains('b'));
    assertFalse(range.contains('c'));
    assertFalse(range.contains('d'));
    assertTrue(range.contains('e'));
    assertTrue(range.contains((char) 0));
    assertTrue(range.contains(Character.MAX_VALUE));
}
```

Código refatorado:

```java
@Test
public void testContains_CharEx1() {
    CharRange range = CharRange.is('c');
    assertFalse(range.contains('b'));
    assertTrue(range.contains('c'));
    assertFalse(range.contains('d'));
    assertFalse(range.contains('e'));
```

```
}

@Test
public void testContains_CharEx2() {
    CharRange range = CharRange.isIn('c', 'd');
    assertFalse(range.contains('b'));
    assertTrue(range.contains('c'));
    assertTrue(range.contains('d'));
    assertFalse(range.contains('e'));
}

@Test
public void testContains_CharEx3() {
    CharRange range = CharRange.isIn('d', 'c');
    assertFalse(range.contains('b'));
    assertTrue(range.contains('c'));
    assertTrue(range.contains('d'));
    assertFalse(range.contains('e'));
}

@Test
public void testContains_CharEx4() {
    CharRange range = CharRange.isNotIn('c', 'd');
    assertTrue(range.contains('b'));
    assertFalse(range.contains('c'));
    assertFalse(range.contains('d'));
    assertTrue(range.contains('e'));
    assertTrue(range.contains((char) 0));
    assertTrue(range.contains(Character.MAX_VALUE));
}
```

3 test smells do tipo eager test no arquivo commons-lang\src\test\java\org\apache\commons\lang3\builder\ToStringBuilderTest.java.

Código original:

```
@Test
public void testConstructorEx1() {
    assertEquals("<null>", new ToStringBuilder(null).toString());
}

@Test
public void testConstructorEx2() {
    assertEquals("<null>", new ToStringBuilder(null, null).toString());
    new ToStringBuilder(this.base, null).toString();
}

@Test
public void testConstructorEx3() {
    assertEquals("<null>", new ToStringBuilder(null, null, null).toString());
    new ToStringBuilder(this.base, null, null).toString();
    new ToStringBuilder(this.base, ToStringStyle.DEFAULT_STYLE, null).toString();
}
```

Código refatorado:

```
private ToStringBuilder defaultBuilder1 = new ToStringBuilder(null);
private ToStringBuilder defaultBuilder2 = new ToStringBuilder(null, null);
private ToStringBuilder defaultBuilder3 = new ToStringBuilder(null, null, null);

@Test
public void testConstructorEx1() {
    assertEquals("<null>", defaultBuilder1.toString());
}

@Test
public void testConstructorEx2() {
    assertEquals("<null>", defaultBuilder2.toString());
}

@Test
public void testConstructorEx3() {
    assertEquals("<null>", defaultBuilder3.toString());
}
```

Minhas principais dificuldades ao remover essas anormalidades foram:

Os eager tests foram mais complicados de serem consertados por causa da ferramenta de detecção, que detecta o uso de mais de uma função da classe testada como eager test (incluindo o uso da construtora). Portanto, tive que separar as construtoras em variáveis separadas, fora da função para que ele fosse consertado.

Também nos eager tests, tive que retirar variáveis que foram instanciadas e chamadas toString, mas não faziam nada nem eram testadas.

Os outros test smells foram simples, apenas separei em funções de teste diferentes.

Eu estou usando as seguintes técnicas de refatoração para remover test smells:

Eager Tests: Extrair variável, separando a instância do teste e também removi instâncias desnecessárias de variáveis.

Duplicate assertion: Extrair método, separando testes individuais em seus próprios métodos..

De 0 a 10, quão prejudicial é esse test smell para o sistema? Por que?

Eager test: 3.
Eager tests geralmente podem causar certos problemas por terem várias funções sendo testadas em um mesmo teste, o que torna o teste confuso e muito acoplado a várias funcionalidades diferentes, mas nesse caso, ele não era prejudicial, pois era simplesmente uma construtora sendo testada e precisava ser chamado algum método para verificar o estado do objeto e testar a construtora, como no caso, o toString().

Sobre as instâncias desnecessárias, não vejo elas como parte do eager test, acredito que tenham sido apenas colocadas lá pelo programador na hora para lembrar de escrever testes para aqueles casos, mas acabou esquecendo de escrevê-los.
Duplicate assertion: 7.
No duplicate assertion, ele está testando casos diferentes do uso das funções para criar ranges de caracteres (CharRange.is() e CharRange.isIn()). Deveriam ser separadas as responsabilidades, separando cada caso em um método de teste diferente.