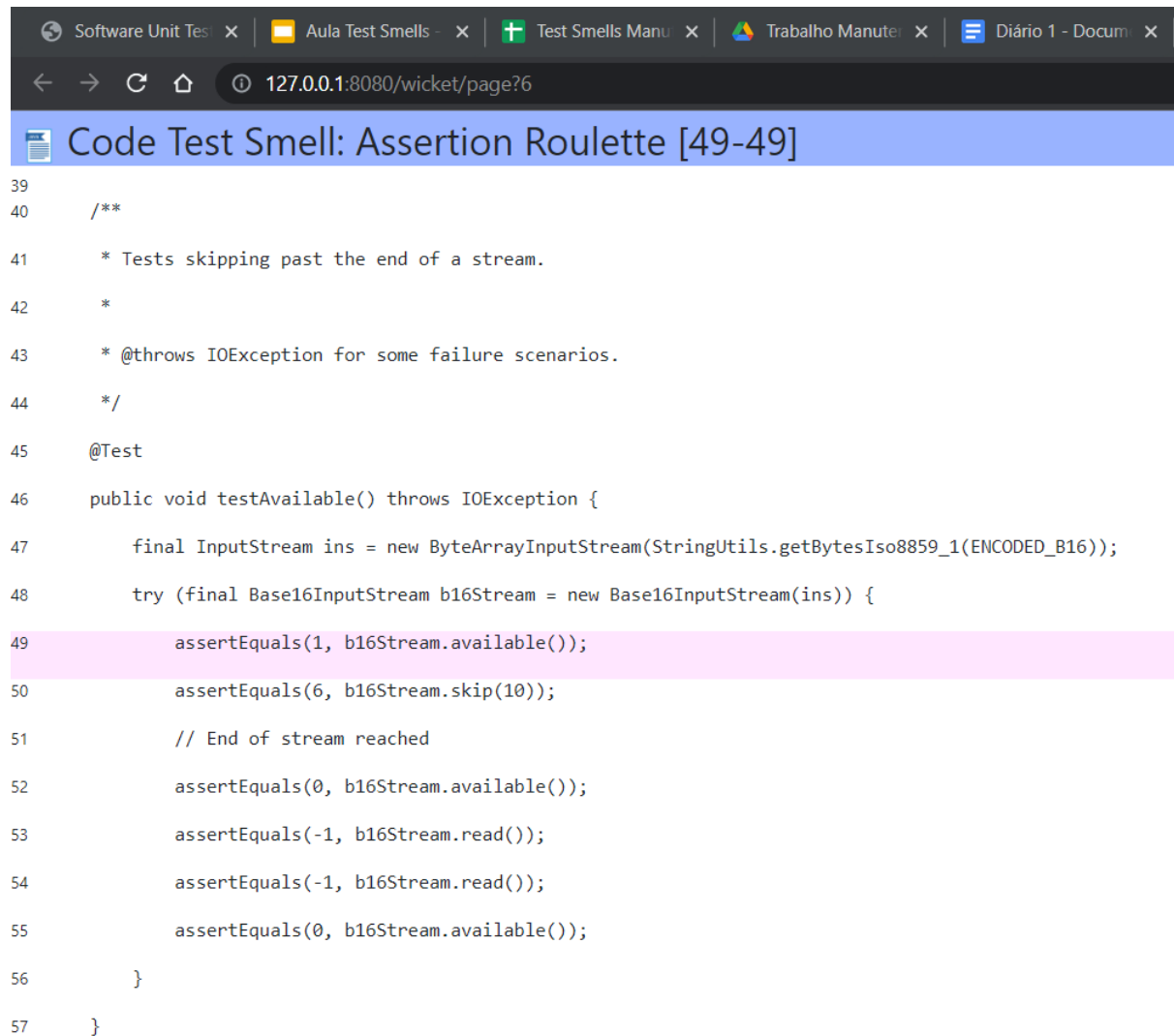


Assertion Roulette - 1 (linha 49-49)



The screenshot shows a web browser with several tabs: 'Software Unit Test', 'Aula Test Smells', 'Test Smells Manu', 'Trabalho Manute', and 'Diário 1 - Docum'. The address bar shows '127.0.0.1:8080/wicket/page?6'. The main content area has a blue header with the text 'Code Test Smell: Assertion Roulette [49-49]'. Below the header, a Java code snippet is displayed with line numbers 39 to 57. Line 49 is highlighted in pink, indicating a test smell. The code is as follows:

```
39
40  /**
41   * Tests skipping past the end of a stream.
42   *
43   * @throws IOException for some failure scenarios.
44   */
45  @Test
46  public void testAvailable() throws IOException {
47      final InputStream ins = new ByteArrayInputStream(StringUtils.getBytesIso8859_1(ENCODED_B16));
48      try (final Base16InputStream b16Stream = new Base16InputStream(ins)) {
49          assertEquals(1, b16Stream.available());
50          assertEquals(6, b16Stream.skip(10));
51          // End of stream reached
52          assertEquals(0, b16Stream.available());
53          assertEquals(-1, b16Stream.read());
54          assertEquals(-1, b16Stream.read());
55          assertEquals(0, b16Stream.available());
56      }
57  }
```

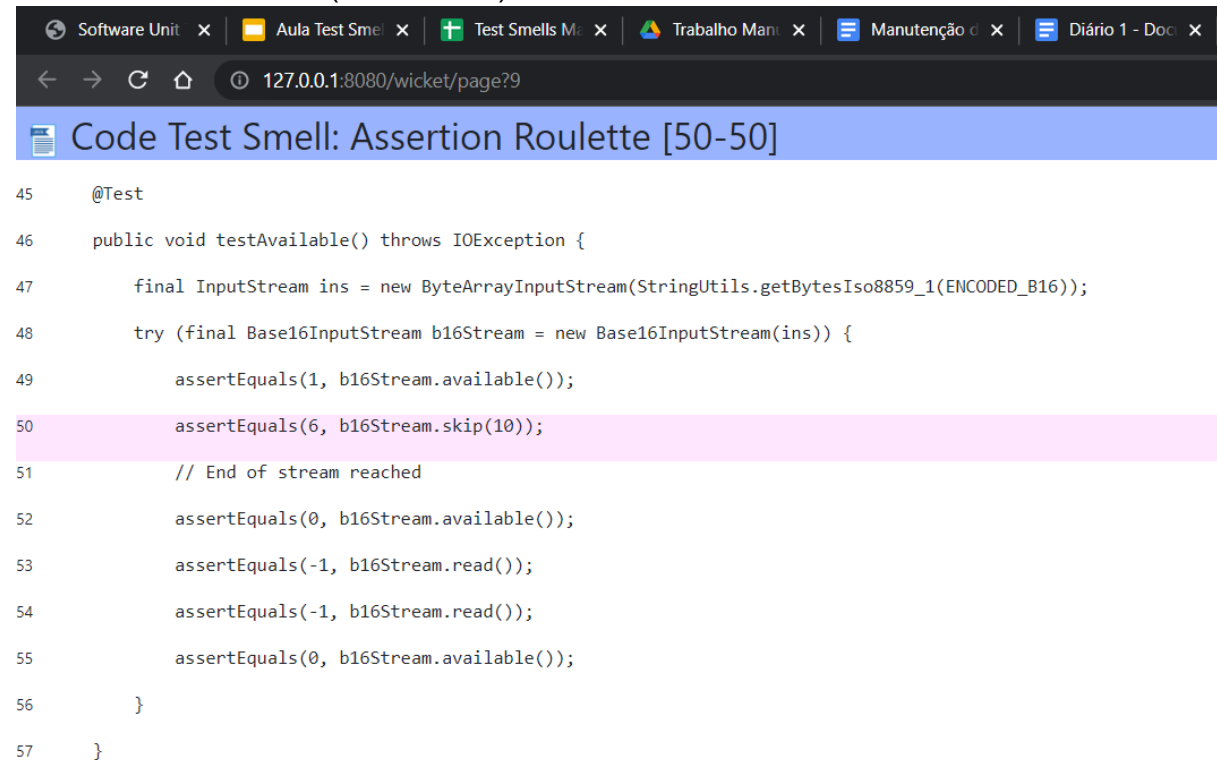
Teste refatorado



The screenshot shows a code editor with a file named 'PrimeiraEntrega.java'. The code is as follows:

```
src > PrimeiraEntrega.java > ...
1  public class PrimeiraEntrega {
2      public void testAvailable() throws IOException {
3          final InputStream ins = new ByteArrayInputStream(StringUtils.getBytesIso8859_1(ENCODED_B16));
4          try (final Base16InputStream b16Stream = new Base16InputStream(ins)) {
5              assertEquals(1, b16Stream.available(), "Testando se o método available() de b16Stream é 1");
6              assertEquals(6, b16Stream.skip(10));
7              // End of stream reached
8              assertEquals(0, b16Stream.available());
9              assertEquals(-1, b16Stream.read());
10             assertEquals(-1, b16Stream.read());
11             assertEquals(0, b16Stream.available());
12         }
13     }
14 }
15
```


Assertion Roulette - 2 (linha 50-50)



The screenshot shows a web browser with several tabs: 'Software Unit', 'Aula Test Smell', 'Test Smells M...', 'Trabalho Man...', 'Manutenção', and 'Diário 1 - Doc...'. The address bar shows '127.0.0.1:8080/wicket/page?9'. The main content area has a blue header with the text 'Code Test Smell: Assertion Roulette [50-50]'. Below the header, a code editor displays Java code for a test method. Line 50 is highlighted in pink, indicating a code test smell. The code is as follows:

```
45     @Test
46     public void testAvailable() throws IOException {
47         final InputStream ins = new ByteArrayInputStream(StringUtils.getBytesIso8859_1(ENCODED_B16));
48         try (final Base16InputStream b16Stream = new Base16InputStream(ins)) {
49             assertEquals(1, b16Stream.available());
50             assertEquals(6, b16Stream.skip(10));
51             // End of stream reached
52             assertEquals(0, b16Stream.available());
53             assertEquals(-1, b16Stream.read());
54             assertEquals(-1, b16Stream.read());
55             assertEquals(0, b16Stream.available());
56         }
57     }
```

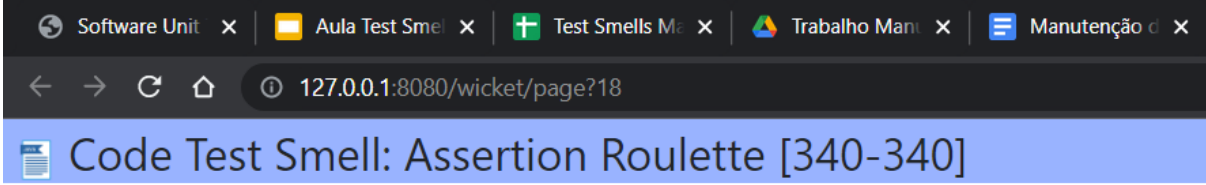
Teste refatorado:



The screenshot shows a code editor with a dark background. The file name is 'PrimeiraEntrega.java'. The code is as follows:

```
src > PrimeiraEntrega.java > ...
1  public class PrimeiraEntrega {
2      public void testAvailable() throws IOException {
3          final InputStream ins = new ByteArrayInputStream(StringUtils.getBytesIso8859_1(ENCODED_B16));
4          try (final Base16InputStream b16Stream = new Base16InputStream(ins)) {
5              assertEquals(1, b16Stream.available());
6              assertEquals(6, b16Stream.skip(10), "Testando se o método skip(10) de b16Stream resulta em 6");
7              // End of stream reached
8              assertEquals(0, b16Stream.available());
9              assertEquals(-1, b16Stream.read());
10             assertEquals(-1, b16Stream.read());
11             assertEquals(0, b16Stream.available());
12         }
13     }
14 }
15
```

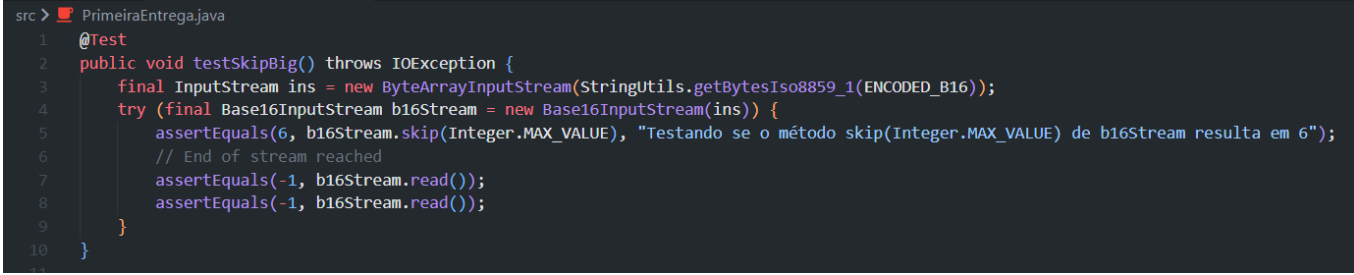
Assertion Roulette - 3 (linha 340-340)



The screenshot shows a web browser with several tabs: 'Software Unit', 'Aula Test Smell', 'Test Smells Me', 'Trabalho Mani', and 'Manutenção d'. The address bar shows '127.0.0.1:8080/wicket/page?18'. The main content area has a blue header with the text 'Code Test Smell: Assertion Roulette [340-340]'. Below the header, a code snippet is displayed with line numbers 336 to 345. Line 340 is highlighted in pink, indicating a test smell.

```
336     @Test
337     public void testSkipBig() throws IOException {
338         final InputStream ins = new ByteArrayInputStream(StringUtils.getBytesIso8859_1(ENCODED_
339         try (final Base16InputStream b16Stream = new Base16InputStream(ins)) {
340             assertEquals(6, b16Stream.skip(Integer.MAX_VALUE));
341             // End of stream reached
342             assertEquals(-1, b16Stream.read());
343             assertEquals(-1, b16Stream.read());
344         }
345     }
```

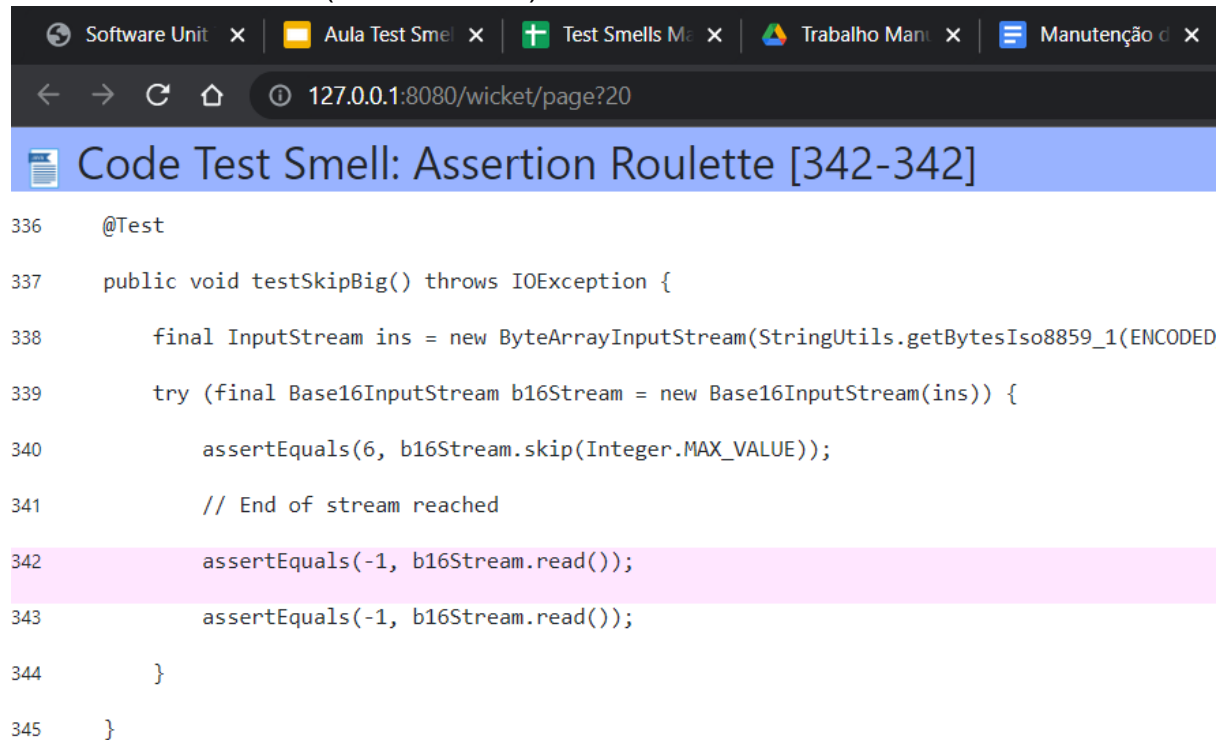
Teste refatorado



The screenshot shows a code editor with a file named 'PrimeiraEntrega.java'. The code is a Java test method 'testSkipBig' that has been refactored. It includes a descriptive message in the 'assertEquals' call on line 5.

```
src > PrimeiraEntrega.java
1  @Test
2  public void testSkipBig() throws IOException {
3      final InputStream ins = new ByteArrayInputStream(StringUtils.getBytesIso8859_1(ENCODED_B16));
4      try (final Base16InputStream b16Stream = new Base16InputStream(ins)) {
5          assertEquals(6, b16Stream.skip(Integer.MAX_VALUE), "Testando se o método skip(Integer.MAX_VALUE) de b16Stream resulta em 6");
6          // End of stream reached
7          assertEquals(-1, b16Stream.read());
8          assertEquals(-1, b16Stream.read());
9      }
10 }
```

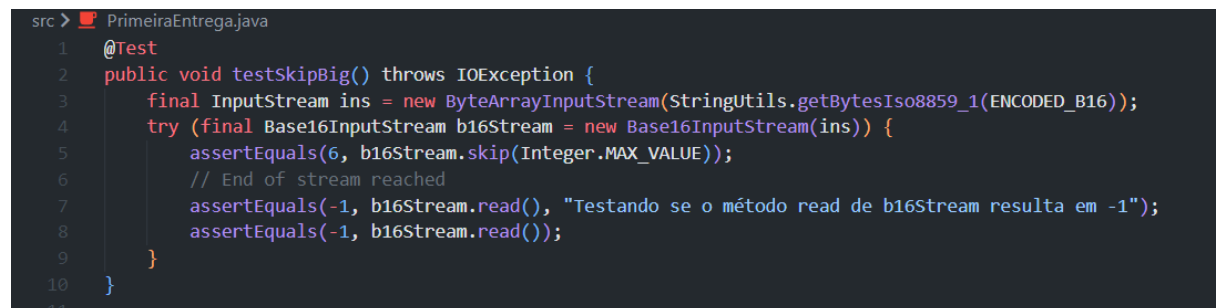
Assertion Roulette - 4 (linha 342-342)



The screenshot shows a web browser with multiple tabs: 'Software Unit', 'Aula Test Sme', 'Test Smells Me', 'Trabalho Man', and 'Manutenção d'. The address bar shows '127.0.0.1:8080/wicket/page?20'. The page title is 'Code Test Smell: Assertion Roulette [342-342]'. Below the title, a code snippet is displayed with line numbers 336 to 345. Line 342 is highlighted in pink, indicating a test smell. The code is as follows:

```
336     @Test
337     public void testSkipBig() throws IOException {
338         final InputStream ins = new ByteArrayInputStream(StringUtils.getBytesIso8859_1(ENCODED
339         try (final Base16InputStream b16Stream = new Base16InputStream(ins)) {
340             assertEquals(6, b16Stream.skip(Integer.MAX_VALUE));
341             // End of stream reached
342             assertEquals(-1, b16Stream.read());
343             assertEquals(-1, b16Stream.read());
344         }
345     }
```

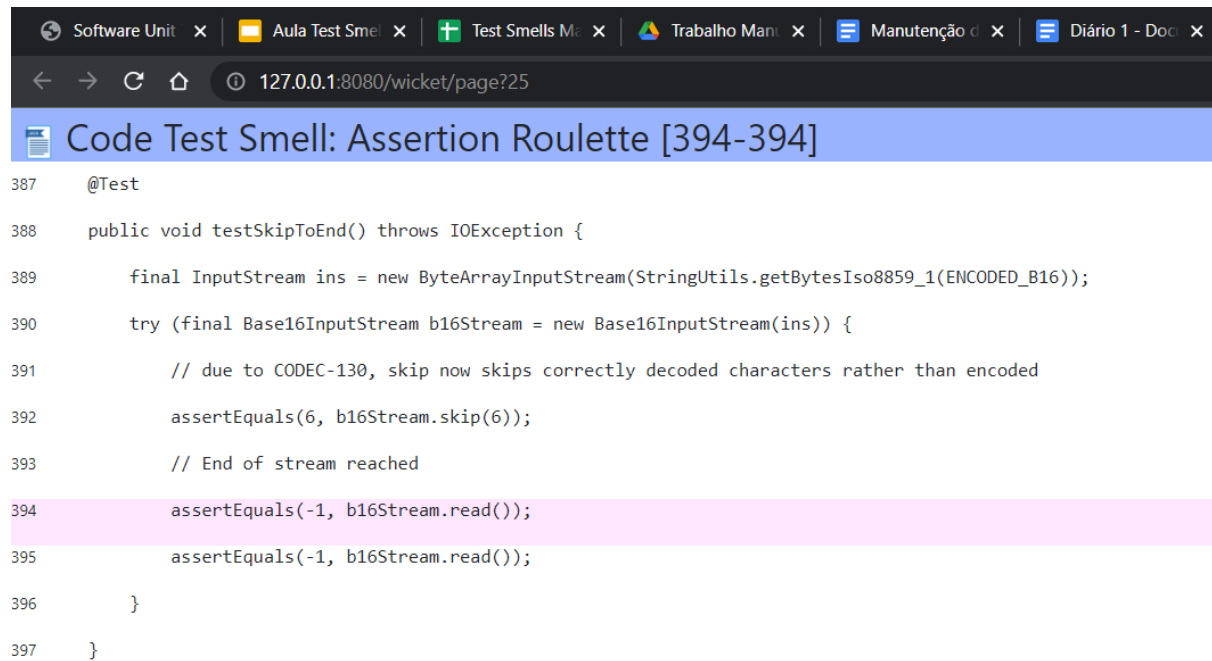
Teste refatorado



The screenshot shows a code editor with a file named 'PrimeiraEntrega.java'. The code is as follows:


```
src > PrimeiraEntrega.java
1  @Test
2  public void testSkipBig() throws IOException {
3      final InputStream ins = new ByteArrayInputStream(StringUtils.getBytesIso8859_1(ENCODED_B16));
4      try (final Base16InputStream b16Stream = new Base16InputStream(ins)) {
5          assertEquals(6, b16Stream.skip(Integer.MAX_VALUE));
6          // End of stream reached
7          assertEquals(-1, b16Stream.read(), "Testando se o método read de b16Stream resulta em -1");
8          assertEquals(-1, b16Stream.read());
9      }
10 }
11
```

Assertion Roulette - 5 (linha 394-394)



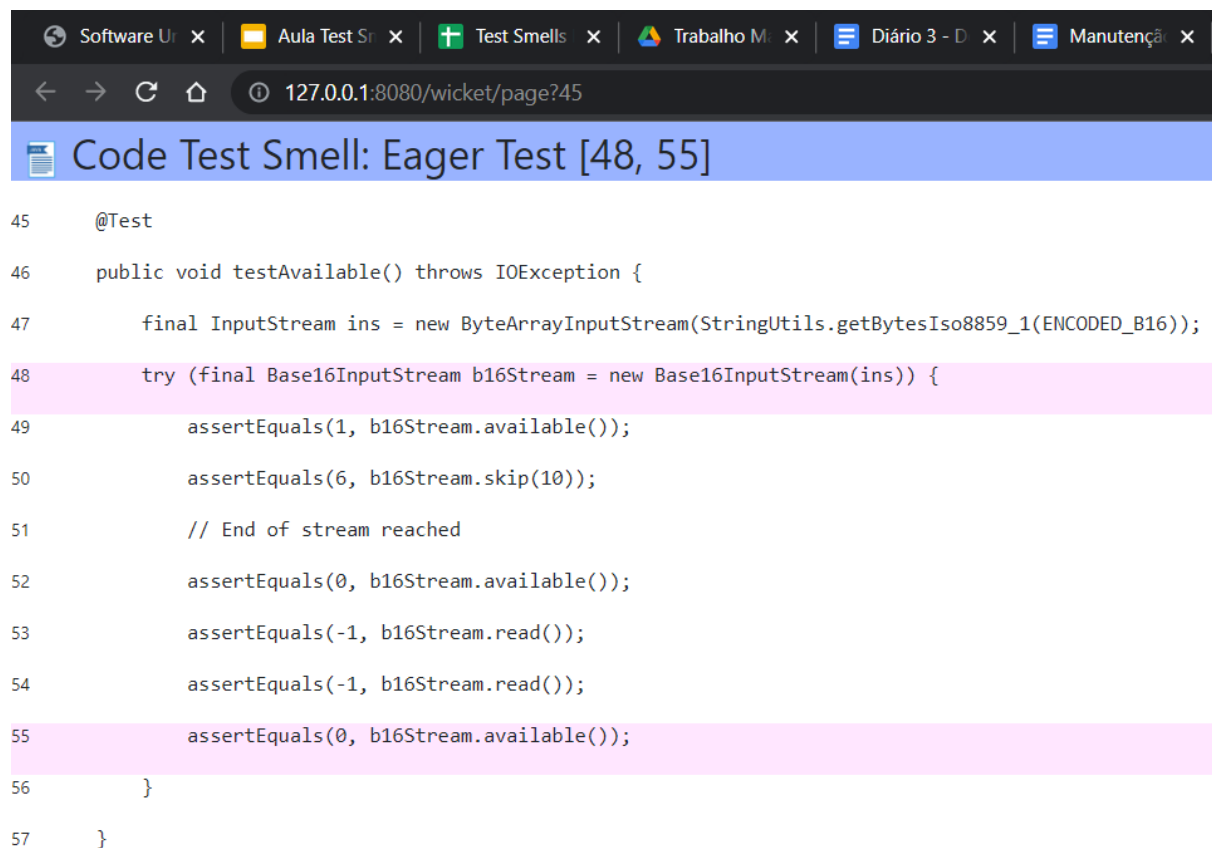
```
387     @Test
388     public void testSkipToEnd() throws IOException {
389         final InputStream ins = new ByteArrayInputStream(StringUtils.getBytesIso8859_1(ENCODED_B16));
390         try (final Base16InputStream b16Stream = new Base16InputStream(ins)) {
391             // due to CODEC-130, skip now skips correctly decoded characters rather than encoded
392             assertEquals(6, b16Stream.skip(6));
393             // End of stream reached
394             assertEquals(-1, b16Stream.read());
395             assertEquals(-1, b16Stream.read());
396         }
397     }
```

Teste refatorado



```
src > PrimeiraEntrega.java
1  @Test
2  public void testSkipToEnd() throws IOException {
3      final InputStream ins = new ByteArrayInputStream(StringUtils.getBytesIso8859_1(ENCODED_B16));
4      try (final Base16InputStream b16Stream = new Base16InputStream(ins)) {
5          // due to CODEC-130, skip now skips correctly decoded characters rather than encoded
6          assertEquals(6, b16Stream.skip(6));
7          // End of stream reached
8          assertEquals(-1, b16Stream.read(), "Testando se o método read de b16Stream resulta em -1");
9          assertEquals(-1, b16Stream.read());
10     }
11 }
12
```

Eager Test - 1 (linha 48-55)



The screenshot shows a web browser with multiple tabs. The active tab is titled "Code Test Smell: Eager Test [48, 55]". The address bar shows the URL "127.0.0.1:8080/wicket/page?45". The main content area displays a Java code snippet with line numbers 45 to 57. The code is a JUnit test method named `testAvailable()` that checks the availability of a `Base64InputStream` after skipping 10 bytes. Lines 48-55 are highlighted in pink, indicating the detected code smell.

```
45     @Test
46     public void testAvailable() throws IOException {
47         final InputStream ins = new ByteArrayInputStream(StringUtils.getBytesIso8859_1(ENCODED_B16));
48         try (final Base64InputStream b16Stream = new Base64InputStream(ins)) {
49             assertEquals(1, b16Stream.available());
50             assertEquals(6, b16Stream.skip(10));
51             // End of stream reached
52             assertEquals(0, b16Stream.available());
53             assertEquals(-1, b16Stream.read());
54             assertEquals(-1, b16Stream.read());
55             assertEquals(0, b16Stream.available());
56         }
57     }
```

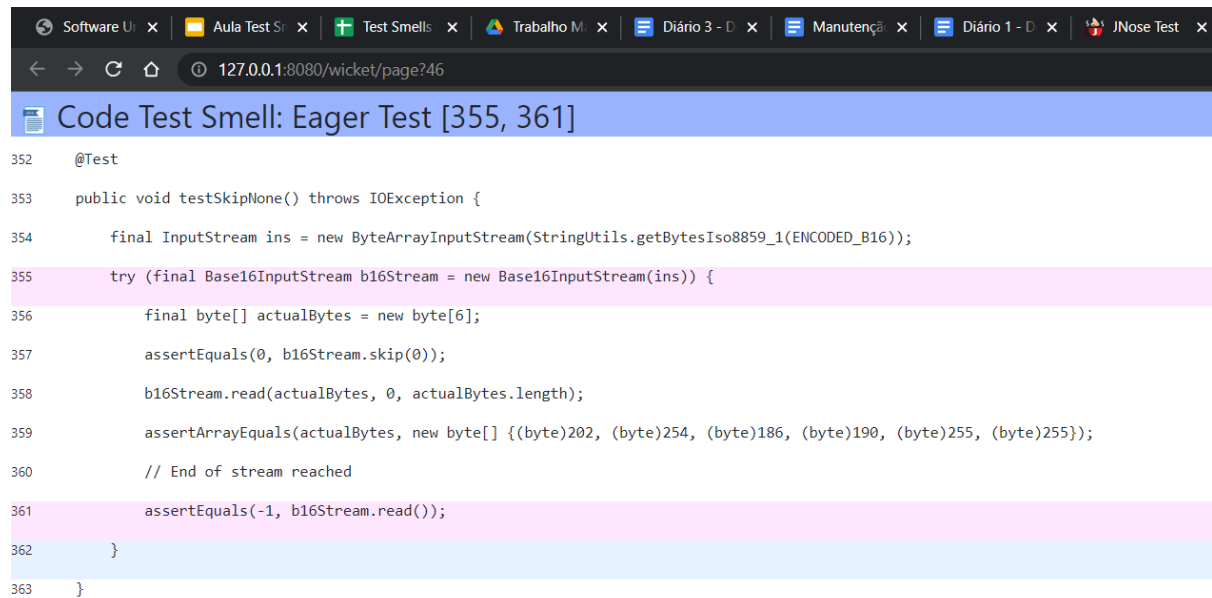
Teste refatorado



The screenshot shows a code editor with a dark background. It displays a refactored Java test class with three methods: `testAvailable()`, `Skip()`, and `Read()`. Each method is a JUnit test that checks the behavior of a `Base64InputStream` under different conditions. The code is color-coded, with keywords in blue, strings in red, and comments in green.

```
1  @Test
2  public void testAvailable() throws IOException {
3      final InputStream ins = new ByteArrayInputStream(StringUtils.getBytesIso8859_1(ENCODED_B16));
4      try (final Base64InputStream b16Stream = new Base64InputStream(ins)) {
5          assertEquals(1, b16Stream.available());
6          assertEquals(0, b16Stream.available());
7          assertEquals(0, b16Stream.available());
8      }
9  }
10
11 public void Skip(){
12     final InputStream ins = new ByteArrayInputStream(StringUtils.getBytesIso8859_1(ENCODED_B16));
13     try (final Base64InputStream b16Stream = new Base64InputStream(ins)) {
14         assertEquals(6, b16Stream.skip(10));
15     }
16 }
17
18
19 public void Read(){
20     final InputStream ins = new ByteArrayInputStream(StringUtils.getBytesIso8859_1(ENCODED_B16));
21     try (final Base64InputStream b16Stream = new Base64InputStream(ins)) {
22         assertEquals(-1, b16Stream.read());
23         assertEquals(-1, b16Stream.read());
24     }
25 }
```

Eager Test - 2 (linha 355-361)



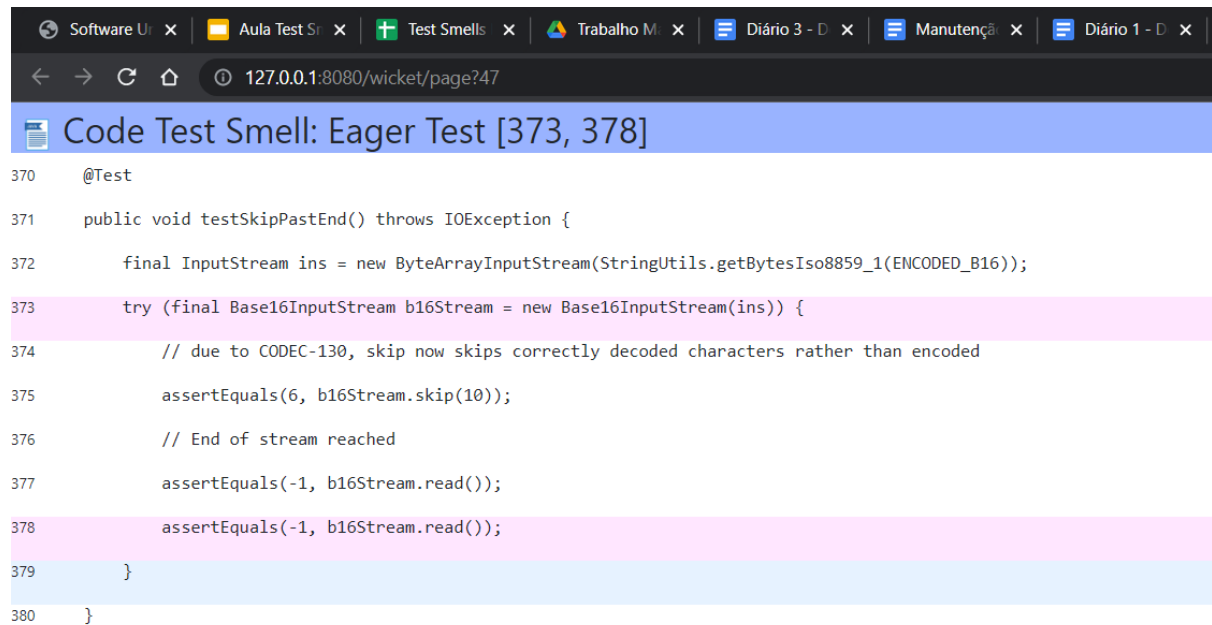
```
352  @Test
353  public void testSkipNone() throws IOException {
354      final InputStream ins = new ByteArrayInputStream(StringUtils.getBytesIso8859_1(ENCODED_B16));
355      try (final Base16InputStream b16Stream = new Base16InputStream(ins)) {
356          final byte[] actualBytes = new byte[6];
357          assertEquals(0, b16Stream.skip(0));
358          b16Stream.read(actualBytes, 0, actualBytes.length);
359          assertEquals(actualBytes, new byte[] {(byte)202, (byte)254, (byte)186, (byte)190, (byte)255, (byte)255});
360          // End of stream reached
361          assertEquals(-1, b16Stream.read());
362      }
363  }
```

Teste refatorado



```
1  @Test
2  public void testSkipNone() throws IOException {
3      final InputStream ins = new ByteArrayInputStream(StringUtils.getBytesIso8859_1(ENCODED_B16));
4      try (final Base16InputStream b16Stream = new Base16InputStream(ins)) {
5          final byte[] actualBytes = new byte[6];
6          assertEquals(0, b16Stream.skip(0));
7      }
8  }
9
10 public void testSkipNone() throws IOException {
11     final InputStream ins = new ByteArrayInputStream(StringUtils.getBytesIso8859_1(ENCODED_B16));
12     try (final Base16InputStream b16Stream = new Base16InputStream(ins)) {
13         final byte[] actualBytes = new byte[6];
14         b16Stream.read(actualBytes, 0, actualBytes.length);
15         assertEquals(actualBytes, new byte[] {(byte)202, (byte)254, (byte)186, (byte)190, (byte)255, (byte)255});
16         // End of stream reached
17         assertEquals(-1, b16Stream.read());
18     }
19 }
```

Eager Test - 3 (linha 373-378)



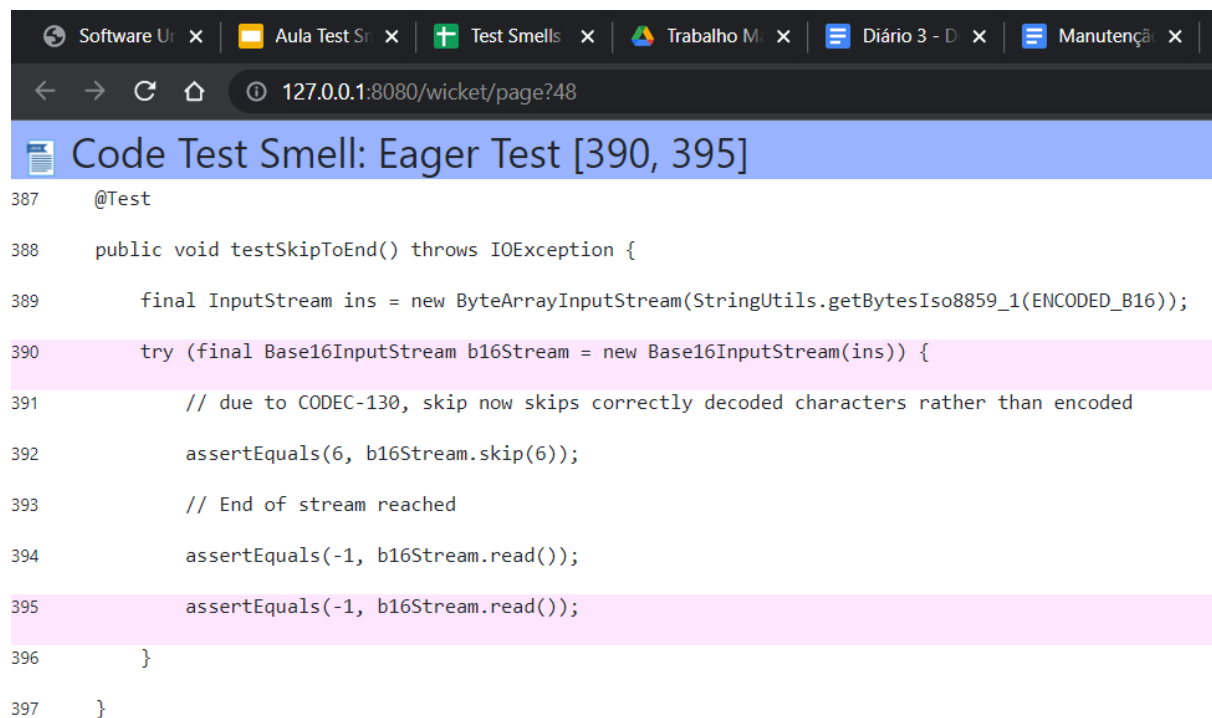
```
370     @Test
371     public void testSkipPastEnd() throws IOException {
372         final InputStream ins = new ByteArrayInputStream(StringUtils.getBytesIso8859_1(ENCODED_B16));
373         try (final Base16InputStream b16Stream = new Base16InputStream(ins)) {
374             // due to CODEC-130, skip now skips correctly decoded characters rather than encoded
375             assertEquals(6, b16Stream.skip(10));
376             // End of stream reached
377             assertEquals(-1, b16Stream.read());
378             assertEquals(-1, b16Stream.read());
379         }
380     }
```

Teste refatorado



```
1  @Test
2  public void testSkipPastEnd() throws IOException {
3      final InputStream ins = new ByteArrayInputStream(StringUtils.getBytesIso8859_1(ENCODED_B16));
4      try (final Base16InputStream b16Stream = new Base16InputStream(ins)) {
5          // due to CODEC-130, skip now skips correctly decoded characters rather than encoded
6          assertEquals(6, b16Stream.skip(10));
7          // End of stream reached
8      }
9  }
10
11 public void testSkipPastEnd() throws IOException {
12     final InputStream ins = new ByteArrayInputStream(StringUtils.getBytesIso8859_1(ENCODED_B16));
13     try (final Base16InputStream b16Stream = new Base16InputStream(ins)) {
14         assertEquals(-1, b16Stream.read());
15         assertEquals(-1, b16Stream.read());
16     }
17 }
```

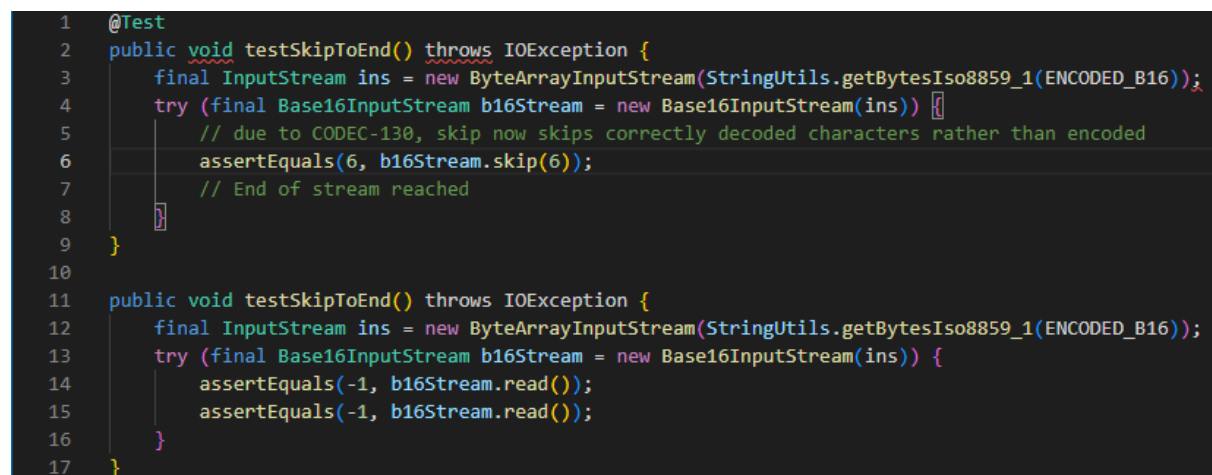

Eager Test - 4(linha 390-395)



The screenshot shows a web browser window with multiple tabs. The active tab is titled 'Code Test Smell: Eager Test [390, 395]'. The address bar shows the URL '127.0.0.1:8080/wicket/page?48'. The main content area displays a code snippet with line numbers 387 to 397. The code is a Java test method named 'testSkipToEnd()' which is annotated with '@Test' and 'throws IOException'. It creates an 'InputStream' from a byte array, wraps it in a 'Base16InputStream', and performs two 'read()' operations, asserting that the first returns 6 and the second returns -1. The lines 390-395 are highlighted in pink, indicating the 'Eager Test' smell.

```
387     @Test
388     public void testSkipToEnd() throws IOException {
389         final InputStream ins = new ByteArrayInputStream(StringUtils.getBytesIso8859_1(ENCODED_B16));
390         try (final Base16InputStream b16Stream = new Base16InputStream(ins)) {
391             // due to CODEC-130, skip now skips correctly decoded characters rather than encoded
392             assertEquals(6, b16Stream.skip(6));
393             // End of stream reached
394             assertEquals(-1, b16Stream.read());
395             assertEquals(-1, b16Stream.read());
396         }
397     }
```

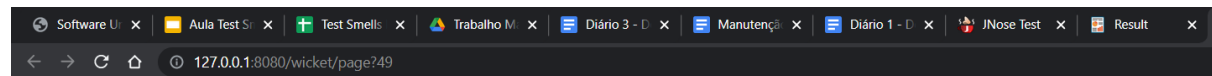
Teste refatorado



The screenshot shows a code editor with a dark background. It displays two versions of the 'testSkipToEnd()' method. The first version (lines 1-9) is the original code with the 'Eager Test' smell. The second version (lines 11-17) is the refactored code, which has been simplified by removing the 'try' block and the 'Base16InputStream' wrapper, directly using the 'InputStream' for the 'read()' operations. The refactored code is shown with line numbers 11 to 17.

```
1  @Test
2  public void testSkipToEnd() throws IOException {
3      final InputStream ins = new ByteArrayInputStream(StringUtils.getBytesIso8859_1(ENCODED_B16));
4      try (final Base16InputStream b16Stream = new Base16InputStream(ins)) {
5          // due to CODEC-130, skip now skips correctly decoded characters rather than encoded
6          assertEquals(6, b16Stream.skip(6));
7          // End of stream reached
8      }
9  }
10
11 public void testSkipToEnd() throws IOException {
12     final InputStream ins = new ByteArrayInputStream(StringUtils.getBytesIso8859_1(ENCODED_B16));
13     try (final Base16InputStream b16Stream = new Base16InputStream(ins)) {
14         assertEquals(-1, b16Stream.read());
15         assertEquals(-1, b16Stream.read());
16     }
17 }
```

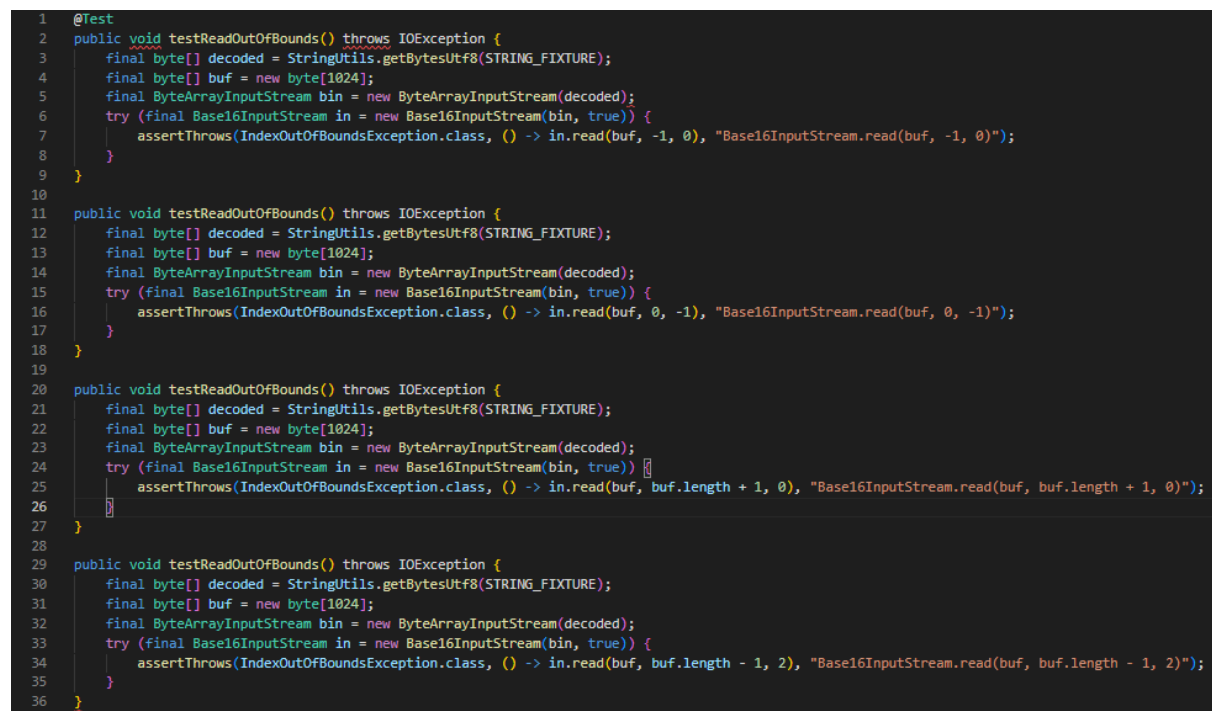
Eager Test - 5 (linha 323-327)



The screenshot shows a web browser with multiple tabs. The active tab is titled 'Code Test Smell: Eager Test [323, 327]'. The address bar shows the URL '127.0.0.1:8080/wicket/page?49'. The code displayed is a Java test method with line numbers 318 to 329. Lines 323 to 327 are highlighted in pink, indicating a test failure.

```
318  @Test
319  public void testReadOutOfBounds() throws IOException {
320      final byte[] decoded = StringUtils.getBytesUtf8(StringFixture);
321      final byte[] buf = new byte[1024];
322      final ByteArrayInputStream bin = new ByteArrayInputStream(decoded);
323      try (final Base16InputStream in = new Base16InputStream(bin, true)) {
324          assertThrows(IndexOutOfBoundsException.class, () -> in.read(buf, -1, 0), "Base16InputStream.read(buf, -1, 0)");
325          assertThrows(IndexOutOfBoundsException.class, () -> in.read(buf, 0, -1), "Base16InputStream.read(buf, 0, -1)");
326          assertThrows(IndexOutOfBoundsException.class, () -> in.read(buf, buf.length + 1, 0), "Base16InputStream.read(buf, buf.length + 1, 0)");
327          assertThrows(IndexOutOfBoundsException.class, () -> in.read(buf, buf.length - 1, 2), "Base16InputStream.read(buf, buf.length - 1, 2)");
328      }
329  }
```

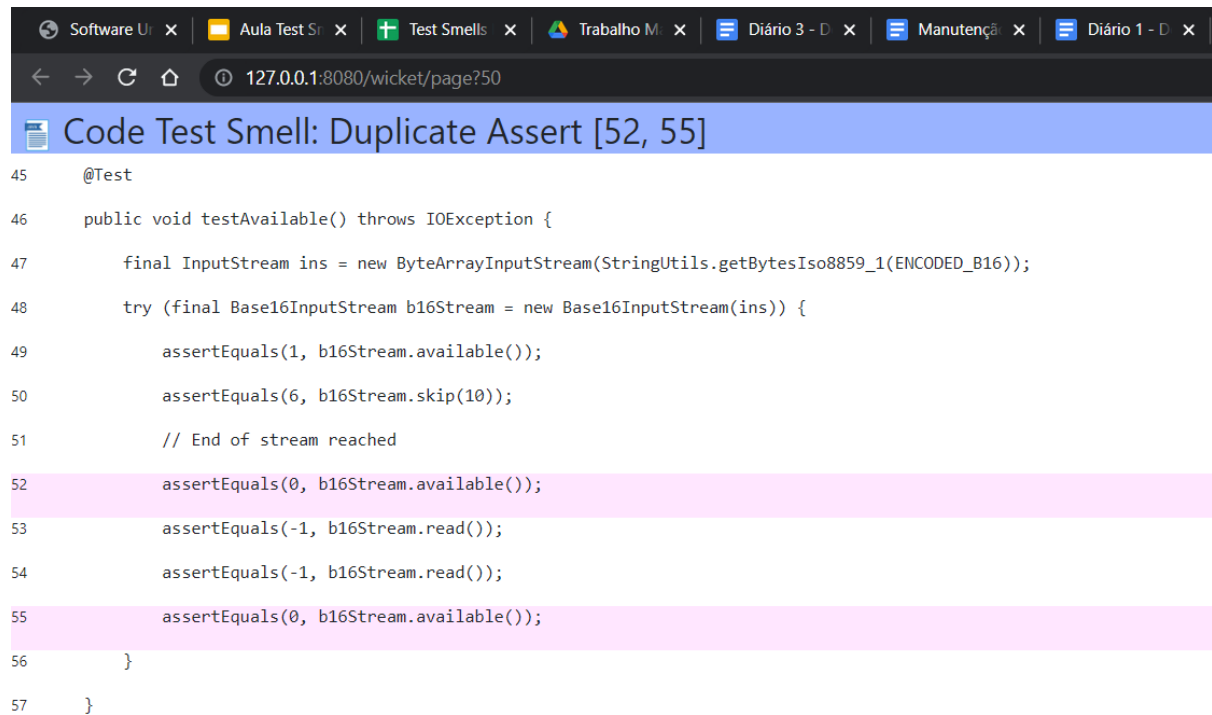
Teste refatorado



The screenshot shows a code editor with a refactored Java test method. The code is organized into four distinct blocks, each representing a different test case for the 'testReadOutOfBounds' method. The blocks are separated by empty lines and use consistent indentation for readability.

```
1  @Test
2  public void testReadOutOfBounds() throws IOException {
3      final byte[] decoded = StringUtils.getBytesUtf8(StringFixture);
4      final byte[] buf = new byte[1024];
5      final ByteArrayInputStream bin = new ByteArrayInputStream(decoded);
6      try (final Base16InputStream in = new Base16InputStream(bin, true)) {
7          assertThrows(IndexOutOfBoundsException.class, () -> in.read(buf, -1, 0), "Base16InputStream.read(buf, -1, 0)");
8      }
9  }
10
11 public void testReadOutOfBounds() throws IOException {
12     final byte[] decoded = StringUtils.getBytesUtf8(StringFixture);
13     final byte[] buf = new byte[1024];
14     final ByteArrayInputStream bin = new ByteArrayInputStream(decoded);
15     try (final Base16InputStream in = new Base16InputStream(bin, true)) {
16         assertThrows(IndexOutOfBoundsException.class, () -> in.read(buf, 0, -1), "Base16InputStream.read(buf, 0, -1)");
17     }
18 }
19
20 public void testReadOutOfBounds() throws IOException {
21     final byte[] decoded = StringUtils.getBytesUtf8(StringFixture);
22     final byte[] buf = new byte[1024];
23     final ByteArrayInputStream bin = new ByteArrayInputStream(decoded);
24     try (final Base16InputStream in = new Base16InputStream(bin, true)) {
25         assertThrows(IndexOutOfBoundsException.class, () -> in.read(buf, buf.length + 1, 0), "Base16InputStream.read(buf, buf.length + 1, 0)");
26     }
27 }
28
29 public void testReadOutOfBounds() throws IOException {
30     final byte[] decoded = StringUtils.getBytesUtf8(StringFixture);
31     final byte[] buf = new byte[1024];
32     final ByteArrayInputStream bin = new ByteArrayInputStream(decoded);
33     try (final Base16InputStream in = new Base16InputStream(bin, true)) {
34         assertThrows(IndexOutOfBoundsException.class, () -> in.read(buf, buf.length - 1, 2), "Base16InputStream.read(buf, buf.length - 1, 2)");
35     }
36 }
```

Duplicate Assertion - 1(linha 52-55)



The screenshot shows a web browser window with several tabs open. The address bar displays the URL `127.0.0.1:8080/wicket/page?50`. The main content area has a blue header with the text "Code Test Smell: Duplicate Assert [52, 55]". Below the header, a code snippet is displayed with line numbers 45 to 57. The code is a Java method `testAvailable()` that uses `assertEquals` to verify the state of a `Base16InputStream`. Lines 52 and 55 both contain `assertEquals(0, b16Stream.available());`, which is highlighted in pink to indicate a duplicate assertion.

```
45  @Test
46  public void testAvailable() throws IOException {
47      final InputStream ins = new ByteArrayInputStream(StringUtils.getBytesIso8859_1(ENCODED_B16));
48      try (final Base16InputStream b16Stream = new Base16InputStream(ins)) {
49          assertEquals(1, b16Stream.available());
50          assertEquals(6, b16Stream.skip(10));
51          // End of stream reached
52          assertEquals(0, b16Stream.available());
53          assertEquals(-1, b16Stream.read());
54          assertEquals(-1, b16Stream.read());
55          assertEquals(0, b16Stream.available());
56      }
57  }
```

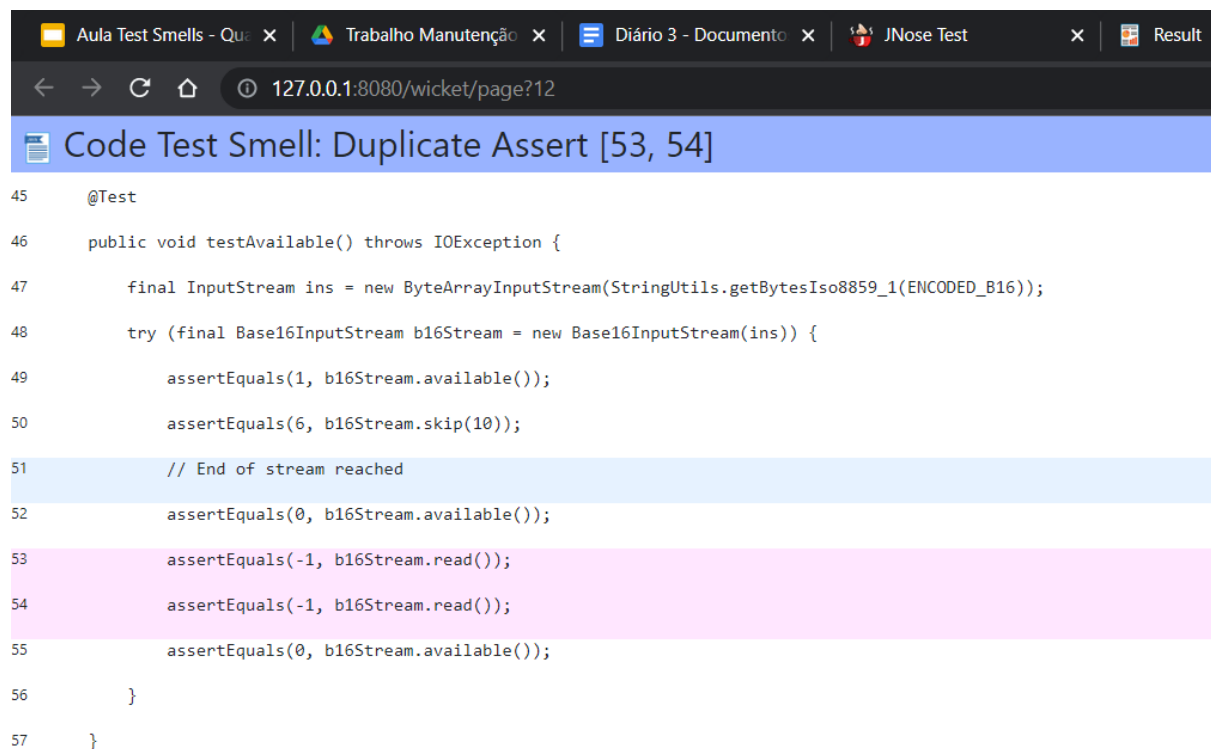
Teste refatorado



The screenshot shows an IDE window with a file named `PrimeiraEntrega.java`. The code is a Java method `testAvailable2()` that uses `assertEquals` to verify the state of a `Base16InputStream`. The code is refactored to avoid duplicate assertions. The method `testAvailable2()` is shown with line numbers 14 to 25. The code is as follows:

```
14  @Test
15  public void testAvailable2() throws IOException {
16      final InputStream ins = new ByteArrayInputStream(StringUtils.getBytesIso8859_1(ENCODED_B16));
17      try (final Base16InputStream b16Stream = new Base16InputStream(ins)) {
18          assertEquals(1, b16Stream.available());
19          assertEquals(6, b16Stream.skip(10));
20          // End of stream reached
21          assertEquals(-1, b16Stream.read());
22          assertEquals(-1, b16Stream.read());
23          assertEquals(0, b16Stream.available());
24      }
25  }
```

Duplicate Assertion - 2 (linha 53-54)

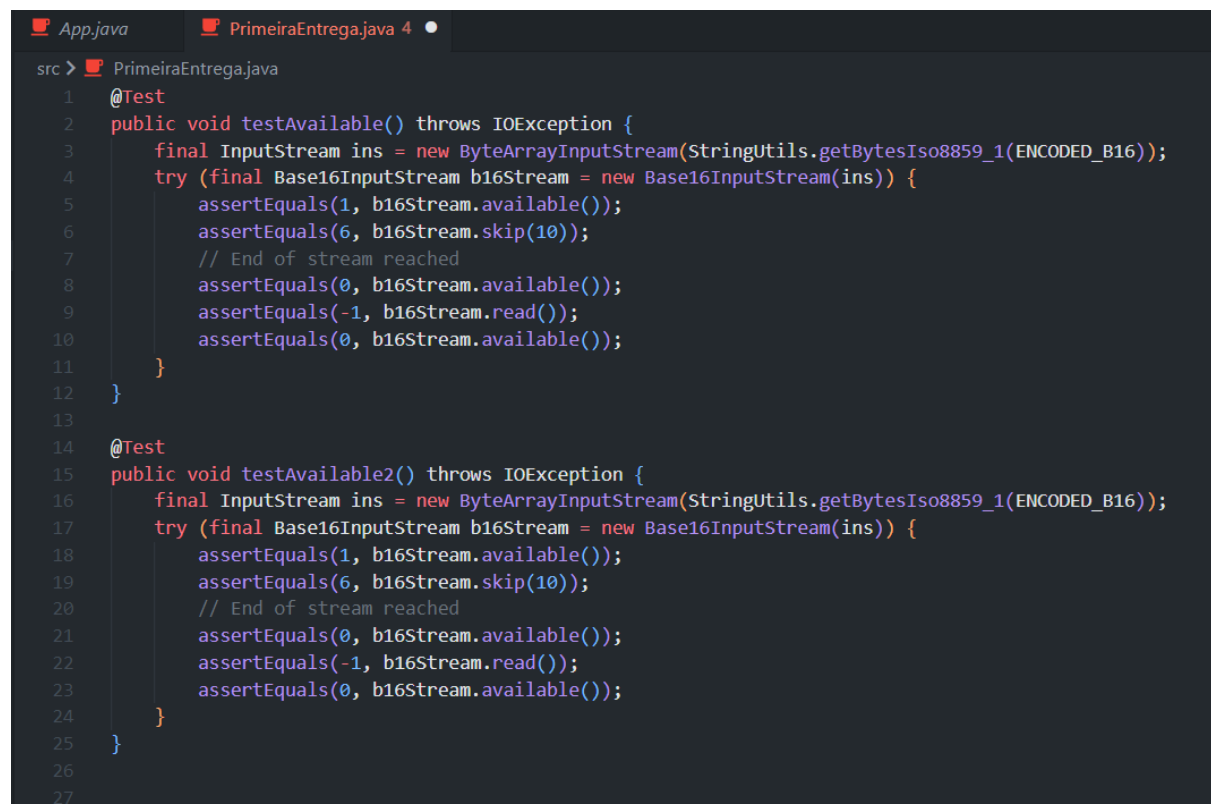


The screenshot shows a web browser window with the address bar displaying `127.0.0.1:8080/wicket/page?12`. The browser has several tabs open: "Aula Test Smells - Que...", "Trabalho Manutenção...", "Diário 3 - Documento...", "JNose Test", and "Result". The main content area displays a blue header with the text "Code Test Smell: Duplicate Assert [53, 54]". Below the header, the code for the `testAvailable()` method is shown. The code is as follows:

```
45  @Test
46  public void testAvailable() throws IOException {
47      final InputStream ins = new ByteArrayInputStream(StringUtils.getBytesIso8859_1(ENCODED_B16));
48      try (final Base16InputStream b16Stream = new Base16InputStream(ins)) {
49          assertEquals(1, b16Stream.available());
50          assertEquals(6, b16Stream.skip(10));
51          // End of stream reached
52          assertEquals(0, b16Stream.available());
53          assertEquals(-1, b16Stream.read());
54          assertEquals(-1, b16Stream.read());
55          assertEquals(0, b16Stream.available());
56      }
57  }
```

Lines 53 and 54 are highlighted in pink, indicating the duplicate assertion.

Teste Refatorado

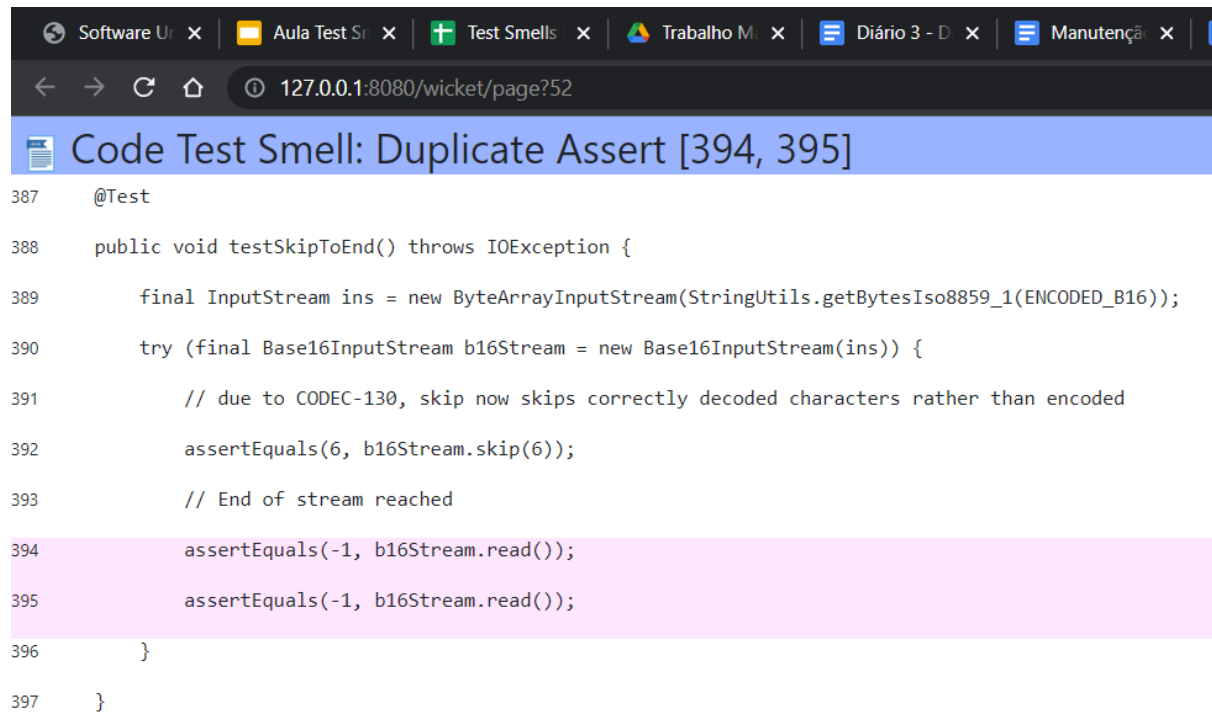


The screenshot shows an IDE window with the file `PrimeiraEntrega.java` open. The code is as follows:

```
1  @Test
2  public void testAvailable() throws IOException {
3      final InputStream ins = new ByteArrayInputStream(StringUtils.getBytesIso8859_1(ENCODED_B16));
4      try (final Base16InputStream b16Stream = new Base16InputStream(ins)) {
5          assertEquals(1, b16Stream.available());
6          assertEquals(6, b16Stream.skip(10));
7          // End of stream reached
8          assertEquals(0, b16Stream.available());
9          assertEquals(-1, b16Stream.read());
10         assertEquals(0, b16Stream.available());
11     }
12 }
13
14 @Test
15 public void testAvailable2() throws IOException {
16     final InputStream ins = new ByteArrayInputStream(StringUtils.getBytesIso8859_1(ENCODED_B16));
17     try (final Base16InputStream b16Stream = new Base16InputStream(ins)) {
18         assertEquals(1, b16Stream.available());
19         assertEquals(6, b16Stream.skip(10));
20         // End of stream reached
21         assertEquals(0, b16Stream.available());
22         assertEquals(-1, b16Stream.read());
23         assertEquals(0, b16Stream.available());
24     }
25 }
26
27
```

The code is identical to the one in the previous screenshot, but it is now part of a refactored test suite with two methods: `testAvailable()` and `testAvailable2()`.

Duplicate Assertion - 3 (linha 394-395)



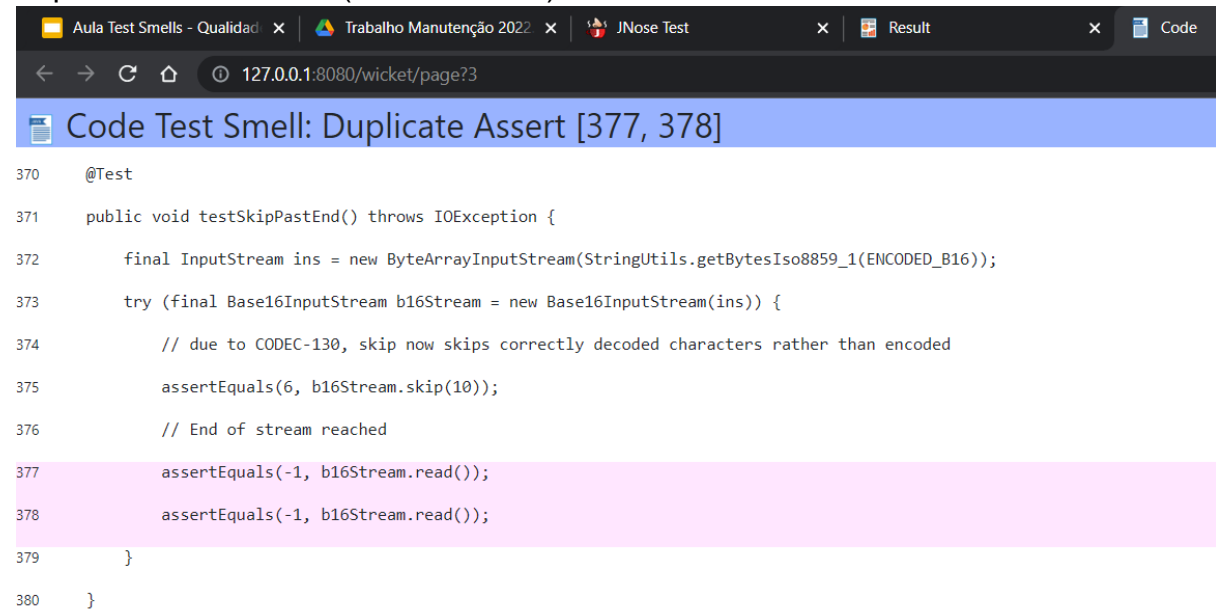
```
387  @Test
388  public void testSkipToEnd() throws IOException {
389      final InputStream ins = new ByteArrayInputStream(StringUtils.getBytesIso8859_1(ENCODED_B16));
390      try (final Base16InputStream b16Stream = new Base16InputStream(ins)) {
391          // due to CODEC-130, skip now skips correctly decoded characters rather than encoded
392          assertEquals(6, b16Stream.skip(6));
393          // End of stream reached
394          assertEquals(-1, b16Stream.read());
395          assertEquals(-1, b16Stream.read());
396      }
397  }
```

Teste refatorado



```
1  @Test
2  public void testSkipToEnd() throws IOException {
3      final InputStream ins = new ByteArrayInputStream(StringUtils.getBytesIso8859_1(ENCODED_B16));
4      try (final Base16InputStream b16Stream = new Base16InputStream(ins)) {
5          assertEquals(-1, b16Stream.read());
6      }
7  }
8
9  public void testSkipToEnd2() throws IOException {
10     final InputStream ins = new ByteArrayInputStream(StringUtils.getBytesIso8859_1(ENCODED_B16));
11     try (final Base16InputStream b16Stream = new Base16InputStream(ins)) {
12         // due to CODEC-130, skip now skips correctly decoded characters rather than encoded
13         assertEquals(6, b16Stream.skip(6));
14         // End of stream reached
15         assertEquals(-1, b16Stream.read());
16     }
17 }
```

Duplicate Assertion - 4 (linha 377-378)



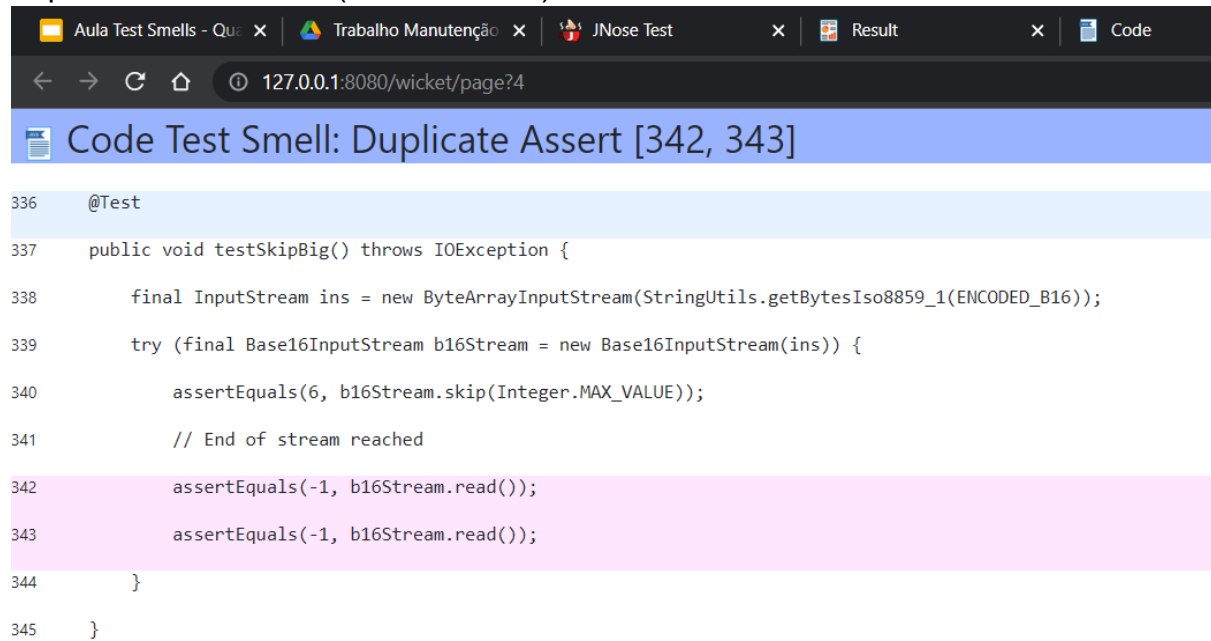
```
370  @Test
371  public void testSkipPastEnd() throws IOException {
372      final InputStream ins = new ByteArrayInputStream(StringUtils.getBytesIso8859_1(ENCODED_B16));
373      try (final Base16InputStream b16Stream = new Base16InputStream(ins)) {
374          // due to CODEC-130, skip now skips correctly decoded characters rather than encoded
375          assertEquals(6, b16Stream.skip(10));
376          // End of stream reached
377          assertEquals(-1, b16Stream.read());
378          assertEquals(-1, b16Stream.read());
379      }
380  }
```

Teste refatorado



```
src > PrimeiraEntrega.java
1  @Test
2  public void testSkipPastEnd() throws IOException {
3      final InputStream ins = new ByteArrayInputStream(StringUtils.getBytesIso8859_1(ENCODED_B16));
4      try (final Base16InputStream b16Stream = new Base16InputStream(ins)) {
5          // due to CODEC-130, skip now skips correctly decoded characters rather than encoded
6          assertEquals(6, b16Stream.skip(10));
7          // End of stream reached
8          assertEquals(-1, b16Stream.read());
9      }
10 }
11
12 @Test
13 public void testSkipPastEnd2() throws IOException {
14     final InputStream ins = new ByteArrayInputStream(StringUtils.getBytesIso8859_1(ENCODED_B16));
15     try (final Base16InputStream b16Stream = new Base16InputStream(ins)) {
16         // due to CODEC-130, skip now skips correctly decoded characters rather than encoded
17         assertEquals(6, b16Stream.skip(10));
18         // End of stream reached
19         assertEquals(-1, b16Stream.read());
20     }
21 }
```

Duplicate Assertion - 5 (linha 342-343)



```
336  @Test
337  public void testSkipBig() throws IOException {
338      final InputStream ins = new ByteArrayInputStream(StringUtils.getBytesIso8859_1(ENCODED_B16));
339      try (final Base16InputStream b16Stream = new Base16InputStream(ins)) {
340          assertEquals(6, b16Stream.skip(Integer.MAX_VALUE));
341          // End of stream reached
342          assertEquals(-1, b16Stream.read());
343          assertEquals(-1, b16Stream.read());
344      }
345  }
```

Teste refatorado:



```
src > PrimeiraEntrega.java
1  @Test
2  public void testSkipBig() throws IOException {
3      final InputStream ins = new ByteArrayInputStream(StringUtils.getBytesIso8859_1(ENCODED_B16));
4      try (final Base16InputStream b16Stream = new Base16InputStream(ins)) {
5          assertEquals(6, b16Stream.skip(Integer.MAX_VALUE));
6          // End of stream reached
7          assertEquals(-1, b16Stream.read());
8      }
9  }
10
11 @Test
12 public void testSkipBig2() throws IOException {
13     final InputStream ins = new ByteArrayInputStream(StringUtils.getBytesIso8859_1(ENCODED_B16));
14     try (final Base16InputStream b16Stream = new Base16InputStream(ins)) {
15         assertEquals(6, b16Stream.skip(Integer.MAX_VALUE));
16         // End of stream reached
17         assertEquals(-1, b16Stream.read());
18     }
19 }
20
```