Apresentação

Neste material você encontrará a definição e um exemplo do conjunto de Test Smells (TS). As definições de TS usadas neste material são traduções das apresentadas no mapeamento sistemático de literatura publicado por Aljedaani et al. (2021).

Os exemplos foram retirados de trechos de código de projetos open source escritos na linguagem Java e que usam o framework de testes JUnit.

Para saber mais:

1) Aljedaani, Wajdi; Peruma, Anthony; Aljohani, Ahmed; Alotaibi, Mazen; Wiem Mkaouer, Mohamed; Ouni, Ali; Newman, Christian D.; Ghallab, Abdullatif; Ludi, Stephanie. 2021. Test Smell Detection Tools: A Systematic Mapping Study. In Evaluation and Assessment in Software Engineering (EASE 2021) June 21–23, 2021, Trondheim, Norway. ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3463274.3463335

2) JNose Test: Java TestSmells Detection
https://jnosetest.github.io

Sumário

1 Assertion Roulette

Definição: um método de teste com múltiplas afirmações sem mensagens explicativas.

Exemplo: na imagem abaixo, as afirmações das linhas 56 e 57 não possuem explicação.

```
49      @Test
        public void testCodec() throws IOException {
51        int inputSize = 500_000;
52
53        byte[] input = generateTestData(inputSize);
54
55        Codec codecInstance = CodecFactory.fromString( s: codec).createInstance();
56        assertTrue( condition: codecClass.isInstance( obj: codecInstance));
57        assertTrue( condition: codecInstance.getName().equals( anObject: codec));
58
59        ByteBuffer inputByteBuffer = ByteBuffer.wrap( array: input);
60        ByteBuffer compressedBuffer = codecInstance.compress( uncompressedData: inputByteBuffer);
```

Figura 1: Classe TestAllCodecs.java do projeto Apache Avro, versão 1.11.1

2 Conditional Test Logic

Definição: um método de teste que contém uma instrução condicional como pré-requisito para executar a instrução de teste.

Exemplo: nas linhas 283 e 286 da figura abaixo podem ser observadas uma estrutura de repetição e uma de seleção, respectivamente.

```
274     @Test
        public void testSettersCreatedByDefault() throws IOException {
276         SpecificCompiler compiler = createCompiler();
277         assertTrue( condition: compiler.isCreateSetters());
278         compiler.compileToDestination( src: this.src,  dst: this.OUTPUT_DIR.getRoot());
279         assertTrue( condition: this.outputFile.exists());
280         int foundSetters = 0;
281         try (BufferedReader reader = new BufferedReader(new FileReader( file: this.outputFile))) {
282             String line;
283             while ((line = reader.readLine()) != null) {
284                 // We should find the setter in the main class
285                 line = line.trim();
286                 if (line.startsWith( prefix: "public void setValue(")) {
287                     foundSetters++;
288                 }
289             }
290         }
291         assertEquals( message: "Found the wrong number of setters",  expected: 1,  actual: foundSetters);
292     }
```

Figura 2: Classe de teste TestSpecificCompiler.java do projeto Apache Avro, versão 1.11.1

3 Constructor Initialization

Definição: uma classe de teste que contém um construtor.

Exemplo: na figura abaixo pode ser observado o construtor declarado na linha 31.

```
29  class MemoryTypeSolverTest extends AbstractTypeSolverTest<MemoryTypeSolver> {
30
31      public MemoryTypeSolverTest() {
32          super(MemoryTypeSolver::new);
33      }
34
35      /**
36       * When solving a type that isn't registered in the memory should fail, while
37       * a existing type should be solved.
38       */
39      @Test
        void solveNonExistentShouldFailAndExistentTypeShouldSolve() {
```

Figura 3: Classe de teste MemoryTypeSolverTest.java do projeto JavaParser, versão 3.25.1

4 Duplicate Assert

Definição: ocorre quando um método de teste tem a afirmação exata diversas vezes no mesmo método de teste.

Exemplo: na figura abaixo, as afirmações das linhas 71, 77 e 82 são iguais. O mesmo ocorre nas linhas 75, 80 e 85.

```
67      @Test
        public void testEquals() {
69          BorderArrangement b1 = new BorderArrangement();
70          BorderArrangement b2 = new BorderArrangement();
71          assertTrue( condition:b1.equals( obj:b2));
72          assertTrue( condition:b2.equals( obj:b1));
73
74          b1.add(new EmptyBlock( width:99.0,  height:99.0),  key:null);
75          assertFalse( condition:b1.equals( obj:b2));
76          b2.add(new EmptyBlock( width:99.0,  height:99.0),  key:null);
77          assertTrue( condition:b1.equals( obj:b2));
78
79          b1.add(new EmptyBlock( width:1.0,  height:1.0),  key:RectangleEdge.LEFT);
80          assertFalse( condition:b1.equals( obj:b2));
81          b2.add(new EmptyBlock( width:1.0,  height:1.0),  key:RectangleEdge.LEFT);
82          assertTrue( condition:b1.equals( obj:b2));
83
84          b1.add(new EmptyBlock( width:2.0,  height:2.0),  key:RectangleEdge.RIGHT);
85          assertFalse( condition:b1.equals( obj:b2));
86          b2.add(new EmptyBlock( width:2.0,  height:2.0),  key:RectangleEdge.RIGHT);
87          assertTrue( condition:b1.equals( obj:b2));
88
89          b1.add(new EmptyBlock( width:3.0,  height:3.0),  key:RectangleEdge.TOP);
90          assertFalse( condition:b1.equals( obj:b2));
91          b2.add(new EmptyBlock( width:3.0,  height:3.0),  key:RectangleEdge.TOP);
92          assertTrue( condition:b1.equals( obj:b2));
```

Figura 4: Classe de teste BorderArrangementTest.java do projeto JFreeChart, versão 1.5.3, afetada pelo Duplicate Assert.

5 Eager Test

Definição: um método de teste que chama vários métodos do objeto a ser testado.

Exemplo: na figura abaixo, é possível verificar que o sistema é estimulado nas linhas 74, 76, 79, 81, entre outras, por meio da chamada do método add(). Após cada estímulo, um teste é realizado na sequência (linhas 75, 77, 80 e 82, entre outras).

```
67      @Test
        public void testEquals() {
69          BorderArrangement b1 = new BorderArrangement();
70          BorderArrangement b2 = new BorderArrangement();
71          assertTrue( condition:b1.equals( obj:b2));
72          assertTrue( condition:b2.equals( obj:b1));
73
74          b1.add(new EmptyBlock( width:99.0,  height:99.0),  key:null);
75          assertFalse( condition:b1.equals( obj:b2));
76          b2.add(new EmptyBlock( width:99.0,  height:99.0),  key:null);
77          assertTrue( condition:b1.equals( obj:b2));
78
79          b1.add(new EmptyBlock( width:1.0,  height:1.0),  key:RectangleEdge.LEFT);
80          assertFalse( condition:b1.equals( obj:b2));
81          b2.add(new EmptyBlock( width:1.0,  height:1.0),  key:RectangleEdge.LEFT);
82          assertTrue( condition:b1.equals( obj:b2));
83
84          b1.add(new EmptyBlock( width:2.0,  height:2.0),  key:RectangleEdge.RIGHT);
85          assertFalse( condition:b1.equals( obj:b2));
86          b2.add(new EmptyBlock( width:2.0,  height:2.0),  key:RectangleEdge.RIGHT);
87          assertTrue( condition:b1.equals( obj:b2));
88
89          b1.add(new EmptyBlock( width:3.0,  height:3.0),  key:RectangleEdge.TOP);
90          assertFalse( condition:b1.equals( obj:b2));
91          b2.add(new EmptyBlock( width:3.0,  height:3.0),  key:RectangleEdge.TOP);
92          assertTrue( condition:b1.equals( obj:b2));
```

Figura 5: Figura 4: Classe de teste BorderArrangementTest.java do projeto JFreeChart, versão 1.5.3, afetada pelo Eager Test.

6 Empty Test

Definição: um método de teste vazio ou que não possui instruções executáveis.

Exemplo: na figura abaixo pode ser observado um método com a anotação @Test que não contém instruções.

```
18  public class OptionTokenizerTest {
19
20      @Test
        public void testEmpty() {
22
23      }
24
25      //
26      // @Test
27      // public void testEmpty() throws ScanException {
```

Figura 6: Classe OptionTokenizerTest.java do projeto Logback, versão 1.4.5, afetada pelo Eager Test

7 Exception Handling

Definição: ocorre quando o tratamento de exceções personalizado é utilizado em vez do recurso de tratamento de exceções do JUnit.

Exemplo: na figura abaixo, pode ser observado o tratamento de exceção das linhas 92 (try) até 102 (catch).

```
86      /**
87       * Draws the chart with a null info object to make sure that no exceptions
88       * are thrown (a problem that was occurring at one point).
89       */
90      @Test
92      public void testDrawWithNullInfo() {
93          try {
94              BufferedImage image = new BufferedImage( width: 200 ,  height: 100,
95                      imageType: BufferedImage.TYPE_INT_RGB);
96              Graphics2D g2 = image.createGraphics();
97              this.chart.draw(g2, new Rectangle2D.Double( x: 0,  y: 0,  w: 200,  h: 100),  anchor: null,
98                      info: null);
99              g2.dispose();
100         }
101         catch (Exception e) {
102             fail( message: "There should be no exception.");
103         }
104     }
```

Figura 7: Classe de teste BarChartTest.java do projeto JFreeChart, versão, 1.5.3, afetada pelo Exception Handling

8 General Fixture

Definição: esse smell surge quando o fixture setUp() cria muitos objetos e os métodos de teste usam apenas um subconjunto.

Exemplo: na figura abaixo, o método “setup” instancia dois objetos nas linhas 38 e 39. Porém, somente o objeto instanciado na linha 39 é referenciado no teste da linha 53.

```
36      @BeforeEach
37      public void setUp() throws Exception {
38          lc = new LoggerContext();
39          converter = new MDCConverter();
40          converter.start();
41          MDC.clear();
42      }
43
44      @AfterEach
45      public void tearDown() throws Exception {...6 lines }
51
52      @Test
54      public void testConvertWithOneEntry() {
55          String k = "MDCConverterTest_k" + diff;
56          String v = "MDCConverterTest_v" + diff;
57
58          MDC.put( key: k,  val: v);
59          ILoggingEvent le = createLoggingEvent();
60          String result = converter.convert( event: le);
61          assertEquals(k + "=" + v,  actual: result);
        }
```

Figura 8: MDCConverterTest.java do projeto Logback, versão 1.4.5, afetado pelo General Fixture

9 Ignored Test

Definição: um método de teste que usa uma anotação de ignorar que impede a execução do método de teste.

Exemplo: na imagem abaixo é possível identificar a anotação @Ignore no método de teste “testBuilderPerformance” na linha 177.

```
177     @Ignore
178     @Test
        public void testBuilderPerformance() {
180         int count = 1000000;
            List<Person> friends = new ArrayList<>( initialCapacity: 0);
182         List<String> languages = new ArrayList<>( c: Arrays.asList( a: "English",  a: "Java"));
183         long startTimeNanos = System.nanoTime();
184         for (int ii = 0; ii < count; ii++) {
                Person.newBuilder().setName("James Gosling").setYearOfBirth(1955).setCountry("US").setState("CA")
186                 .setFriends(friends).setLanguages(languages).build();
187         }
188         long durationNanos = System.nanoTime() - startTimeNanos;
189         double durationMillis = durationNanos / 1e6d;
190         System.out.println("Built " + count + " records in " + durationMillis + "ms (" + (count / (durationMillis / 1000d))
191             + " records/sec, " + (durationMillis / count) + "ms/record");
192     }
```

Figura 9: A classe TestSpecificRecordBuilder.java do projeto Apache Avro, versão 1.11.1, afetada pelo Ignored Test.

10 Lazy Test

Definição: ocorre quando vários métodos de teste verificam o mesmo método de objeto de produção.

Exemplo: na figura abaixo, o método “equals” é invocado no método de teste “testEquals”, linhas 62, 66 e 68, e no método de teste “testCloning”, na linha 81.

```
55      /** Confirm that the equals method can distinguish all the required fields ...3 lines */
58      @Test
        public void testEquals() {
60          DialCap c1 = new DialCap();
61          DialCap c2 = new DialCap();
62          assertTrue( condition: c1.equals( obj: c2));
63
64          // visible
65          c1.setVisible( visible: false);
66          assertFalse( condition: c1.equals( obj: c2));
67          c2.setVisible( visible: false);
68          assertTrue( condition: c1.equals( obj: c2));
69      }
70
71      /** Confirm that cloning works ...3 lines */
74      @Test
        public void testCloning() throws CloneNotSupportedException {
76          // test a default instance
77          DialCap c1 = new DialCap();
78          DialCap c2 = (DialCap) c1.clone();
79          assertTrue(c1 != c2);
80          assertTrue(c1.getClass() == c2.getClass());
81          assertTrue( condition: c1.equals( obj: c2));
```

Figura 10: Na classe AbstractDialLayerTest.java do projeto JFreeChart, versão 1.5.3, afetada pelo Lazy Test.

11 Magic Number Test

Definição: um método de teste que contém valores numéricos não documentados.

Exemplo: na figura abaixo é possível identificar constantes numéricas nas linhas 136, 147 e 150.

```java
@Test
public void testReplaceDataset() {
    Number[][] data = new Integer[][]{{-30, -20}, {-10, 10}, {20, 30}};

    CategoryDataset newData = DatasetUtils.createCategoryDataset(
        rowKeyPrefix:"S", columnKeyPrefix:"C", data);
    LocalListener l = new LocalListener();
    this.chart.addChangeListener( listener:l);
    CategoryPlot plot = (CategoryPlot) this.chart.getPlot();
    plot.setDataset( dataset:newData);
    assertEquals( expected:true, actual:l.flag);
    ValueAxis axis = plot.getRangeAxis();
    Range range = axis.getRange();
    assertTrue(range.getLowerBound() <= -30,
        "Expecting the lower bound of the range to be around -30: "
        + range.getLowerBound());
    assertTrue(range.getUpperBound() >= 30,
        "Expecting the upper bound of the range to be around 30: "
        + range.getUpperBound());
}
```

Figura 11: A classe AreaChartTest.java do projeto JFreeChart, versão 1.5.3, afetada pelo Magic Number Test

12 Mystery Guest

Definição: um teste que utiliza recursos externos, como um banco de dados, que contém dados de teste.

Exemplo: na figura abaixo é possível identificar a criação de um arquivo na linha 118.

```java
@Test
public void testDirConcat() throws Exception {
    Map<String, String> metadata = new HashMap<>();

    for (int i = 0; i < 3; i++) {
        generateData(name.getMethodName() + "-" + i + ".avro", type:Type.STRING, metadata, codec:DEFLATE);
    }

    File output = new File( parent:OUTPUT_DIR.getRoot(), name.getMethodName() + ".avro");

    List<String> args = asList( a:INPUT_DIR.getRoot().getAbsolutePath(), a:output.getAbsolutePath());
    int returnCode = new ConcatTool().run( in:System.in, out:System.out, err:System.err, args);

    assertEquals( expected:0, actual:returnCode);
    assertEquals(ROWS_IN_INPUT_FILES * 3, actual:numRowsInFile(output));
}
```

Figura 12: A classe TestConcatTool.java do projeto Apache Avro, versão 1.11.1, afetada pelo Mystery Guest.

13 Redundant Assertion

Definição: um método de teste que possui uma afirmação que é permanentemente verdadeira ou falsa.

Exemplo: na figura abaixo é possível identificar que as afirmações usam as mesmas constantes na comparação de igualdade, o que sempre produz o valor lógico verdadeiro.

```java
@Test
public void testEquals() {
    assertEquals( expected:AreaRendererEndType.LEVEL, actual:AreaRendererEndType.LEVEL);
    assertEquals( expected:AreaRendererEndType.TAPER, actual:AreaRendererEndType.TAPER);
    assertEquals( expected:AreaRendererEndType.TRUNCATE, actual:AreaRendererEndType.TRUNCATE);
}
```

Figura 13: A classe AreaRendererEndTypeTest.java do projeto JFreeChart, versão 1.5.3, afetada pelo Redundant Assertion.

14 Redundant Print

Definição: um método de teste que possui instrução print.

Exemplo: na figura abaixo, é possível identificar o uso da instrução print nas linhas 270, 274 e 277.

```
267     @Test
        public void testSort1() {
269         DoubleColumn numberColumn = DoubleColumn.create( name: "test",  initialSize: 1_000_000_000);
270         System.out.println( x: "Adding doubles to column");
271         for (int i = 0; i < 100_000_000; i++) {
272             numberColumn.append( d: Math.random());
273         }
274         System.out.println( x: "Sorting");
275         Stopwatch stopwatch = Stopwatch.createStarted();
276         numberColumn.sortAscending();
277         System.out.println("Sort time in ms = " + stopwatch.elapsed( desiredUnit: TimeUnit.MILLISECONDS));
278     }
```

Figura 14: A classe NumberColumnTest.java do projeto TableSaw, versão 0.43.1, afetada pelo Redundant Print

15 Resource Optimism

Definição: um teste que faz uma suposição sobre a existência de recursos externos.

Exemplo: na figura abaixo, ocorre a instanciação de um objeto da classe “File” na linha 118. Porém, os métodos do objeto são invocados sem identificar se o arquivo existe.

```
110     @Test
        public void testDirConcat() throws Exception {
112         Map<String, String> metadata = new HashMap<>();
113
114         for (int i = 0; i < 3; i++) {
115             generateData(name.getMethodName() + "-" + i + ".avro",  type: Type.STRING, metadata,  codec: DEFLATE);
116         }
117
118         File output = new File( parent: OUTPUT_DIR.getRoot(), name.getMethodName() + ".avro");
119
120         List<String> args = asList( a: INPUT_DIR.getRoot().getAbsolutePath(),  a: output.getAbsolutePath());
121         int returnCode = new ConcatTool().run( in: System.in,  out: System.out,  err: System.err, args);
122
123         assertEquals( expected: 0,  actual: returnCode);
124         assertEquals(ROWS_IN_INPUT_FILES * 3,  actual: numRowsInFile(output));
125     }
```

Figura 15: A classe TestConcatTool.java do projeto Apache Avro, versão 1.11.1, afetada pelo Resource Optimism.

16 Sensitive Equality

Definição: ocorre quando uma asserção tem uma verificação de igualdade usando o método toString

Exemplo: na figura abaixo, é possível identificar o método toString sendo invocado nas linhas em destaque.

```
124     @Test
        public void declaredVsNonDeclaredMethods() {
126         try (ScanResult scanResult = new ClassGraph().enableAllInfo()
127             .acceptPackages( packageNames: DeclaredVsNonDeclaredTest.class.getPackage().getName()).scan()) {
128             final ClassInfo A = scanResult.getClassInfo( className: A.class.getName());
129             final ClassInfo B = scanResult.getClassInfo( className: B.class.getName());
130             assertThat( actual: B.getFieldInfo( fieldName: "x").getClassInfo().getName()).isEqualTo( expected: B.class.getName());
131             assertThat( actual: B.getFieldInfo( fieldName: "z").getClassInfo().getName()).isEqualTo( expected: A.class.getName());
132             assertThat( actual: A.getFieldInfo().get( index: 0).getTypeDescriptor().toString()).isEqualTo( expected: "float");
133             assertThat( actual: B.getFieldInfo().get( index: 0).getTypeDescriptor().toString()).isEqualTo( expected: "int");
134             assertThat( actual: B.getMethodInfo().toString())
135                 .isEqualTo("[void y(final int x, final int y), void w(), abstract void y(java.lang.String x), "
136                     + "abstract void y(java.lang.Integer x)]");
137             assertThat( actual: B.getDeclaredMethodInfo().toString())
138                 .isEqualTo( expected: "[void y(final int x, final int y), void w()]");
139         }
140     }
```

Figura 16: A classe CompositeTitleTest.java do JFreeChart, versão 1.5.3, afetada pelo Sensitive Equality (em destaque)

17 Sleepy Test

Definição: ocorre quando um método de teste possui uma espera explícita (Thread.sleep()).

Exemplo: na figura abaixo, é possível identificar a invocação do método “sleep” na linha 163.

```
156     @Test
        public void testTryLock_Unlock5() throws Exception {
158         when( methodCall: cache.PUT_IF_ABSENT( key: any(), value: any(), expireAfterWrite: anyLong(), timeUnit: any())).then(i ->
159                 //change expire time
160             concreteCache.PUT_IF_ABSENT( key: "key", value: i.getArgument( i: 1).toString(), expireAfterWrite: 100, timeUnit: TimeUnit.HOURS)
161         );
            AutoReleaseLock lock = cache.tryLock( key: "key", expire: 1, timeUnit: TimeUnit.MILLISECONDS);
163         Thread.sleep( millis: 2);
164         lock.close();
165         assertNotNull( actual: concreteCache.GET( key: "key"));
166     }
```

Figura 17: A classe CacheTest.java do projeto JetCache, versão 2.7.3, afetada pelo Sleepy Test.

18 Test Code Duplication

Definição: ocorre quando os clones de código estão contidos no teste.

Exemplo: na figura abaixo, os dois métodos de teste exibidos apresentam diferença entre as linhas 236 e 254.

```
226     @Test
        public void test2502355_zoomInDomain() {
228         DefaultXYDataset dataset = new DefaultXYDataset();
229         JFreeChart chart = ChartFactory.createXYLineChart( title: "TestChart", xAxisLabel: "X",
230                 yAxisLabel: "Y", dataset, orientation: PlotOrientation.VERTICAL, legend: false, tooltips: false, urls: false);
231         XYPlot plot = (XYPlot) chart.getPlot();
232         plot.setDomainAxis( index: 1, new NumberAxis( label: "X2"));
233         ChartPanel panel = new ChartPanel(chart);
234         chart.addChangeListener( listener: this);
235         this.chartChangeEvents.clear();
236         panel.zoomInDomain( x: 1.0, y: 2.0);
237         assertEquals( expected: 1, actual: this.chartChangeEvents.size());
238     }
239
240     /** Checks that a call to the zoomOutDomain() method, for a plot with more ...4 lines */
244     @Test
        public void test2502355_zoomOutDomain() {
246         DefaultXYDataset dataset = new DefaultXYDataset();
247         JFreeChart chart = ChartFactory.createXYLineChart( title: "TestChart", xAxisLabel: "X",
248                 yAxisLabel: "Y", dataset, orientation: PlotOrientation.VERTICAL, legend: false, tooltips: false, urls: false);
249         XYPlot plot = (XYPlot) chart.getPlot();
250         plot.setDomainAxis( index: 1, new NumberAxis( label: "X2"));
251         ChartPanel panel = new ChartPanel(chart);
252         chart.addChangeListener( listener: this);
253         this.chartChangeEvents.clear();
254         panel.zoomOutDomain( x: 1.0, y: 2.0);
255         assertEquals( expected: 1, actual: this.chartChangeEvents.size());
256     }
```

Figura 18: A classe ChartPanelTest.java do projeto JFreeChart, versão 1.5.3, afetada pelo Test Code Duplication

19 Unknown Test

Definição: um método de teste sem declaração de afirmação e nome não descritivo.

Exemplo: na figura abaixo, pode ser observado um método que não contém nenhuma afirmação.

```
62      @Test
        public void testDrawWithNullInfo() {
64          MeterPlot plot = new MeterPlot(new DefaultValueDataset( value: 60.0));
65          plot.addInterval(new MeterInterval( label: "Normal", new Range( lower: 0.0, upper: 80.0)));
66          JFreeChart chart = new JFreeChart(plot);
67          BufferedImage image = new BufferedImage( width: 200, height: 100,
68                  imageType: BufferedImage.TYPE_INT_RGB);
69          Graphics2D g2 = image.createGraphics();
70          chart.draw(g2, new Rectangle2D.Double( x: 0, y: 0, w: 200, h: 100), anchor: null, info: null);
71          g2.dispose();
72          //FIXME we should really assert a value here
73      }
```

Figura 19: A classe MeterChartTest.java do projeto JFreeChart, versão 1.5.3, afetada pelo Unknown Test

20 Verbose Test

Definição: teste código complexo e não simples ou limpo. Por padrão, o JNose considera um teste com 30 linhas ou mais como afetado pelo Verbose Test smell.

Exemplo: na figura abaixo, pode ser observada uma ocorrência do Verbose Test smell na qual o método de teste tem mais de 30 linhas.

```
103     @Test
        public void testGetListeners() {
105         ChartPanel p = new ChartPanel( chart:null);
106         p.addChartMouseListener( listener:this);
107         EventListener[] listeners = p.getListeners( listenerType:ChartMouseListener.class);
108         assertEquals( expected: 1,  actual: listeners.length);
109         assertEquals( expected: this, listeners[0]);
110         // try a listener type that isn't registered
111         listeners = p.getListeners( listenerType:CaretListener.class);
112         assertEquals( expected: 0,  actual: listeners.length);
113         p.removeChartMouseListener( listener:this);
114         listeners = p.getListeners( listenerType:ChartMouseListener.class);
115         assertEquals( expected: 0,  actual: listeners.length);
116
117         // try a null argument
118         boolean pass = false;
119         try {
            listeners = p.getListeners((Class) null);
121         }
122         catch (NullPointerException e) {
123             pass = true;
124         }
125         assertTrue( condition:pass);
126
127         // try a class that isn't a listener
128         pass = false;
129         try {
            listeners = p.getListeners( listenerType:Integer.class);
131         }
132         catch (ClassCastException e) {
133             pass = true;
134         }
135         assertTrue( condition:pass);
136     }
```

Figura 20: A classe ChartPanelTest.java do projeto JFreeChart, versão 1.5.3, afetada pelo Verbose Test.