# DataUtilities.java

```
1    /* ===========================================================
2     * JFreeChart : a free chart library for the Java(tm) platform
3     * ===========================================================
4     *
5     * (C) Copyright 2000-2013, by Object Refinery Limited and Contributors.
6     *
7     * Project Info:  http://www.jfree.org/jfreechart/index.html
8     *
9     * This library is free software; you can redistribute it and/or modify it
10    * under the terms of the GNU Lesser General Public License as published by
11    * the Free Software Foundation; either version 2.1 of the License, or
12    * (at your option) any later version.
13    *
14    * This library is distributed in the hope that it will be useful, but
15    * WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY
16    * or FITNESS FOR A PARTICULAR PURPOSE. See the GNU Lesser General Public
17    * License for more details.
18    *
19    * You should have received a copy of the GNU Lesser General Public
20    * License along with this library; if not, write to the Free Software
21    * Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA  02110-1301,
22    * USA.
23    *
24    * [Oracle and Java are registered trademarks of Oracle and/or its affiliates.
25    * Other names may be trademarks of their respective owners.]
26    *
27    * ------------------
28    * DataUtilities.java
29    * ------------------
30    * (C) Copyright 2003-2013, by Object Refinery Limited and contributors.
31    *
32    * Original Author:  David Gilbert (for Object Refinery Limited);
33    * Contributor(s):   Peter Kolb (patch 2511330);
34    *
35    * Changes
36    * -------
37    * 05-Mar-2003 : Version 1 (DG);
38    * 03-Mar-2005 : Moved createNumberArray() and createNumberArray2D() methods
39    *               from the DatasetUtilities class (DG);
40    * 17-May-2005 : Added calculateColumnTotal() and calculateRowTotal()
41    *               methods (DG);
42    * 28-Jan-2009 : Added equal(double[][], double[][]) method (DG);
43    * 28-Jan-2009 : Added clone(double[][]) method (DG);
44    * 04-Feb-2009 : Added calculateColumnTotal/RowTotal variants (PK);
45    * 03-Jul-2013 : Use ParamChecks (DG);
46    *
47    */
48
```

```
49   package org.jfree.data;
50
51   import java.util.Arrays;
52   import org.jfree.chart.util.ParamChecks;
53   import org.jfree.data.general.DatasetUtilities;
54
55   /**
56    * Utility methods for use with some of the data classes (but not the datasets,
57    * see {@link DatasetUtilities}).
58    */
59   public abstract class DataUtilities {
60
61       /**
62        * Tests two arrays for equality.  To be considered equal, the arrays must
63        * have exactly the same dimensions, and the values in each array must also
64        * match (two values that qre both NaN or both INF are considered equal
65        * in this test).
66        *
67        * @param a  the first array (<code>null</code> permitted).
68        * @param b  the second array (<code>null</code> permitted).
69        *
70        * @return A boolean.
71        *
72        * @since 1.0.13
73        */
74       public static boolean equal(double[][] a, double[][] b) {
75 1         if (a == null) {
76 3             return (b == null);
77            }
78 1         if (b == null) {
79 1             return false;  // already know 'a' isn't null
80            }
81 1         if (a.length != b.length) {
82 1             return false;
83            }
84 2         for (int i = 0; i < a.length; i++) {
85 1             if (!Arrays.equals(a[i], b[i])) {
86 1                 return false;
87              }
88            }
89 1         return true;
90       }
91
92       /**
93        * Returns a clone of the specified array.
94        *
95        * @param source  the source array (<code>null</code> not permitted).
96        *
97        * @return A clone of the array.
98        *
99        * @since 1.0.13
100       */
101      public static double[][] clone(double[][] source) {
```

```
102 1         ParamChecks.nullNotPermitted(source, "source");
103           double[][] clone = new double[source.length][];
104 2         for (int i = 0; i < source.length; i++) {
105 1             if (source[i] != null) {
106                   double[] row = new double[source[i].length];
107 1                 System.arraycopy(source[i], 0, row, 0, source[i].length);
108                   clone[i] = row;
109               }
110           }
111 1         return clone;
112       }
113
114       /**
115        * Returns the total of the values in one column of the supplied data
116        * table.
117        *
118        * @param data  the table of values (<code>null</code> not permitted).
119        * @param column  the column index (zero-based).
120        *
121        * @return The total of the values in the specified column.
122        */
123       public static double calculateColumnTotal(Values2D data, int column) {
124 1         ParamChecks.nullNotPermitted(data, "data");
125           double total = 0.0;
126           int rowCount = data.getRowCount();
127 3         for (int r = 0; r < rowCount; r++) {
128               Number n = data.getValue(r, column);
129 1             if (n != null) {
130 1                 total += n.doubleValue();
131               }
132           }
133 1         return total;
134       }
135
136       /**
137        * Returns the total of the values in one column of the supplied data
138        * table by taking only the row numbers in the array into account.
139        *
140        * @param data  the table of values (<code>null</code> not permitted).
141        * @param column  the column index (zero-based).
142        * @param validRows the array with valid rows (zero-based).
143        *
144        * @return The total of the valid values in the specified column.
145        *
146        * @since 1.0.13
147        */
148       public static double calculateColumnTotal(Values2D data, int column,
149                int[] validRows) {
150 1         ParamChecks.nullNotPermitted(data, "data");
151           double total = 0.0;
152           int rowCount = data.getRowCount();
153 3         for (int v = 0; v < validRows.length; v++) {
154               int row = validRows[v];
```

```
155 2            if (row < rowCount) {
156                  Number n = data.getValue(row, column);
157 1                if (n != null) {
158 1                    total += n.doubleValue();
159                  }
160              }
161          }
162 1        return total;
163      }
164
165      /**
166       * Returns the total of the values in one row of the supplied data
167       * table.
168       *
169       * @param data  the table of values (<code>null</code> not permitted).
170       * @param row  the row index (zero-based).
171       *
172       * @return The total of the values in the specified row.
173       */
174      public static double calculateRowTotal(Values2D data, int row) {
175 1        ParamChecks.nullNotPermitted(data, "data");
176          double total = 0.0;
177          int columnCount = data.getColumnCount();
178 3        for (int c = 0; c < columnCount; c++) {
179              Number n = data.getValue(row, c);
180 1            if (n != null) {
181 1                total += n.doubleValue();
182              }
183          }
184 1        return total;
185      }
186
187      /**
188       * Returns the total of the values in one row of the supplied data
189       * table by taking only the column numbers in the array into account.
190       *
191       * @param data  the table of values (<code>null</code> not permitted).
192       * @param row  the row index (zero-based).
193       * @param validCols the array with valid cols (zero-based).
194       *
195       * @return The total of the valid values in the specified row.
196       *
197       * @since 1.0.13
198       */
199      public static double calculateRowTotal(Values2D data, int row,
200              int[] validCols) {
201 1        ParamChecks.nullNotPermitted(data, "data");
202          double total = 0.0;
203          int colCount = data.getColumnCount();
204 3        for (int v = 0; v < validCols.length; v++) {
205              int col = validCols[v];
206 2            if (col < colCount) {
207                  Number n = data.getValue(row, col);
```

```
208 1                if (n != null) {
209 1                    total += n.doubleValue();
210                  }
211              }
212          }
213 1        return total;
214      }
215
216      /**
217       * Constructs an array of <code>Number</code> objects from an array of
218       * <code>double</code> primitives.
219       *
220       * @param data  the data (<code>null</code> not permitted).
221       *
222       * @return An array of <code>Double</code>.
223       */
224      public static Number[] createNumberArray(double[] data) {
225 1        ParamChecks.nullNotPermitted(data, "data");
226          Number[] result = new Number[data.length];
227 2        for (int i = 0; i < data.length; i++) {
228              result[i] = new Double(data[i]);
229          }
230 1        return result;
231      }
232
233      /**
234       * Constructs an array of arrays of <code>Number</code> objects from a
235       * corresponding structure containing <code>double</code> primitives.
236       *
237       * @param data  the data (<code>null</code> not permitted).
238       *
239       * @return An array of <code>Double</code>.
240       */
241      public static Number[][] createNumberArray2D(double[][] data) {
242 1        ParamChecks.nullNotPermitted(data, "data");
243          int l1 = data.length;
244          Number[][] result = new Number[l1][];
245 3        for (int i = 0; i < l1; i++) {
246              result[i] = createNumberArray(data[i]);
247          }
248 1        return result;
249      }
250
251      /**
252       * Returns a {@link KeyedValues} instance that contains the cumulative
253       * percentage values for the data in another {@link KeyedValues} instance.
254       * <p>
255       * The percentages are values between 0.0 and 1.0 (where 1.0 = 100%).
256       *
257       * @param data  the data (<code>null</code> not permitted).
258       *
259       * @return The cumulative percentages.
260       */
```

```
261      public static KeyedValues getCumulativePercentages(KeyedValues data) {
262 1        ParamChecks.nullNotPermitted(data, "data");
263          DefaultKeyedValues result = new DefaultKeyedValues();
264          double total = 0.0;
265 2        for (int i = 0; i < data.getItemCount(); i++) {
266              Number v = data.getValue(i);
267 1            if (v != null) {
268 1                total = total + v.doubleValue();
269              }
270          }
271          double runningTotal = 0.0;
272 2        for (int i = 0; i < data.getItemCount(); i++) {
273              Number v = data.getValue(i);
274 1            if (v != null) {
275 1                runningTotal = runningTotal + v.doubleValue();
276              }
277 2            result.addValue(data.getKey(i), new Double(runningTotal / total));
278          }
279 1        return result;
280      }
281
282  }
```

## Mutations

| | |
|---|---|
| 75 | 1. negated conditional → KILLED |
| 76 | 1. replaced boolean return with false for org/jfree/data/DataUtilities::equal → KILLED<br>2. replaced boolean return with true for org/jfree/data/DataUtilities::equal → KILLED<br>3. negated conditional → KILLED |
| 78 | 1. negated conditional → KILLED |
| 79 | 1. replaced boolean return with true for org/jfree/data/DataUtilities::equal → KILLED |
| 81 | 1. negated conditional → KILLED |
| 82 | 1. replaced boolean return with true for org/jfree/data/DataUtilities::equal → KILLED |
| 84 | 1. changed conditional boundary → KILLED<br>2. negated conditional → KILLED |
| 85 | 1. negated conditional → KILLED |
| 86 | 1. replaced boolean return with true for org/jfree/data/DataUtilities::equal → KILLED |
| 89 | 1. replaced boolean return with false for org/jfree/data/DataUtilities::equal → KILLED |
| 102 | 1. removed call to org/jfree/chart/util/ParamChecks::nullNotPermitted → SURVIVED |
| 104 | 1. changed conditional boundary → KILLED<br>2. negated conditional → KILLED |
| 105 | 1. negated conditional → KILLED |
| 107 | 1. removed call to java/lang/System::arraycopy → KILLED |
| 111 | 1. replaced return value with null for org/jfree/data/DataUtilities::clone → KILLED |
| 124 | 1. removed call to org/jfree/chart/util/ParamChecks::nullNotPermitted → SURVIVED |
| 127 | 1. changed conditional boundary → KILLED |

| | |
|---|---|
| | 2. Changed increment from 1 to -1 → KILLED<br>3. negated conditional → KILLED |
| 129 | 1. negated conditional → KILLED |
| 130 | 1. Replaced double addition with subtraction → KILLED |
| 133 | 1. replaced double return with 0.0d for org/jfree/data/DataUtilities::calculateColumnTotal → KILLED |
| 150 | 1. removed call to org/jfree/chart/util/ParamChecks::nullNotPermitted → SURVIVED |
| 153 | 1. changed conditional boundary → KILLED<br>2. Changed increment from 1 to -1 → KILLED<br>3. negated conditional → KILLED |
| 155 | 1. changed conditional boundary → SURVIVED<br>2. negated conditional → KILLED |
| 157 | 1. negated conditional → KILLED |
| 158 | 1. Replaced double addition with subtraction → KILLED |
| 162 | 1. replaced double return with 0.0d for org/jfree/data/DataUtilities::calculateColumnTotal → KILLED |
| 175 | 1. removed call to org/jfree/chart/util/ParamChecks::nullNotPermitted → SURVIVED |
| 178 | 1. changed conditional boundary → KILLED<br>2. Changed increment from 1 to -1 → KILLED<br>3. negated conditional → KILLED |
| 180 | 1. negated conditional → KILLED |
| 181 | 1. Replaced double addition with subtraction → KILLED |
| 184 | 1. replaced double return with 0.0d for org/jfree/data/DataUtilities::calculateRowTotal → KILLED |
| 201 | 1. removed call to org/jfree/chart/util/ParamChecks::nullNotPermitted → SURVIVED |
| 204 | 1. changed conditional boundary → KILLED<br>2. Changed increment from 1 to -1 → KILLED<br>3. negated conditional → KILLED |
| 206 | 1. changed conditional boundary → SURVIVED<br>2. negated conditional → KILLED |
| 208 | 1. negated conditional → KILLED |
| 209 | 1. Replaced double addition with subtraction → KILLED |
| 213 | 1. replaced double return with 0.0d for org/jfree/data/DataUtilities::calculateRowTotal → KILLED |
| 225 | 1. removed call to org/jfree/chart/util/ParamChecks::nullNotPermitted → SURVIVED |
| 227 | 1. changed conditional boundary → KILLED<br>2. negated conditional → KILLED |
| 230 | 1. replaced return value with null for org/jfree/data/DataUtilities::createNumberArray → KILLED |
| 242 | 1. removed call to org/jfree/chart/util/ParamChecks::nullNotPermitted → SURVIVED |
| 245 | 1. changed conditional boundary → KILLED<br>2. Changed increment from 1 to -1 → KILLED<br>3. negated conditional → KILLED |
| 248 | 1. replaced return value with null for org/jfree/data/DataUtilities::createNumberArray2D → KILLED |
| 262 | 1. removed call to org/jfree/chart/util/ParamChecks::nullNotPermitted → SURVIVED |
| 265 | 1. changed conditional boundary → KILLED<br>2. negated conditional → KILLED |
| 267 | 1. negated conditional → KILLED |
| 268 | 1. Replaced double addition with subtraction → KILLED |
| 272 | 1. changed conditional boundary → KILLED<br>2. negated conditional → KILLED |

```
274  1. negated conditional → KILLED
275  1. Replaced double addition with subtraction → KILLED
277  1. Replaced double division with multiplication → KILLED
     2. removed call to org/jfree/data/DefaultKeyedValues::addValue → KILLED
279  1. replaced return value with null for
     org/jfree/data/DataUtilities::getCumulativePercentages → KILLED
```

## Active mutators

- BOOLEAN_FALSE_RETURN
- BOOLEAN_TRUE_RETURN
- CONDITIONALS_BOUNDARY_MUTATOR
- EMPTY_RETURN_VALUES
- INCREMENTS_MUTATOR
- INVERT_NEGS_MUTATOR
- MATH_MUTATOR
- NEGATE_CONDITIONALS_MUTATOR
- NULL_RETURN_VALUES
- PRIMITIVE_RETURN_VALS_MUTATOR
- VOID_METHOD_CALL_MUTATOR

## Tests examined

- org.jfree.data.testA3.DataUtilitiesTest_v2.notEqualFor10By10ArrayTest (org.jfree.data.testA3.DataUtilitiesTest_v2) (1 ms)
- org.jfree.data.testA3.DataUtilitiesTest_v2.calculateRowTotalForOneValues (org.jfree.data.testA3.DataUtilitiesTest_v2) (3 ms)
- org.jfree.data.DataUtilitiesTest_v3.calculateColumnTotalForZeroValues(org.jfree.data.DataUtilitiesTest_v3) (1 ms)
- org.jfree.data.DataUtilitiesTest_v3.equalForInequalLengths(org.jfree.data.DataUtilitiesTest_v3) (0 ms)
- org.jfree.data.testA3.DataUtilitiesTest_v2.createNumberArrayOfSize1 (org.jfree.data.testA3.DataUtilitiesTest_v2) (0 ms)
- org.jfree.data.testA2.DataUtilitiesTest.notCloneFor10By10ArrayTest(org.jfree.data.testA2.DataUtilitiesTest) (1 ms)
- org.jfree.data.testA3.DataUtilitiesTest_v2.equalFor1By10ArrayTest(org.jfree.data.testA3.DataUtilitiesTest_v2) (0 ms)
- org.jfree.data.testA2.DataUtilitiesTest.getCumulativePercentagesForThreeValues (org.jfree.data.testA2.DataUtilitiesTest) (0 ms)
- org.jfree.data.DataUtilitiesTest_v3.calculateRowTotalForOneValues(org.jfree.data.DataUtilitiesTest_v3) (1 ms)
- org.jfree.data.testA2.DataUtilitiesTest.calculateColumnTotalForZeroValues (org.jfree.data.testA2.DataUtilitiesTest) (1 ms)
- org.jfree.data.testA2.DataUtilitiesTest.equalFor1By10ArrayTest(org.jfree.data.testA2.DataUtilitiesTest) (0 ms)
- org.jfree.data.DataUtilitiesTest_v3.calculateColumnTotalForNegativeValues (org.jfree.data.DataUtilitiesTest_v3) (54 ms)
- org.jfree.data.testA2.DataUtilitiesTest.calculateColumnTotalForOneValues (org.jfree.data.testA2.DataUtilitiesTest) (0 ms)
- org.jfree.data.testA3.DataUtilitiesTest_v2.getCumulativePercentagesForZeroValues (org.jfree.data.testA3.DataUtilitiesTest_v2) (0 ms)
- org.jfree.data.testA3.DataUtilitiesTest_v2.cloneForNullRowArrayTest (org.jfree.data.testA3.DataUtilitiesTest_v2) (1 ms)
- org.jfree.data.testA3.DataUtilitiesTest_v2.equalFor10By10ArrayTest (org.jfree.data.testA3.DataUtilitiesTest_v2) (0 ms)
- org.jfree.data.testA3.DataUtilitiesTest_v2.createNumberArray2DFor1By1Array (org.jfree.data.testA3.DataUtilitiesTest_v2) (0 ms)
- org.jfree.data.testA2.DataUtilitiesTest.createNumberArray2DFor10By10Array (org.jfree.data.testA2.DataUtilitiesTest) (12 ms)
- org.jfree.data.testA3.DataUtilitiesTest_v2.calculateColumnTotalForZeroValues (org.jfree.data.testA3.DataUtilitiesTest_v2) (1 ms)
- org.jfree.data.DataUtilitiesTest_v3.equalForFirstValueNull(org.jfree.data.DataUtilitiesTest_v3) (0 ms)
- org.jfree.data.DataUtilitiesTest_v3.calculateRowTotalForNullValue(org.jfree.data.DataUtilitiesTest_v3) (0 ms)
- org.jfree.data.DataUtilitiesTest_v3.calculateColumnTotalForTwoValues(org.jfree.data.DataUtilitiesTest_v3) (1 ms)
- org.jfree.data.testA2.DataUtilitiesTest.equalFor10By10ArrayTest(org.jfree.data.testA2.DataUtilitiesTest) (1 ms)

- org.jfree.data.DataUtilitiesTest_v3.calculateColumnTotalForOneValues(org.jfree.data.DataUtilitiesTest_v3) (1 ms)
- org.jfree.data.testA2.DataUtilitiesTest.cloneFor10By10ArrayTest(org.jfree.data.testA2.DataUtilitiesTest) (1 ms)
- org.jfree.data.DataUtilitiesTest_v3.equalForSecondValueNull(org.jfree.data.DataUtilitiesTest_v3) (0 ms)
- org.jfree.data.testA3.DataUtilitiesTest_v2.calculateRowTotalForNullValue (org.jfree.data.testA3.DataUtilitiesTest_v2) (2 ms)
- org.jfree.data.testA3.DataUtilitiesTest_v2.equalFor1By1ArrayTest(org.jfree.data.testA3.DataUtilitiesTest_v2) (1 ms)
- org.jfree.data.DataUtilitiesTest_v3.getCumulativePercentagesForZeroValues (org.jfree.data.DataUtilitiesTest_v3) (2 ms)
- org.jfree.data.DataUtilitiesTest_v3.calculateRowTotalForTwoValues(org.jfree.data.DataUtilitiesTest_v3) (1 ms)
- org.jfree.data.DataUtilitiesTest_v3.calculateRowTotalForNegativeValues(org.jfree.data.DataUtilitiesTest_v3) (1 ms)
- org.jfree.data.DataUtilitiesTest_v3.equalFor1By10ArrayTest(org.jfree.data.DataUtilitiesTest_v3) (0 ms)
- org.jfree.data.testA2.DataUtilitiesTest.createNumberArrayForNULL(org.jfree.data.testA2.DataUtilitiesTest) (0 ms)
- org.jfree.data.testA3.DataUtilitiesTest_v2.equalForBothValuesNull(org.jfree.data.testA3.DataUtilitiesTest_v2) (0 ms)
- org.jfree.data.testA3.DataUtilitiesTest_v2.createNumberArray2DFor10By1Array (org.jfree.data.testA3.DataUtilitiesTest_v2) (1 ms)
- org.jfree.data.testA3.DataUtilitiesTest_v2.getCumulativePercentagesForThreeValues (org.jfree.data.testA3.DataUtilitiesTest_v2) (2 ms)
- org.jfree.data.testA3.DataUtilitiesTest_v2.cloneFor1By1ArrayTest(org.jfree.data.testA3.DataUtilitiesTest_v2) (1 ms)
- org.jfree.data.DataUtilitiesTest_v3.createNumberArray2DForNULL(org.jfree.data.DataUtilitiesTest_v3) (2 ms)
- org.jfree.data.testA2.DataUtilitiesTest.cloneFor10By1ArrayTest(org.jfree.data.testA2.DataUtilitiesTest) (1 ms)
- org.jfree.data.testA3.DataUtilitiesTest_v2.calculateColumnTotalForTwoValues (org.jfree.data.testA3.DataUtilitiesTest_v2) (5 ms)
- org.jfree.data.DataUtilitiesTest_v3.createNumberArrayOfSize1(org.jfree.data.DataUtilitiesTest_v3) (0 ms)
- org.jfree.data.DataUtilitiesTest_v3.notEqualFor10By10ArrayTest(org.jfree.data.DataUtilitiesTest_v3) (0 ms)
- org.jfree.data.DataUtilitiesTest_v3.cloneFor10By1ArrayTest(org.jfree.data.DataUtilitiesTest_v3) (1 ms)
- org.jfree.data.testA2.DataUtilitiesTest.createNumberArray2DFor10By1Array (org.jfree.data.testA2.DataUtilitiesTest) (2 ms)
- org.jfree.data.DataUtilitiesTest_v3.createNumberArray2DFor1By1Array(org.jfree.data.DataUtilitiesTest_v3) (19 ms)
- org.jfree.data.DataUtilitiesTest_v3.equalFor1By1ArrayTest(org.jfree.data.DataUtilitiesTest_v3) (0 ms)
- org.jfree.data.DataUtilitiesTest_v3.calculateRowTotalForZeroValues(org.jfree.data.DataUtilitiesTest_v3) (1 ms)
- org.jfree.data.testA3.DataUtilitiesTest_v2.equalForInequalLengths(org.jfree.data.testA3.DataUtilitiesTest_v2) (1 ms)
- org.jfree.data.DataUtilitiesTest_v3.cloneFor1By1ArrayTest(org.jfree.data.DataUtilitiesTest_v3) (4 ms)
- org.jfree.data.DataUtilitiesTest_v3.createNumberArrayForNULL(org.jfree.data.DataUtilitiesTest_v3) (4 ms)
- org.jfree.data.testA3.DataUtilitiesTest_v2.calculateRowTotalForTwoValues (org.jfree.data.testA3.DataUtilitiesTest_v2) (1 ms)
- org.jfree.data.testA2.DataUtilitiesTest.calculateColumnTotalForTwoValues (org.jfree.data.testA2.DataUtilitiesTest) (1 ms)
- org.jfree.data.testA3.DataUtilitiesTest_v2.calculateRowTotalForZeroValues (org.jfree.data.testA3.DataUtilitiesTest_v2) (0 ms)
- org.jfree.data.DataUtilitiesTest_v3.cloneFor10By10ArrayTest(org.jfree.data.DataUtilitiesTest_v3) (4 ms)
- org.jfree.data.testA2.DataUtilitiesTest.getCumulativePercentagesForZeroValues (org.jfree.data.testA2.DataUtilitiesTest) (1 ms)
- org.jfree.data.testA2.DataUtilitiesTest.cloneFor1By1ArrayTest(org.jfree.data.testA2.DataUtilitiesTest) (1 ms)
- org.jfree.data.testA3.DataUtilitiesTest_v2.calculateColumnTotalForOneValues (org.jfree.data.testA3.DataUtilitiesTest_v2) (1 ms)
- org.jfree.data.DataUtilitiesTest_v3.createNumberArray2DFor10By1Array(org.jfree.data.DataUtilitiesTest_v3) (0 ms)
- org.jfree.data.testA3.DataUtilitiesTest_v2.cloneFor10By1ArrayTest(org.jfree.data.testA3.DataUtilitiesTest_v2) (0 ms)
- org.jfree.data.testA2.DataUtilitiesTest.createNumberArrayOfSize10(org.jfree.data.testA2.DataUtilitiesTest) (2 ms)
- org.jfree.data.testA3.DataUtilitiesTest_v2.createNumberArrayOfSize10 (org.jfree.data.testA3.DataUtilitiesTest_v2) (0 ms)
- org.jfree.data.testA2.DataUtilitiesTest.equalFor10By1ArrayTest(org.jfree.data.testA2.DataUtilitiesTest) (5 ms)
- org.jfree.data.testA3.DataUtilitiesTest_v2.createNumberArrayForNULL (org.jfree.data.testA3.DataUtilitiesTest_v2) (1 ms)

- org.jfree.data.DataUtilitiesTest_v3.calculateColumnTotalForNullValues(org.jfree.data.DataUtilitiesTest_v3) (2 ms)
- org.jfree.data.testA3.DataUtilitiesTest_v2.cloneFor10By10ArrayTest (org.jfree.data.testA3.DataUtilitiesTest_v2) (1 ms)
- org.jfree.data.testA2.DataUtilitiesTest.calculateColumnTotalForNegativeValues (org.jfree.data.testA2.DataUtilitiesTest) (1 ms)
- org.jfree.data.testA3.DataUtilitiesTest_v2.createNumberArray2DFor1By10Array (org.jfree.data.testA3.DataUtilitiesTest_v2) (0 ms)
- org.jfree.data.DataUtilitiesTest_v3.calculateRowTotalValidColumnsForNullValue (org.jfree.data.DataUtilitiesTest_v3) (1 ms)
- org.jfree.data.testA3.DataUtilitiesTest_v2.equalForFirstValueNull(org.jfree.data.testA3.DataUtilitiesTest_v2) (3 ms)
- org.jfree.data.DataUtilitiesTest_v3.equalForBothValuesNull(org.jfree.data.DataUtilitiesTest_v3) (1 ms)
- org.jfree.data.DataUtilitiesTest_v3.createNumberArrayOfSize10(org.jfree.data.DataUtilitiesTest_v3) (1 ms)
- org.jfree.data.testA3.DataUtilitiesTest_v2.notCloneFor10By10ArrayTest (org.jfree.data.testA3.DataUtilitiesTest_v2) (3 ms)
- org.jfree.data.testA2.DataUtilitiesTest.createNumberArray2DForNULL(org.jfree.data.testA2.DataUtilitiesTest) (1 ms)
- org.jfree.data.testA2.DataUtilitiesTest.createNumberArray2DFor1By1Array (org.jfree.data.testA2.DataUtilitiesTest) (6 ms)
- org.jfree.data.testA3.DataUtilitiesTest_v2.calculateRowTotalForNegativeValues (org.jfree.data.testA3.DataUtilitiesTest_v2) (3 ms)
- org.jfree.data.testA2.DataUtilitiesTest.equalFor1By1ArrayTest(org.jfree.data.testA2.DataUtilitiesTest) (0 ms)
- org.jfree.data.testA3.DataUtilitiesTest_v2.calculateRowTotalValidColumnsForNullValue (org.jfree.data.testA3.DataUtilitiesTest_v2) (1 ms)
- org.jfree.data.testA3.DataUtilitiesTest_v2.calculateColumnTotalValidRowsForNullValue (org.jfree.data.testA3.DataUtilitiesTest_v2) (4 ms)
- org.jfree.data.DataUtilitiesTest_v3.createNumberArray2DFor10By10Array(org.jfree.data.DataUtilitiesTest_v3) (0 ms)
- org.jfree.data.DataUtilitiesTest_v3.cloneFor1By10ArrayTest(org.jfree.data.DataUtilitiesTest_v3) (1 ms)
- org.jfree.data.testA3.DataUtilitiesTest_v2.calculateColumnTotalForNegativeValues (org.jfree.data.testA3.DataUtilitiesTest_v2) (1 ms)
- org.jfree.data.testA3.DataUtilitiesTest_v2.equalFor10By1ArrayTest(org.jfree.data.testA3.DataUtilitiesTest_v2) (1 ms)
- org.jfree.data.DataUtilitiesTest_v3.calculateColumnTotalValidRowsForNullValue (org.jfree.data.DataUtilitiesTest_v3) (4 ms)
- org.jfree.data.DataUtilitiesTest_v3.createNumberArray2DFor1By10Array(org.jfree.data.DataUtilitiesTest_v3) (0 ms)
- org.jfree.data.testA3.DataUtilitiesTest_v2.equalForSecondValueNull (org.jfree.data.testA3.DataUtilitiesTest_v2) (1 ms)
- org.jfree.data.testA2.DataUtilitiesTest.createNumberArrayOfSize1(org.jfree.data.testA2.DataUtilitiesTest) (1 ms)
- org.jfree.data.testA2.DataUtilitiesTest.cloneFor1By10ArrayTest(org.jfree.data.testA2.DataUtilitiesTest) (0 ms)
- org.jfree.data.DataUtilitiesTest_v3.equalFor10By10ArrayTest(org.jfree.data.DataUtilitiesTest_v3) (1 ms)
- org.jfree.data.testA3.DataUtilitiesTest_v2.calculateColumnTotalForNullValues (org.jfree.data.testA3.DataUtilitiesTest_v2) (1 ms)
- org.jfree.data.DataUtilitiesTest_v3.equalFor10By1ArrayTest(org.jfree.data.DataUtilitiesTest_v3) (1 ms)
- org.jfree.data.DataUtilitiesTest_v3.getCumulativePercentagesForThreeValues (org.jfree.data.DataUtilitiesTest_v3) (5 ms)
- org.jfree.data.testA3.DataUtilitiesTest_v2.createNumberArray2DFor10By10Array (org.jfree.data.testA3.DataUtilitiesTest_v2) (1 ms)
- org.jfree.data.DataUtilitiesTest_v3.cloneForNullRowArrayTest(org.jfree.data.DataUtilitiesTest_v3) (1 ms)
- org.jfree.data.testA3.DataUtilitiesTest_v2.createNumberArray2DForNULL (org.jfree.data.testA3.DataUtilitiesTest_v2) (0 ms)
- org.jfree.data.testA3.DataUtilitiesTest_v2.cloneFor1By10ArrayTest(org.jfree.data.testA3.DataUtilitiesTest_v2) (0 ms)
- org.jfree.data.DataUtilitiesTest_v3.notCloneFor10By10ArrayTest(org.jfree.data.DataUtilitiesTest_v3) (0 ms)
- org.jfree.data.testA2.DataUtilitiesTest.createNumberArray2DFor1By10Array (org.jfree.data.testA2.DataUtilitiesTest) (1 ms)
- org.jfree.data.testA2.DataUtilitiesTest.notEqualFor10By10ArrayTest(org.jfree.data.testA2.DataUtilitiesTest) (1 ms)

Report generated by PIT 1.4.11