MutableIntTest.java - MutableLongTest.java

```java
/*
 * Copyright 2002-2005 The Apache Software Foundation.
 *
 * Licensed under the Apache License, Version 2.0 (the "License");
 * you may not use this file except in compliance with the License.
 * You may obtain a copy of the License at
 *
 *      http://www.apache.org/licenses/LICENSE-2.0
 *
 * Unless required by applicable law or agreed to in writing, software
 * distributed under the License is distributed on an "AS IS" BASIS,
 * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
 * See the License for the specific language governing permissions and
 * limitations under the License.
 */
package org.apache.commons.lang.mutable;

import junit.framework.Test;
import junit.framework.TestCase;
import junit.framework.TestSuite;
import junit.textui.TestRunner;

/**
 * JUnit tests.
 *
 * @version $Id: MutableIntTest.java 161244 2005-04-14 06:16:36Z ggregory $
 * @see MutableInt
 */
public class MutableIntTest extends TestCase {

    public MutableIntTest(String testName) {
        super(testName);
    }

    public static void main(String[] args) {
        TestRunner.run(suite());
    }

    public static Test suite() {
        return new TestSuite(MutableIntTest.class);
    }

    // ----------------------------------------------------------------
    public void testConstructors() {
        assertEquals(0, new MutableInt().intValue());

        assertEquals(1, new MutableInt(1).intValue());

        assertEquals(2, new MutableInt(new Integer(2)).intValue());
        assertEquals(3, new MutableInt(new MutableLong(3)).intValue());
        try {
            new MutableInt(null);
            fail();
        } catch (NullPointerException ex) {}
    }

```

```java
/*
 * Copyright 2002-2005 The Apache Software Foundation.
 *
 * Licensed under the Apache License, Version 2.0 (the "License");
 * you may not use this file except in compliance with the License.
 * You may obtain a copy of the License at
 *
 *      http://www.apache.org/licenses/LICENSE-2.0
 *
 * Unless required by applicable law or agreed to in writing, software
 * distributed under the License is distributed on an "AS IS" BASIS,
 * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
 * See the License for the specific language governing permissions and
 * limitations under the License.
 */
package org.apache.commons.lang.mutable;

import junit.framework.Test;
import junit.framework.TestCase;
import junit.framework.TestSuite;
import junit.textui.TestRunner;

/**
 * JUnit tests.
 *
 * @version $Id: MutableLongTest.java 161244 2005-04-14 06:16:36Z ggregory $
 * @see MutableLong
 */
public class MutableLongTest extends TestCase {

    public MutableLongTest(String testName) {
        super(testName);
    }

    public static void main(String[] args) {
        TestRunner.run(suite());
    }

    public static Test suite() {
        return new TestSuite(MutableLongTest.class);
    }

    // ----------------------------------------------------------------
    public void testConstructors() {
        assertEquals(0, new MutableLong().longValue());

        assertEquals(1, new MutableLong(1).longValue());

        assertEquals(2, new MutableLong(new Long(2)).longValue());
        assertEquals(3, new MutableLong(new MutableLong(3)).longValue());
        try {
            new MutableLong(null);
            fail();
        } catch (NullPointerException ex) {}
    }

```

MutableIntTest.java - MutableLongTest.java

```
57    public void testGetSet() {
58        final MutableInt mutNum = new MutableInt(0);
59        assertEquals(0, new MutableInt().intValue());
60        assertEquals(new Integer(0), new MutableInt().getValue());
61
62        mutNum.setValue(1);
63        assertEquals(1, mutNum.intValue());
64        assertEquals(new Integer(1), mutNum.getValue());
65
66        mutNum.setValue(new Integer(2));
67        assertEquals(2, mutNum.intValue());
68        assertEquals(new Integer(2), mutNum.getValue());
69
70        mutNum.setValue(new MutableLong(3));
71        assertEquals(3, mutNum.intValue());
72        assertEquals(new Integer(3), mutNum.getValue());
73        try {
74            mutNum.setValue(null);
75            fail();
76        } catch (NullPointerException ex) {}
77        try {
78            mutNum.setValue("0");
79            fail();
80        } catch (ClassCastException ex) {}
81    }
82
83    public void testEquals() {
84        final MutableInt mutNumA = new MutableInt(0);
85        final MutableInt mutNumB = new MutableInt(0);
86        final MutableInt mutNumC = new MutableInt(1);
87
88        assertEquals(true, mutNumA.equals(mutNumA));
89        assertEquals(true, mutNumA.equals(mutNumB));
90        assertEquals(true, mutNumB.equals(mutNumA));
91        assertEquals(true, mutNumB.equals(mutNumB));
92        assertEquals(false, mutNumA.equals(mutNumC));
93        assertEquals(false, mutNumB.equals(mutNumC));
94        assertEquals(true, mutNumC.equals(mutNumC));
95        assertEquals(false, mutNumA.equals(null));
96        assertEquals(false, mutNumA.equals(new Integer(0)));
97        assertEquals(false, mutNumA.equals("0"));
98    }
99
100   public void testHashCode() {
101       final MutableInt mutNumA = new MutableInt(0);
102       final MutableInt mutNumB = new MutableInt(0);
103       final MutableInt mutNumC = new MutableInt(1);
104
105       assertEquals(true, mutNumA.hashCode() == mutNumA.hashCode());
106       assertEquals(true, mutNumA.hashCode() == mutNumB.hashCode());
107       assertEquals(false, mutNumA.hashCode() == mutNumC.hashCode());
108       assertEquals(true, mutNumA.hashCode() == new Integer(0).hashCode());
109   }
110
111   public void testCompareTo() {
112       final MutableInt mutNum = new MutableInt(0);
```

```
57    public void testGetSet() {
58        final MutableLong mutNum = new MutableLong(0);
59        assertEquals(0, new MutableLong().longValue());
60        assertEquals(new Long(0), new MutableLong().getValue());
61
62        mutNum.setValue(1);
63        assertEquals(1, mutNum.longValue());
64        assertEquals(new Long(1), mutNum.getValue());
65
66        mutNum.setValue(new Long(2));
67        assertEquals(2, mutNum.longValue());
68        assertEquals(new Long(2), mutNum.getValue());
69
70        mutNum.setValue(new MutableLong(3));
71        assertEquals(3, mutNum.longValue());
72        assertEquals(new Long(3), mutNum.getValue());
73        try {
74            mutNum.setValue(null);
75            fail();
76        } catch (NullPointerException ex) {}
77        try {
78            mutNum.setValue("0");
79            fail();
80        } catch (ClassCastException ex) {}
81    }
82
83    public void testEquals() {
84        final MutableLong mutNumA = new MutableLong(0);
85        final MutableLong mutNumB = new MutableLong(0);
86        final MutableLong mutNumC = new MutableLong(1);
87
88        assertEquals(true, mutNumA.equals(mutNumA));
89        assertEquals(true, mutNumA.equals(mutNumB));
90        assertEquals(true, mutNumB.equals(mutNumA));
91        assertEquals(true, mutNumB.equals(mutNumB));
92        assertEquals(false, mutNumA.equals(mutNumC));
93        assertEquals(false, mutNumB.equals(mutNumC));
94        assertEquals(true, mutNumC.equals(mutNumC));
95        assertEquals(false, mutNumA.equals(null));
96        assertEquals(false, mutNumA.equals(new Long(0)));
97        assertEquals(false, mutNumA.equals("0"));
98    }
99
100   public void testHashCode() {
101       final MutableLong mutNumA = new MutableLong(0);
102       final MutableLong mutNumB = new MutableLong(0);
103       final MutableLong mutNumC = new MutableLong(1);
104
105       assertEquals(true, mutNumA.hashCode() == mutNumA.hashCode());
106       assertEquals(true, mutNumA.hashCode() == mutNumB.hashCode());
107       assertEquals(false, mutNumA.hashCode() == mutNumC.hashCode());
108       assertEquals(true, mutNumA.hashCode() == new Long(0).hashCode());
109   }
110
111   public void testCompareTo() {
112       final MutableLong mutNum = new MutableLong(0);
```

MutableIntTest.java - MutableLongTest.java

```
113
114         assertEquals(0, mutNum.compareTo(new MutableInt(0)));
115         assertEquals(+1, mutNum.compareTo(new MutableInt(-1)));
116         assertEquals(-1, mutNum.compareTo(new MutableInt(1)));
117         try {
118             mutNum.compareTo(null);
119             fail();
120         } catch (NullPointerException ex) {}
121         try {
122             mutNum.compareTo(new Integer(0));
123             fail();
124         } catch (ClassCastException ex) {}
125         try {
126             mutNum.compareTo("0");
127             fail();
128         } catch (ClassCastException ex) {}
129     }
130
131     public void testPrimitiveValues() {
132         MutableInt mutNum = new MutableInt(1);
133
134         assertEquals( (byte) 1, mutNum.byteValue() );
135         assertEquals( (short) 1, mutNum.shortValue() );
136         assertEquals( 1.0F, mutNum.floatValue(), 0 );
137         assertEquals( 1.0, mutNum.doubleValue(), 0 );
138         assertEquals( 1L, mutNum.longValue() );
139     }
140
141     public void testToString() {
142         assertEquals("0", new MutableInt(0).toString());
143         assertEquals("10", new MutableInt(10).toString());
144         assertEquals("-123", new MutableInt(-123).toString());
145     }
146
147}
148
```

```
113
114         assertEquals(0, mutNum.compareTo(new MutableLong(0)));
115         assertEquals(+1, mutNum.compareTo(new MutableLong(-1)));
116         assertEquals(-1, mutNum.compareTo(new MutableLong(1)));
117         try {
118             mutNum.compareTo(null);
119             fail();
120         } catch (NullPointerException ex) {}
121         try {
122             mutNum.compareTo(new Long(0));
123             fail();
124         } catch (ClassCastException ex) {}
125         try {
126             mutNum.compareTo("0");
127             fail();
128         } catch (ClassCastException ex) {}
129     }
130
131     public void testPrimitiveValues() {
132         MutableLong mutNum = new MutableLong(1L);
133
134         assertEquals( 1.0F, mutNum.floatValue(), 0 );
135         assertEquals( 1.0, mutNum.doubleValue(), 0 );
136         assertEquals( (byte) 1, mutNum.byteValue() );
137         assertEquals( (short) 1, mutNum.shortValue() );
138         assertEquals( 1, mutNum.intValue() );
139         assertEquals( 1L, mutNum.longValue() );
140     }
141
142     public void testToString() {
143         assertEquals("0", new MutableLong(0).toString());
144         assertEquals("10", new MutableLong(10).toString());
145         assertEquals("-123", new MutableLong(-123).toString());
146     }
147
148}
149
```