TestWhileDoProcedure.java - TestDoWhileProcedure.java

```java
/*
 * Licensed to the Apache Software Foundation (ASF) under one or more
 * contributor license agreements.  See the NOTICE file distributed with
 * this work for additional information regarding copyright ownership.
 * The ASF licenses this file to You 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.functor.core.composite;

import junit.framework.Test;
import junit.framework.TestSuite;

import org.apache.commons.functor.BaseFunctorTest;
import org.apache.commons.functor.Predicate;
import org.apache.commons.functor.Procedure;
import org.apache.commons.functor.adapter.BoundPredicate;
import org.apache.commons.functor.core.Constant;
import org.apache.commons.functor.core.NoOp;
import org.apache.commons.functor.core.collection.IsEmpty;

import java.util.LinkedList;
import java.util.List;

/**
 * @version $Revision: 666345 $ $Date: 2008-06-11 07:13:44 +0900 (水, 11  6月 2008) $
 * @author Herve Quiroz
 */
public class TestWhileDoProcedure extends BaseFunctorTest {

    // Conventional
    // -------------------------------------------------------------------

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

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

    // Functor Testing Framework
    // -------------------------------------------------------------------

    protected Object makeFunctor() {
        return new WhileDoProcedure(Constant.FALSE, NoOp.INSTANCE);
    }
```

```java
/*
 * Licensed to the Apache Software Foundation (ASF) under one or more
 * contributor license agreements.  See the NOTICE file distributed with
 * this work for additional information regarding copyright ownership.
 * The ASF licenses this file to You 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.functor.core.composite;

import junit.framework.Test;
import junit.framework.TestSuite;

import org.apache.commons.functor.BaseFunctorTest;
import org.apache.commons.functor.Predicate;
import org.apache.commons.functor.Procedure;
import org.apache.commons.functor.adapter.BoundPredicate;
import org.apache.commons.functor.core.Constant;
import org.apache.commons.functor.core.NoOp;
import org.apache.commons.functor.core.collection.IsEmpty;

import java.util.LinkedList;
import java.util.List;

/**
 * @version $Revision: 666345 $ $Date: 2008-06-11 07:13:44 +0900 (水, 11  6月 2008) $
 * @author Herve Quiroz
 */
public class TestDoWhileProcedure extends BaseFunctorTest {

    // Conventional
    // -------------------------------------------------------------------

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

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

    // Functor Testing Framework
    // -------------------------------------------------------------------

    protected Object makeFunctor() {
        return new DoWhileProcedure(NoOp.INSTANCE, Constant.FALSE);
    }
```

TestWhileDoProcedure.java - TestDoWhileProcedure.java

```
56
57      // Lifecycle
58      // ----------------------------------------------------------------------
59
60      public void setUp() throws Exception {
61          super.setUp();
62      }
63
64      public void tearDown() throws Exception {
65          super.tearDown();
66      }
67
68      // Tests
69      // ----------------------------------------------------------------------
70      public class ListRemoveFirstProcedure implements Procedure {
71          protected List<Object> list;
72
73
74          public ListRemoveFirstProcedure(List<Object> list) {
75              this.list=list;
76          }
77
78
79          public void run() {
80              list.remove(0);
81          }
82      }
83
84
85      private List<Object> getList() {
86          List<Object> list=new LinkedList<Object>();
87          list.add("a");
88          list.add("b");
89          list.add("c");
90          list.add("d");
91          return list;
92      }
93
94
95      public void testLoopWithAction() throws Exception {
96          List<Object> list=getList();
97
98          Procedure action=new ListRemoveFirstProcedure(list);
99          Predicate condition=new Not(new BoundPredicate(new
  IsEmpty<List<Object>>(), list));
100         Procedure procedure=new WhileDoProcedure(condition, action);
101
102         assertTrue("The condition should be true before running the loop",
  condition.test());
103         assertFalse("The list should not be empty then", list.isEmpty());
104         procedure.run();
105         assertFalse("The condition should be false after running the loop",
  condition.test());
106         assertTrue("The list should be empty then", list.isEmpty());
107
108         list=getList();
```

```
56
57      // Lifecycle
58      // ----------------------------------------------------------------------
59
60      public void setUp() throws Exception {
61          super.setUp();
62      }
63
64      public void tearDown() throws Exception {
65          super.tearDown();
66      }
67
68      // Tests
69      // ----------------------------------------------------------------------
70      public class ListRemoveFirstProcedure implements Procedure {
71          protected List<Object> list;
72
73
74          public ListRemoveFirstProcedure(List<Object> list) {
75              this.list=list;
76          }
77
78
79          public void run() {
80              list.remove(0);
81          }
82      }
83
84
85      private List<Object> getList() {
86          List<Object> list = new LinkedList<Object>();
87          list.add("a");
88          list.add("b");
89          list.add("c");
90          list.add("d");
91          return list;
92      }
93
94
95      public void testLoopWithAction() throws Exception {
96          List<Object> list=getList();
97
98          Procedure action=new ListRemoveFirstProcedure(list);
99          Predicate condition=new Not(new BoundPredicate(new
  IsEmpty<List<Object>>(), list));
100         Procedure procedure=new DoWhileProcedure(action, condition);
101
102         assertTrue("The condition should be true before running the loop",
  condition.test());
103         assertFalse("The list should not be empty then", list.isEmpty());
104         procedure.run();
105         assertFalse("The condition should be false after running the loop",
  condition.test());
106         assertTrue("The list should be empty then", list.isEmpty());
107
108         list=getList();
```

TestWhileDoProcedure.java - TestDoWhileProcedure.java

```
109         action=new ListRemoveFirstProcedure(list);
110         condition=new Predicate() {
111                         private int count=2;
112
113                         public boolean test() {
114                             return count-- > 0;
115                         }
116                     };
117         procedure=new WhileDoProcedure(condition, action);
118         procedure.run();
119         assertFalse("The list should not contain ¥"a¥" anymore",
    list.contains("a"));
120         assertFalse("The list should not contain ¥"b¥" anymore",
    list.contains("b"));
121         assertTrue("The list should still contain ¥"c¥"", list.contains("c"));
122         assertTrue("The list should still contain ¥"d¥"", list.contains("d"));
123     }
124
125     public void testLoopForNothing() {
126         List<Object> list=getList();
127         Procedure action=new ListRemoveFirstProcedure(list);
128         Procedure procedure=new WhileDoProcedure(Constant.FALSE, action);
129         assertTrue("The list should contain 4 elements before runnning the loop",
    list.size()==4);
130         procedure.run();
131         assertTrue("The list should contain 4 elements after runnning the loop",
    list.size()==4);
132     }
133}
134
135
```

```
109         action=new ListRemoveFirstProcedure(list);
110         condition=new Predicate() {
111                         private int count=2;
112
113                         public boolean test() {
114                             return count-- > 0;
115                         }
116                     };
117         procedure=new DoWhileProcedure(action, condition);
118         procedure.run();
119         assertFalse("The list should not contain ¥"a¥" anymore",
    list.contains("a"));
120         assertFalse("The list should not contain ¥"b¥" anymore",
    list.contains("b"));
121         assertFalse("The list should not contain ¥"c¥" anymore",
    list.contains("c"));
122         assertTrue("The list should still contain ¥"d¥"", list.contains("d"));
123     }
124
125     public void testLoopForNothing() {
126         List<Object> list=getList();
127         Procedure action=new ListRemoveFirstProcedure(list);
128         Procedure procedure=new DoWhileProcedure(action, Constant.FALSE);
129         assertTrue("The list should contain 4 elements before runnning the loop",
    list.size()==4);
130         procedure.run();
131         assertTrue("The list should contain 3 elements after runnning the loop",
    list.size()==3);
132     }
133}
134
```