70094 ExerciseTypes.CW3

Test-Driven Development

Submitters

sf23	Shihan Fu
nl1023	Ningqi Luo

Emarking

```
Final Tests: Summary for sf23 of v5
-----------------------------------

  Public Tests:
    Compiles:                  1 / 1
    Tests Pass:                1 / 1
    Coverage and Style Checks: 1 / 1

Git Repo: git@gitlab.doc.ic.ac.uk:lab2324_spring/SE_Design_Ex3_sf23.git
Commit ID: 1e7c9
```

+

16/20

Generally a good implementation.

Pay attention to the test names: they should describe the expected behaviour, not what the test does.
Use generics to support multiple types, not Objects.
Your API is a bit cumbersome.
Try to reduce code duplication, even in tests.

See comments below.

```
1: package ic.doc;
2:
3: import static org.hamcrest.MatcherAssert.assertThat;
4: import static org.hamcrest.core.Is.is;
5: import static org.junit.Assert.assertEquals;
6: import static org.junit.Assert.assertTrue;
7: import static org.junit.Assert.assertFalse;
8:
9: import org.junit.Test;
10:
11: public class RecentlyUsedListTest {
12:
13:   @Test
14:   public void canAnswerTheUniversalQuestion() {
15:     assertThat(new RecentlyUsedList().answer(), is(42));
16:   }
17:
18:   @Test
19:   public void getSize() {
20:     assertThat(new RecentlyUsedList().getSize(), is(0));
21:   }
22:
23:   @Test
24:   public void addElement() {
25:     assertTrue(new RecentlyUsedList().addElement("Google"));
26:   }
27:
28:   @Test
29:   public void getRecentlyUsed() {
30:     RecentlyUsedList r = new RecentlyUsedList();
31:     r.addElement("Hello");
32:     r.addElement("Google");
33:     assertEquals(r.getRecentlyUsed(), "Google");
34:   }
35:
36:   @Test
37:   public void testOtherTypes() {
38:     RecentlyUsedList r = new RecentlyUsedList();
39:     r.addElement("Hello");
40:     r.addElement(1);
41:     assertTrue(r.retrieveElement(1));
42:   }
43:
44:   @Test
45:   public void retrieveElement() {
46:     RecentlyUsedList r = new RecentlyUsedList();
47:     r.addElement("Hello");
48:     r.addElement("Google");
49:     assertTrue(r.retrieveElement("Google"));
50:     assertFalse(r.retrieveElement("Hi"));
51:   }
52:
53:   @Test
54:   public void checkUnique() {
55:     RecentlyUsedList r = new RecentlyUsedList();
56:     r.addElement("Hello");
57:     r.addElement("Google");
58:     r.addElement("Hello");
59:
60:     String str = r.toString();
61:     String findStr = "Hello";
62:     int lastIndex = 0;
63:     int count = 0;
64:     while (lastIndex != -1) {
65:       lastIndex = str.indexOf(findStr, lastIndex);
66:       if (lastIndex != -1) {
```

You're creating a new list in each test: reduce code duplication and instantiate it one the class level or in a set-up method.

You should describe the expected behaviour, not what the test does.

You are storing them as Objects, so it's clear it works.


```
67:         count++;
68:         lastIndex += findStr.length();
69:       }
70:     }
71:     assertEquals(count, 1);
72:   }
73:
74:
75: }
```

```java
package ic.doc;

import java.util.LinkedList;

public class RecentlyUsedList {
  private LinkedList<Object> sites = new LinkedList<>();

  /**
   * add element to the list.
   *
   * @param s string
   * @return true when success, false when fail
   */
  public boolean addElement(Object s) {
    // check whether it exists
    if (!sites.contains(s)) {
      // add directly to the beginning of the list
      return sites.offerFirst(s);
    } else {
      if (sites.remove(s)) {
        return sites.offerFirst(s);
      }
      return false;
    }
  }

  /**
   * retrieve element.
   *
   * @param s string
   * @return true when found, false when not find
   */
  public boolean retrieveElement(Object s) {
    return sites.contains(s);
  }

  /**
   * list elements to string.
   *
   * @return string
   */
  public String toString() {
    String ret = "";
    for (Object site : sites) {
      ret = ret + site.toString() + " ";
    }
    return ret;
  }

  /**
   * get size of the RUL.
   *
   * @return the size of the list
   */
  public int getSize() {
    return sites.size();
  }

  /**
   * get the most recently used element ( the first one ).
   *
   * @return o the recently used object
   */
  public Object getRecentlyUsed() {
    if (getSize() == 0) {
      System.out.println("empty sites!");
```

Comments aren't required in this course.

You can just do .remove() and store its result.

This comment is useless since it's already clear from the method name.

Do you need this method?


```java
      return null;
    }
    return sites.get(0);
  }

  public int answer() {
    return 42;
  }
}
```

Why?

Better make a .get() method that retrieves by index or add an .toArray() method that is similar to List.toArray()

```
1: -------- Test Output --------
2: Running LabTS build... (Thu  1 Feb 13:35:32 UTC 2024)
3:
4: Submission summary...
5: You made 3 commits
6:   - a6e5f4b implement the RUL class and tests [4 files changed, 142 insertions, 24 deletions]
7:   - 2a058a6 add test for different types [2 files changed, 11 insertions, 3 deletions]
8:   - 1e7c910 add access modifiers when declaring variable [1 file changed, 1 insertion, 1 deletion]
9:
10: Preparing...
11:
12: BUILD SUCCESSFUL in 1s
13:
14: Compiling...
15: BUILD SUCCESSFUL in 3s
16:
17: Running tests...
18:
19: ic.doc.RecentlyUsedListTest > canAnswerTheUniversalQuestion PASSED
20:
21: ic.doc.RecentlyUsedListTest > testOtherTypes PASSED
22:
23: ic.doc.RecentlyUsedListTest > getRecentlyUsed PASSED
24:
25: ic.doc.RecentlyUsedListTest > getSize PASSED
26:
27: ic.doc.RecentlyUsedListTest > addElement PASSED
28:
29: ic.doc.RecentlyUsedListTest > checkUnique PASSED
30:
31: ic.doc.RecentlyUsedListTest > retrieveElement PASSED
32:
33: BUILD SUCCESSFUL in 1s
34:
35: Checking test coverage and code style...
36: BUILD SUCCESSFUL in 6s
37: Finished auto test. (Thu  1 Feb 13:35:54 UTC 2024)
38:
39: -------- Test Errors --------
40:
```

It doesn't seem like you've used TDD. Commit more often.

Prefer imperative mood.

See https://www.kernel.org/doc/html/v6.7/process/submitting-patches.html#describe-your-changes