@Test

public void testRemoveDuplicates()

{

words.add("cookie");

words.add("cake");

words.add("cake");

words.add("pie");

words.removeDuplicates();

assertTrue("removeDuplicates method", words.getFirst().equals("cookie"));

}

* The assertion only checks part of the final state (the first string “cookie” only), and it doesn’t reach the string “cake” that has the duplicates, so if the fault propagates to another part of the final state, the failure will not be revealed.
* More details:

Final state (expected): [“cookie”, “cake”, “pie”]

Final state (observed): [“cookie”]

* So if the program returns [“cookie”, “cake”, “cake”, “pie”] instead of the expected output, the test oracle would not reveal the problem.

We can rewrite the test method using **getLast()** instead of **getFirst()**, and change the order of the strings in the object as below:

@Test

public void testRemoveDuplicates()

{

words.add("cookie");

words.add("cake");

words.add("pie");

words.add("cake");

words.removeDuplicates();

assertTrue("removeDuplicates method", words.getLast().equals("pie"));

}

Or we can also rewrite the test method using **size()** instead of **getFirst()**, as below:

@Test

public void testRemoveDuplicates()

{

words.add("cookie");

words.add("cake");

words.add("cake");

words.add("pie");

words.removeDuplicates();

assertTrue("removeDuplicates method", words.size() == 3);

}