

EECS2030 Advanced Object-Oriented Programming
(Fall 2021)

Q&A - Review Tutorial Part 2

Wednesday, September 22

Announcement

2pm EST.

- LabOP2 (due: Sep. 24)
- Lecture W3 (released: Sep. 20)
- Lab1 (released: Sep. 22) 9 days -
- Written Test (due: Sep. 30 - Oct. 1)

Hello professor! In the tutorial video you said

assertTrue(e.getSerialNumber() == "F9DN4NKQ1GC");

is not necessarily working

but then you also said assertTrue(e.getProduct() == p) works.

Can you please explain why the first one is wrong and the second one is right.

Thank you!

most of time, when comparing string values, you mean to compare their contents
avoid! : using == to compare string values (equals) may fail!



```
@Test
public void test_entry_1() {
    Product p = new Product("iPad Pro 12.9", 1700.00);
    p.setFinish("Space Grey");
    p.setStorage(1000); // 1TB
    p.setHasCellularConnectivity(true);
    p.setDiscountValue(220.00);
    Entry e = new Entry("F9DN4NKQ1GC", p);
    // assertTrue(e.getSerialNumber() == "F9DN4NKQ1GC"); // not necessarily working a ways why?
    assertEquals("F9DN4NKQ1GC", e.getSerialNumber());
    assertTrue(e.getProduct() == p);
}
```



$e.product = product$

PRINCIPLE:

For space efficiency

reason, all occurrences of the same string literal will reference the same obj at runtime.

are they pointing to the same obj? new string ("F9DN4NKQ1GC")



"York University"

S.
Y.

New York

S.
U.

S.
N.

Why attributes should be private?

① Attributes are mainly for implementation.

↳ private array
ArrayList []
↳ LinkedList

⇒ ① attributes should be hidden as private
② let methods public

return values of those private attributes

② For long-lived projects, imp. strategies are subject to changes constantly.

⇒ when there's a change on a public attribute all callers of this att. will be affected
compilation errors

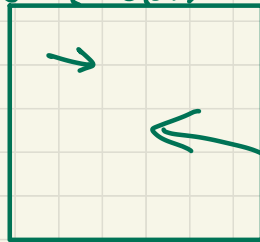
- Some methods are private for internal use only.

- How can a JUnit test method test it?

① You cannot call upon that private method from the test method.

② You can create some "public accessor" for that method for testing purposes.

JUnit Class



Model class

```
private void m() { ... }  
/* Testing */  
public void accessor() { m(); }
```

String Literals vs. String Objects

```

public class StringValues {
    public static void main(String[] args) {
        String s1 = "York University";
        String s2 = "York University";
        String s3 = new String("York University");

        ① System.out.println("s1 == s2: " + (s1 == s2));
        ② System.out.println("s1 == s3: " + (s1 == s3));
        ③ System.out.println("s2 == s3: " + (s2 == s3));

        ④ System.out.println("s1.equals(s2): " + s1.equals(s2));
        ⑤ System.out.println("s1.equals(s3): " + s1.equals(s3));
        ⑥ System.out.println("s2.equals(s3): " + s2.equals(s3));

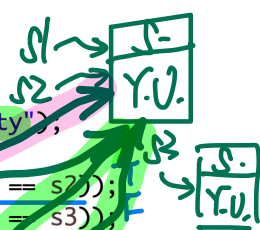
        Foo obj = new Foo();

        System.out.println("Set to a string literal value...");
        obj.fm("York University");

        T ⑦ System.out.println(obj.getS() == "York University");
        T ⑧ System.out.println(obj.getS().equals("York University"));

        String s4 = new String("York University");
        System.out.println("Set to a new string value...");
        obj.fm(s4);

        F ⑨ System.out.println(obj.getS() == "York University");
        T ⑩ System.out.println(obj.getS().equals("York University"));
    }
}
    
```

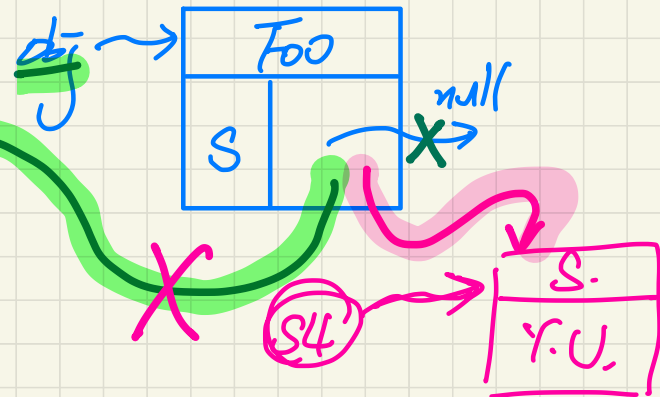


```

public class Foo {
    private String s;

    public void fm(String s) {
        this.s = s;
    }

    public String getS() {
        return s;
    }
}
    
```



I don't understand why the entries need to be this.no - 1.

Is it because noe doesn't use 0 as the first element index?

So if noe is 4, then index for the most recent entry should be 3, considering 0, 1, 2 and 3 as the entries?

