Wednesday Sep. 19

Lecture 5

Lab Test 1 : Oct. 1

Slides:

- Classes & Objects
- Exceptions
- JUnit
- Monday Sep. 24

Programming:

- Lab 1 (2D arrays, nested loops)
- Practice Problem

Testing from Console (V1): Test 1

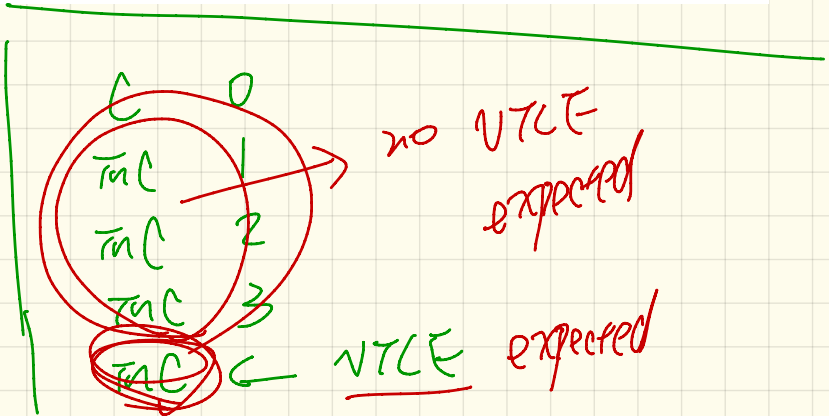
```
public class CounterTester1 {
    public static void main(String[] args) {
        Counter c = new Counter();
        System.out.println("Init val: " + c.getValue());
        try {
            c.decrement();
            System.out.println("ValueTooSmallException NOT thrown as expected.");
        }catch (ValueTooSmallException e) {
            System.out.println("ValueTooSmallException thrown as expected.");
        }
    }
}
```

VTSE

+ c.getVal()

l.v. for catch block

VTSE did not occur



Testing from Console (V1): Test 2

assume: correct imp.

```
public class CounterTester2 {
    public static void main(String[] args) {
        Counter c = new Counter();
        System.out.println("Current val: " + c.getValue());
        try {
            c.increment();
            c.increment();
            c.increment();
        } catch (ValueTooLargeException e) {
            System.out.println("ValueTooLargeException was thrown unexpectedly.");
        }
        System.out.println("Current val: " + c.getValue());
        try {
            c.increment();
            System.out.println("ValueTooLargeException was NOT thrown as expected.");
        } catch (ValueTooLargeException e) {
            System.out.println("ValueTooLargeException thrown as expected.");
        }
    }
}
```

0

for(int i=0; i<3; i++){ c.increment(); }

we don't expect VTLE to occur, but it did.

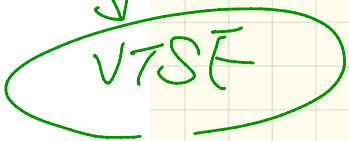
3

c.getVal()

3

Testing from Console (v2)

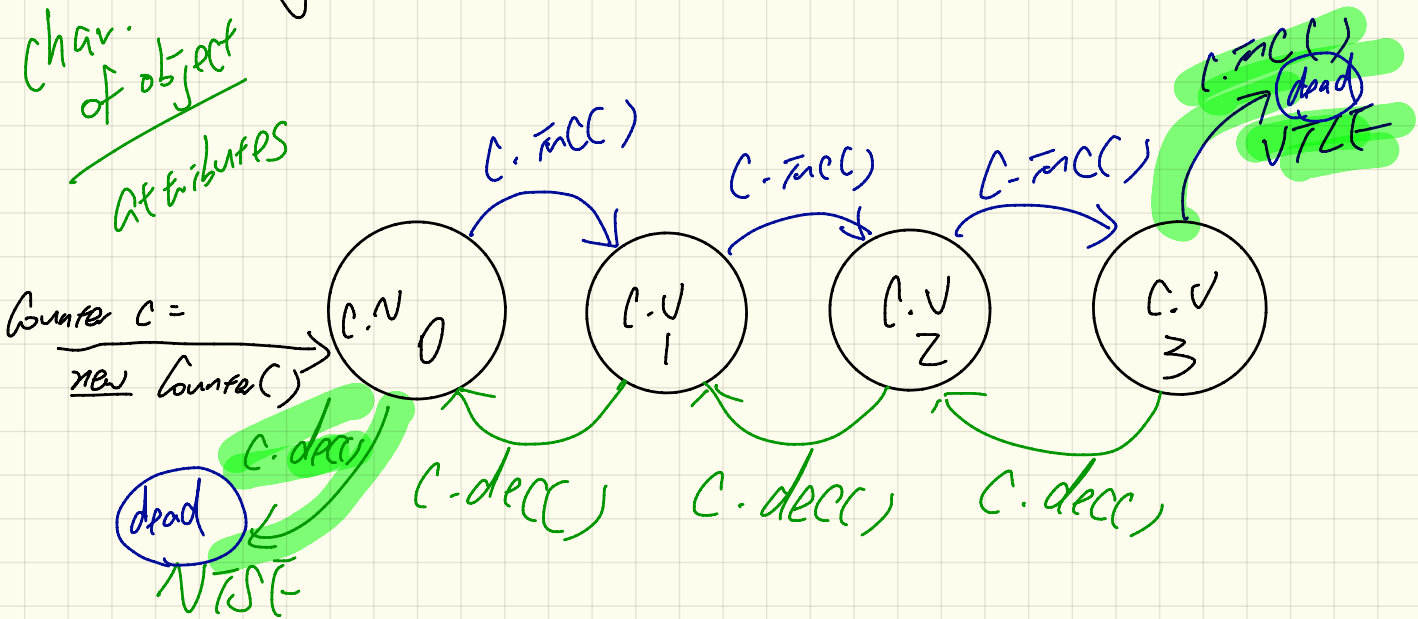
```
public class CounterTester3 {
    public static void main(String[] args) {
        Scanner input = new Scanner(System.in);
        String cmd = null;
        Counter c = new Counter();
        boolean userWantsToContinue = true;
        while (userWantsToContinue) {
            System.out.println("Enter \"inc\", \"dec\", or \"val\":");
            cmd = input.nextLine();
            try {
                if (cmd.equals("inc")) {
                    c.increment();
                } else if (cmd.equals("dec")) {
                    c.decrement();
                } else if (cmd.equals("val")) {
                    System.out.println(c.getValue());
                } else {
                    userWantsToContinue = false;
                    System.out.println("Bye!");
                }
            } catch (ValueTooLargeException e) {
                System.out.println("Value too big!");
            } catch (ValueTooSmallException e) {
                System.out.println("Value too small!");
            }
        }
        input.close();
    }
}
```



State Diagram for Counter Object

char. of object

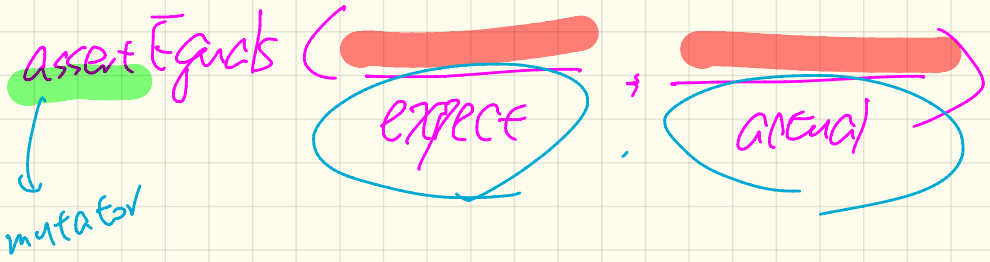
attributes



JUnit Test Case 1

```java
@Test
public void testIncAfterCreation() {
    Counter c = new Counter();
    assertTrue(Counter.MIN_VALUE == c.getValue());
    assertEquals(Counter.MIN_VALUE, c.getValue());
    assertEquals("Initial counter value is Counter.MIN_VALUE", Counter.MIN_VALUE, c.getValue());
    try {
        c.increment();
        assertEquals(1, c.getValue());
    }
    catch (ValueTooLargeException e) {
        fail("ValueTooLargeException thrown unexpectedly.");
    }
}
```

NTCE



JUnit Test Case 2

```java
@Test
public void testDecFromMinValue() {
    /*
     * This test automates what's done in CounterTester1
     */
    Counter c = new Counter();
    assertEquals(Counter.MIN_VALUE, c.getValue());
    try {
        c.decrement();
        /* reaching this line means that c.decrement() did not throw an exception */
        fail("ValueTooSmallException was NOT thrown as expected.");
    } catch (ValueTooSmallException e) {
        /*
         *  Do nothing - ValueTooSmallException thrown as expected.
         */
    }
}
```

case 1 : VTSE thrown

case 2 : VTSE not thrown

JUnit Test Case 3

```java
@Test
public void testIncFromMaxValue() {
    /*
     * This test automates what's done in CounterTester2
     */
    Counter c = new Counter();
    try {
        c.increment();
        c.increment();
        c.increment();
    } catch (ValueTooLargeException e) {
        fail("ValueTooLargeException was thrown unexpectedly.");
    }
    assertEquals("Counter reaches max", Counter.MAX_VALUE, c.getValue());
    try {
        c.increment();
        fail("ValueTooLargeException was NOT thrown as expected.");
    } catch (ValueTooLargeException e) {
        /*
         * Do nothing - ValueTooLargeException thrown as expected.
         */
    }
}
```

Question: Is this alternative version appropriate?