Lecture 10 - Oct 6

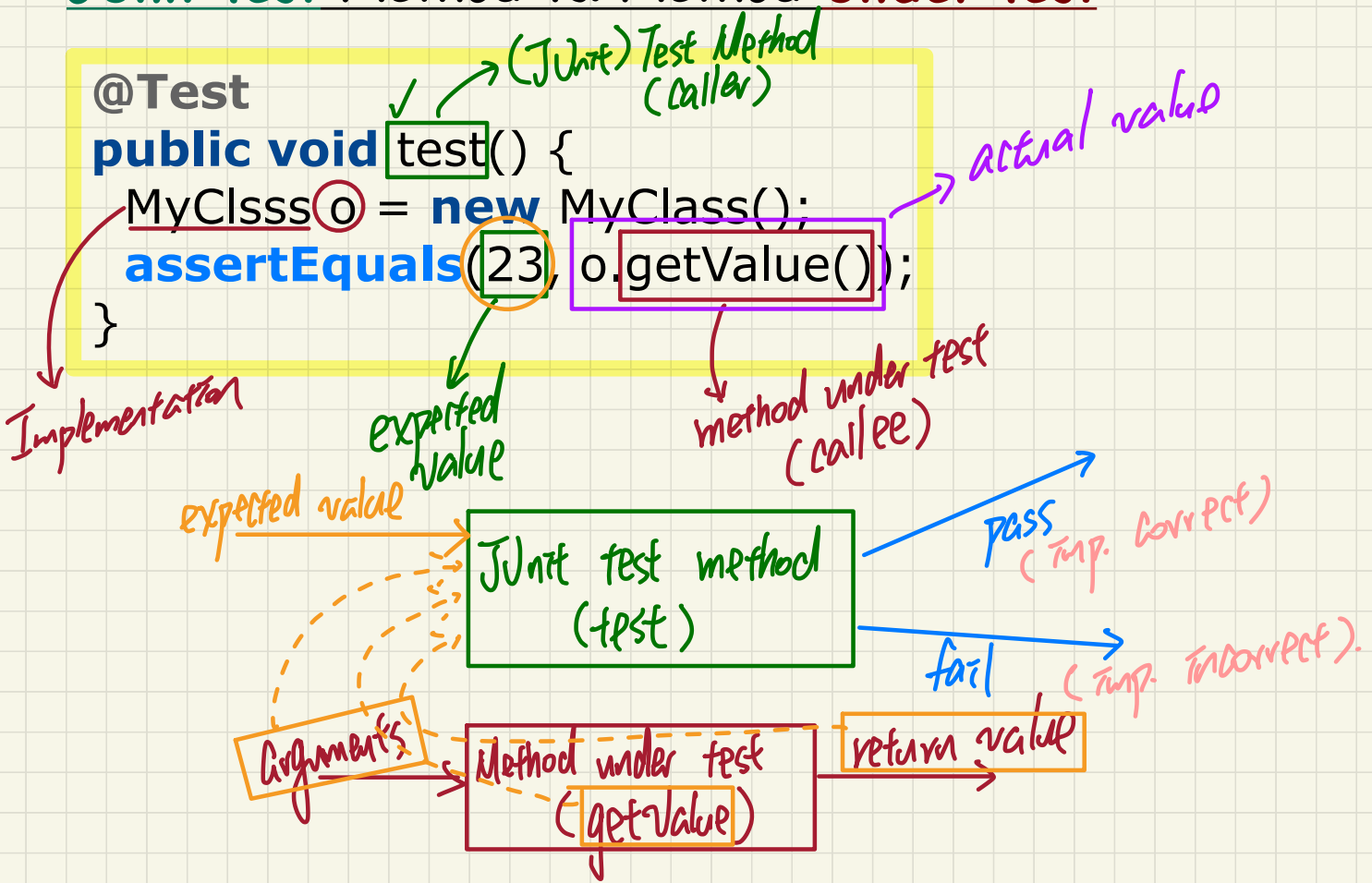
TDD with JUnit

Regression Testing
Example Test 1: VTLE Not Expected
Example Test 2: VTSE Expected

Announcements/Reminders

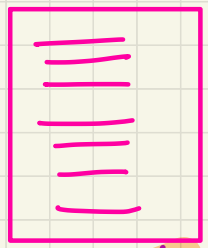
- Today's class: notes template posted
- Priorities:
  - + Lab1 solution video released
  - + Lab2 released
- ProgTest1
  - + guide released
  - + PracticeTest1 released
  - + Last Friday's Review session materials posted

JUnit Test Method vs. Method Under Test



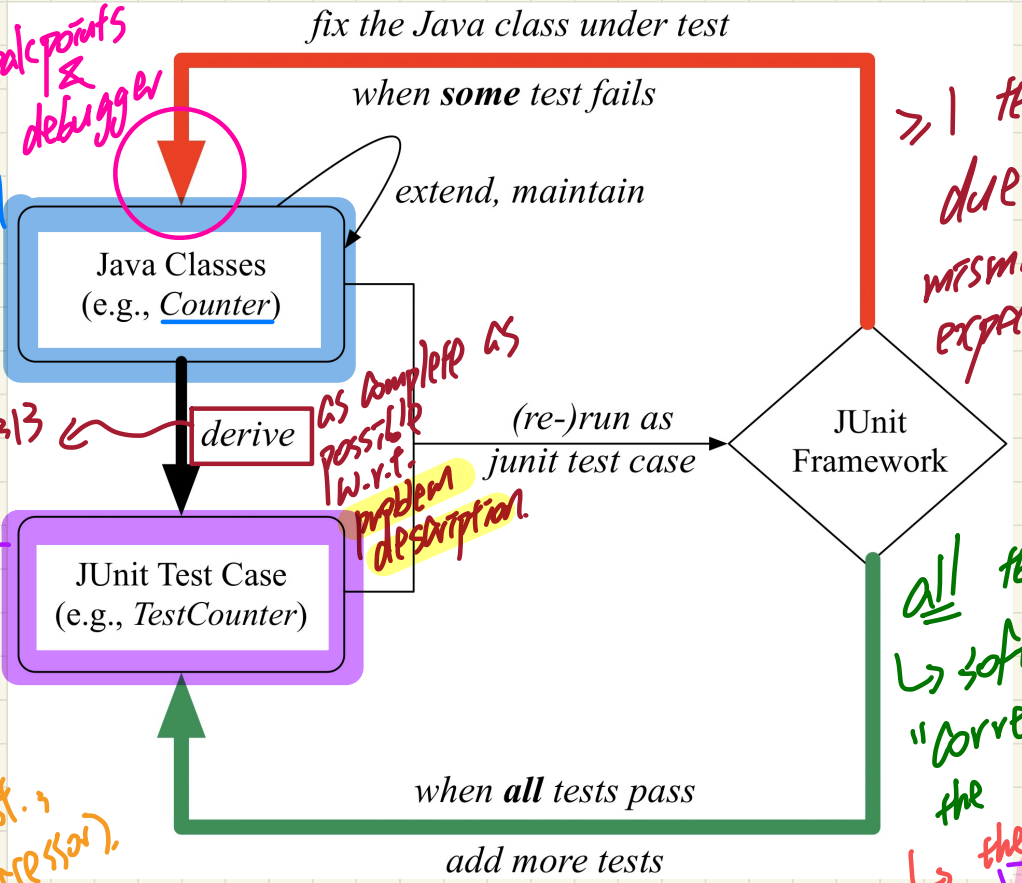
Test-Driven Development (TDD): Regression Testing

slide 11



regression.

breakpoints & debugger



model

incremental development

⇒ start testing the code as soon as it becomes "testable"

↳ some logical unit is completed (e.g. const., accessor).

EECS4313

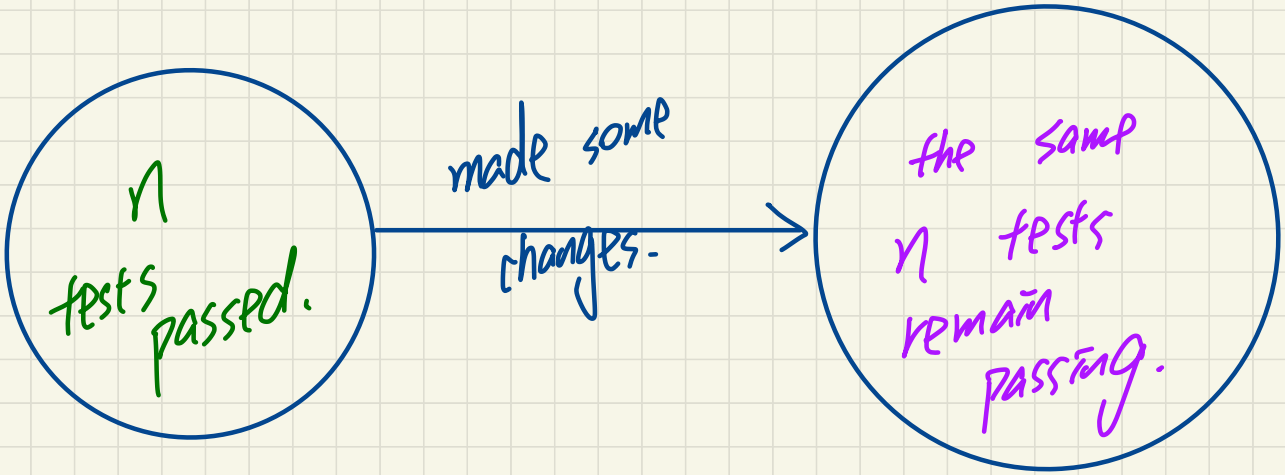
as complete as possible w.r.t. problem description.

≥ 1 test failed due to a mismatch between expected & actual values.

all tests pass

↳ software is "correct" w.r.t. the tests run.

↳ the more & diverse tests you have, the better


n
tests passed.
made some changes.
the same n tests remain passing.

Self Studies

Creating & Running JUnit Test Cases: Slides 12 to 21

Common Assertions in JUnit: Slide 22

Examples on JUnit: Slides 23 - 25

A Default Test Case that Fails

slide 16

The result of running a test is considered:

- ***Failure*** if either
  - an ***assertion failure*** (e.g., caused by `fail`, `assertTrue`, `assertEquals`) occurs
  - an unexpected ***exception*** (e.g., `NullPointerException`, `ArrayIndexOutOfBoundException`) thrown
- ***Success*** if neither ***assertion failures*** nor (unexpected) ***exceptions*** occur.

fail(); ↳ cause the test to fail unconditionally & immediately.

slide 19

TestCounter.java

```
1  package tests;
2  import static org.junit.Assert.*;
3  import org.junit.Test;
4  public class TestCounter {
5      @Test
6      public void test() {
7          //fail("Not yet implemented");
8      }
9  }
10
```

no exception no assertion failure occurred.

after commenting out fail assertion, the test passes trivially

Q: What is the easiest way to making this test pass?

Design of a (JUnit) Test

assess if some implementation is correct.

(e.g. inc when counter value is max)

→ Expectation: certain exception occurs

↳ that kind of exception is thrown by imp. → pass

↳ that kind of exception is not thrown by imp. → fail

not expected

→ Expectation: certain exception does not occur

(e.g. inc when counter value is 0)

not expecting

↳ that kind of exception is thrown → fail

↳ that kind of exception is not thrown → pass

JUnit: An Exception Not Expected

```
1  @Test
2  public void testIncAfterCreation() {
3    Counter c = new Counter();
4    assertEquals(Counter.MIN_VALUE, c.getValue());
5    try {
6      c.increment();
7      assertEquals(1, c.getValue());
8    }
9    catch(ValueTooLargeException e) {
10     /* Exception is not expected to be thrown. */
11     fail("ValueTooLargeException is not expected.");
12   }
13 }
```

may throw VTLE
VTLE did not occur
pass the test

```
1  @Test
2  public void testIncAfterCreation() {
3    Counter c = new Counter();
4    assertEquals(Counter.MIN_VALUE, c.getValue());
5    try {
6      c.increment();
7      assertEquals(1, c.getValue());
8    }
9    catch(ValueTooLargeException e) {
10     /* Exception is not expected to be thrown. */
11     fail("ValueTooLargeException is not expected.");
12   }
13 }
```

may throw VTLE
VTLE occurred
fail the test

What if increment is implemented correctly?

does not throw VTLE

Expected Behaviour:

Calling c.increment()
when c.value is 0 should **not**
trigger a ValueTooLargeException

not expected.

What if increment is implemented **incorrectly**?

throw VTLE.

e.g., It throws VTLE when
c.value < Counter.MAX_VALUE

Running JUnit Test 1 on Correct Implementation

Expectation: VTLE does not occur

c.v == 0

```
public void increment() throws ValueTooLargeException {
  if(value == Counter.MAX_VALUE) {
  throw new ValueTooLargeException("counter value is " + value);
  }
  else { value ++; }
}
```

0 == 3 ; 0 → 1.

```
1  @Test
2  public void testIncAfterCreation() {
3  Counter c = new Counter();
4  assertEquals(Counter.MIN_VALUE, c.getValue());
5    try {
6    c.increment();
7    assertEquals(1, c.getValue());
8    }
9    catch(ValueTooLargeException e) {
10     /* Exception is not expected to be thrown. */
11     fail ("ValueTooLargeException is not expected.");
12   }
13 }
```

c.v == 0.

may throw VTLE

did not throw VTLE as expected

test passes because the given imp. is correct.

pass

Running JUnit Test 1 on Incorrect Implementation

```
public void increment() throws ValueTooLargeException {
  if(value < Counter.MAX_VALUE) {
    throw new ValueTooLargeException("counter value is " + value);
  }
  else { value ++; }
}
```

```
1  @Test
2  public void testIncAfterCreation() {
3    Counter c = new Counter();
4    assertEquals(Counter.MIN_VALUE, c.getValue());
5    try {
6      c.increment();
7      assertEquals(1, c.getValue());
8    }
9    catch(ValueTooLargeException e) {
10     /* Exception is not expected to be thrown. */
11     fail("ValueTooLargeException is not expected.");
12   }
13 }
```

JUnit: An Exception Expected

```
1  @Test
2  public void testDecFromMinValue() {
3    Counter c = new Counter();
4    assertEquals(Counter.MIN_VALUE, c.getValue());
5    try {
6      c.decrement();
7      fail ("ValueTooSmallException is expected.");
8    }
9    catch(ValueTooSmallException e) {
10     /* Exception is expected to be thrown. */
11   }
12 }
```

may throw VTSE
if the VTSE is thrown as expected
pass

What if decrement is implemented correctly?

Expected Behaviour:

Calling c.decrement() when c.value is 0 should trigger a ValueTooSmallException.

```
1  @Test
2  public void testDecFromMinValue() {
3    Counter c = new Counter();
4    assertEquals(Counter.MIN_VALUE, c.getValue());
5    try {
6      c.decrement();
7      fail ("ValueTooSmallException is expected.");
8    }
9    catch(ValueTooSmallException e) {
10     /* Exception is expected to be thrown. */
11   }
12 }
```

may throw VTSE
if VTSE is not thrown
fail the test
the expected VTSE is not thrown.

What if decrement is implemented incorrectly?

e.g., It only throws VTSE when c.value < Counter.MIN_VALUE

Running JUnit Test 2 on Correct Implementation

Exception VTSE occurs.

c.v == 0

```
public void decrement() throws ValueTooSmallException {
  if(value == Counter.MIN_VALUE) {
    throw new ValueTooSmallException("counter value is " + value);
  }
  else { value --; }
 }
}
```

VTSE is expected

∴ the test passes given the implementation is correct

throws VTSE as expected.

```
1  @Test
2  public void testDecFromMinValue() {
3    Counter c = new Counter();
4    assertEquals(Counter.MIN_VALUE, c.getValue());
5    try {
6      c.decrement();
7      fail("ValueTooSmallException is expected.");
8    }
9    catch(ValueTooSmallException e) {
10     /* Exception is expected to be thrown. */
11   }
12 }
```

c.v == 0

VTSE thrown as expected

pass

Running JUnit Test 2 on Incorrect Implementation

Expectation
VTSE occurs

C.V == 0

```
public void decrement() throws ValueTooSmallException {
  if(value == Counter.MIN_VALUE) {
    throw new ValueTooSmallException("counter value is " + value);
    }
   else { value --; }
  }
}
```

0 → -1

Incorrect Implementation.

```
1  @Test
2  public void testDecFromMinValue() {
3    Counter c = new Counter();
4    assertEquals(Counter.MIN_VALUE, c.getValue());
5    try {
6      c.decrement();
7      fail("ValueTooSmallException is expected.");
8    }
9    catch(ValueTooSmallException e) {
10     /* Exception is expected to be thrown. */
11   }
12 }
```

C.V == 0

VTSE not thrown

fail

the test fails :(

the given impl. is Incorrect

VTSE not thrown when dec. counter with value 0.

JUnit: Exception Sometimes Expected, Somtimes Not

```java
@Test
public void testIncFromMaxValue() {
  Counter c = new Counter();
  try {
    c.increment(); c.increment(); c.increment();
  }
  catch (ValueTooLargeException e) {
    fail("ValueTooLargeException was thrown unexpectedly.");
  }
  assertEquals(Counter.MAX_VALUE, c.getValue());
  try {
    c.increment();
    fail("ValueTooLargeException was NOT thrown as expected.");
  }
  catch (ValueTooLargeException e) {
    /* Do nothing: ValueTooLargeException thrown as expected. */
  }
}
```

P1: VTLE not expected

P2: VTLE expected

Expected Behaviour:

Calling c.increment()
3 times to reach c's max should not
trigger any ValueTooLargeException.

Calling c.increment()
when c is already at its max should
trigger a ValueTooLargeException