September 19

Tuesday

Lecture 4

② ✗ e.g. counter == 1 . 1 <= 3 → T

↳ exception is thrown.

```
increment(int value) {
    if ( ① counter > MAX
         ② counter <= MAX
         ③ counter >= MAX ) {
        throw
    } else {
        # counter ++;
    }
}
```

else : counter ≤ MAX

counter < MAX

③ ✓

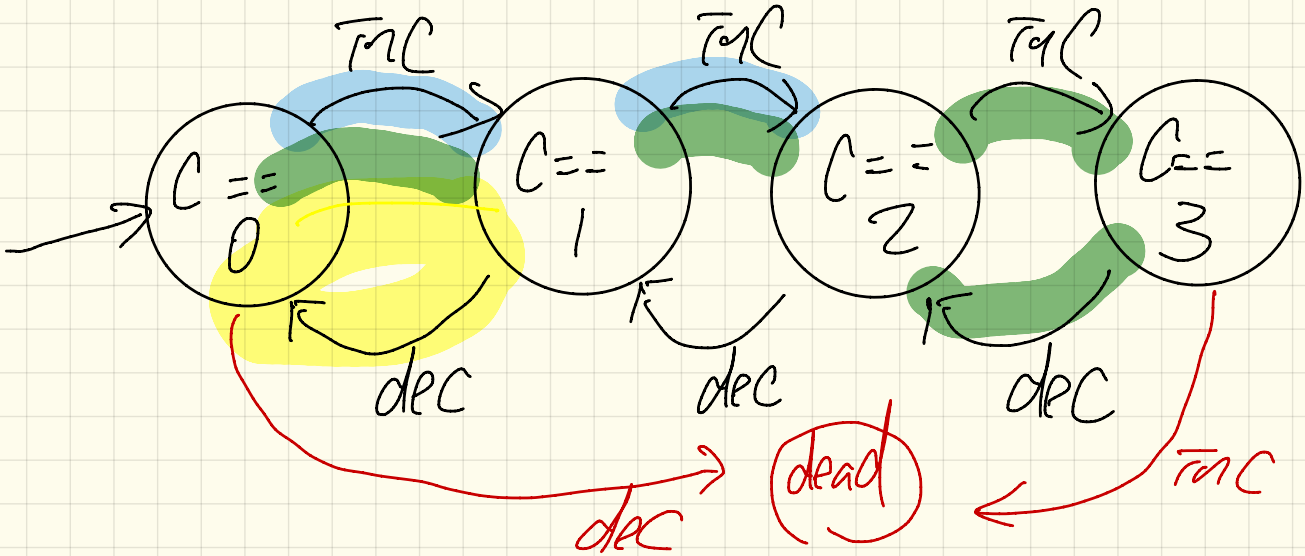
① ✗ : if counter == MAX
MAX > MAX → False

State Machine

How many test cases?
How many paths?

Transitions (mutator method calls)

[state] values of attributes



Criteria for judging your program

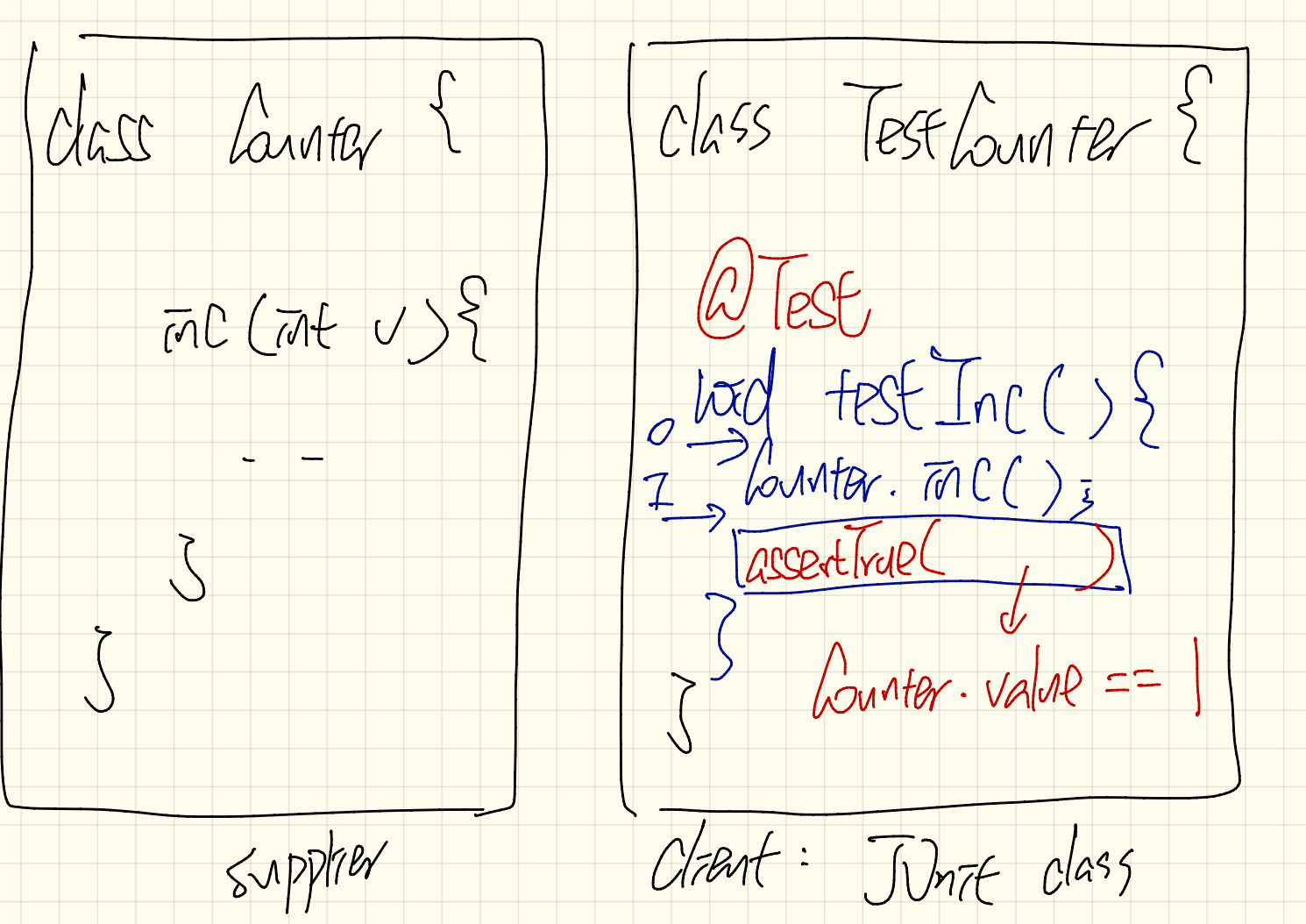
① Correctness

② Efficiency.

Opt A. ① ∧ ¬②

Opt B. ¬① ∧ ②

define a reasonably complete set of tests.

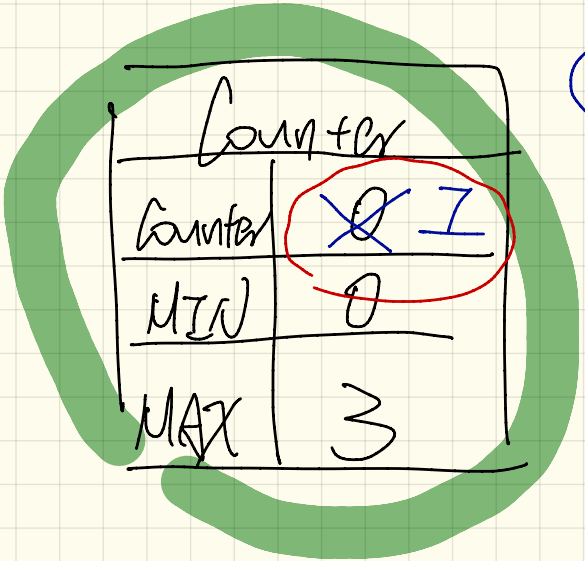

class Counter {
inc(int v){
}
}
supplier
class TestCounter {
@Test
void testInc(){
Counter.inc();
assertTrue(
Counter.value == 1
}
}
0
1
client: JUnit class

Now we want to test an abnormal use of the class, for which we expect a precondition violation to occur.

```
@Test
void testDecFromZero() {
    try {
        counter.value = MIN;
        // init val is 0
        counter.dec(); // expect exception
        // We do not expect to reach this line
        fail("no precondition violation occurred");
    } catch (IAE e) {
        // precondition violation occurred as expected
    }
}
```

breakpoints and debugger

↳ slow down your program and execute line by line-



① testOneIncrement()

② testDecFromZero()

When starting each test method we should reset the attributes to their initial values to make sure it's a fresh start.