cs304
Software Engineering

TAN, Shin Hwei

陈馨慧

Southern University of Science and Technology

Slides adapted from cs427 (UIUC) and cs304(SUSTech)

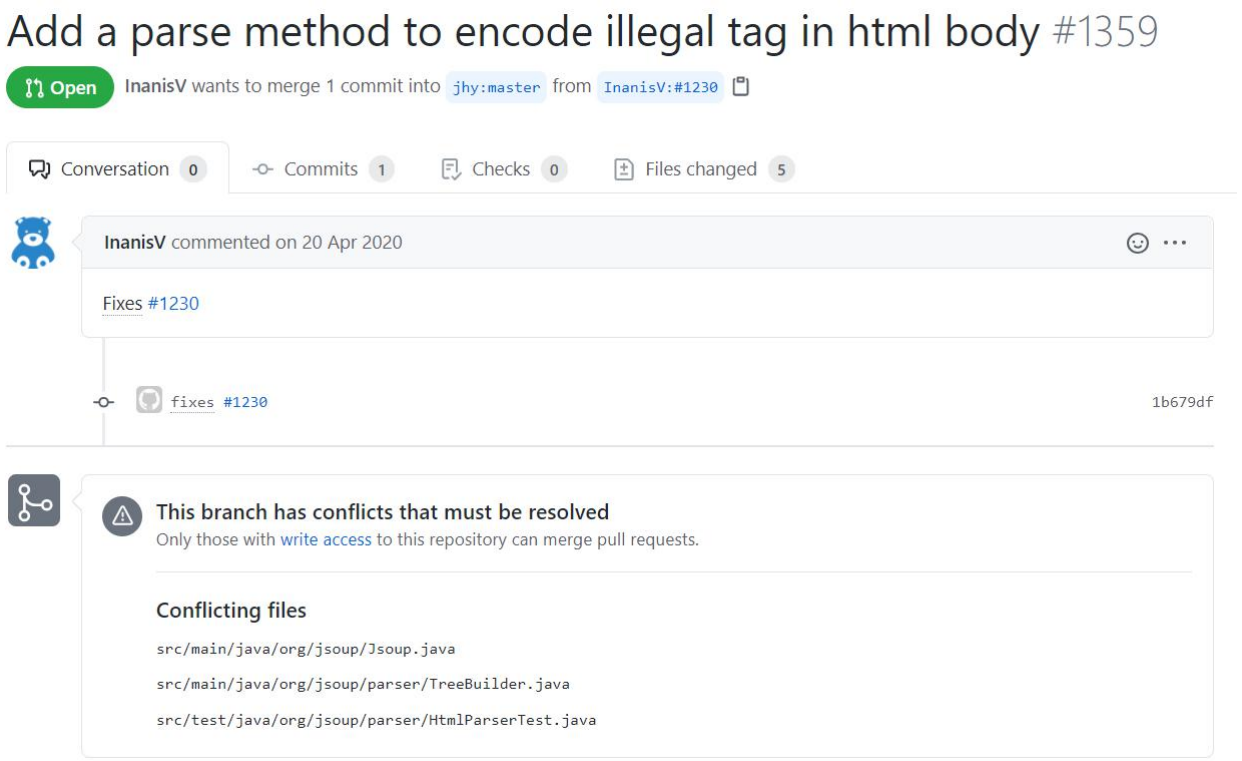
Administrative Info

- **The deadline for MP0 and Project Proposal has passed**. Late submission get 0 score!
- **Progress Report has been posted due on** April 24
- **All assignments should be written in English**
  - One exception: The selected issues could be written by the developers in Chinese
- **All lab exercise should be submitted before next lab to avoid accumulating too much assignments**
- **Attend lab today for coverage lab!**

Project

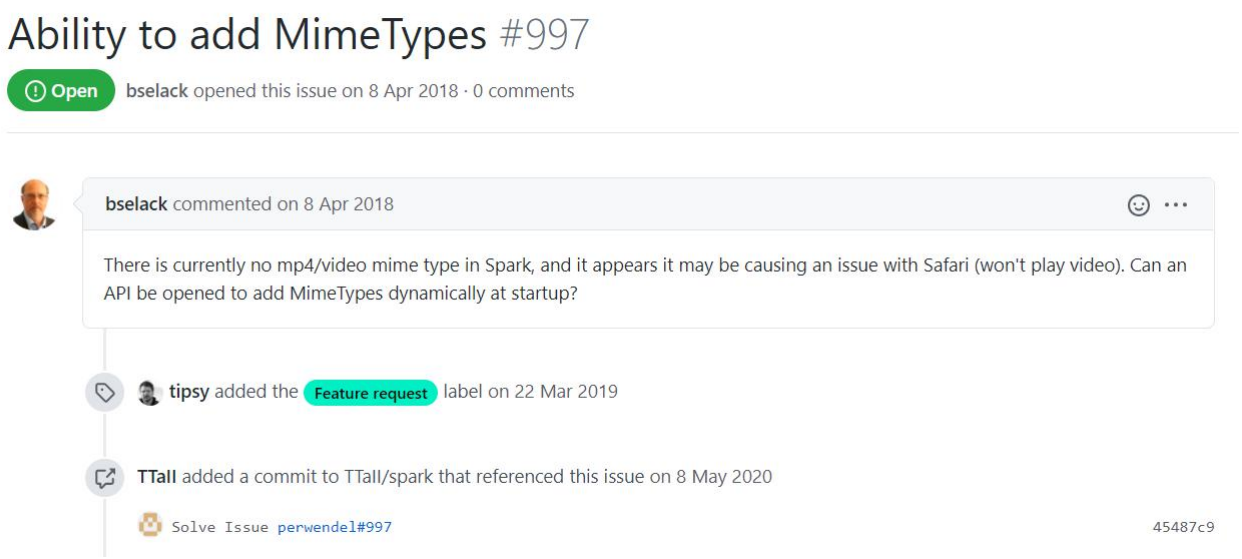
Rules for submitting pull requests

Project Rules:
What not to do when making pull request (PR)?



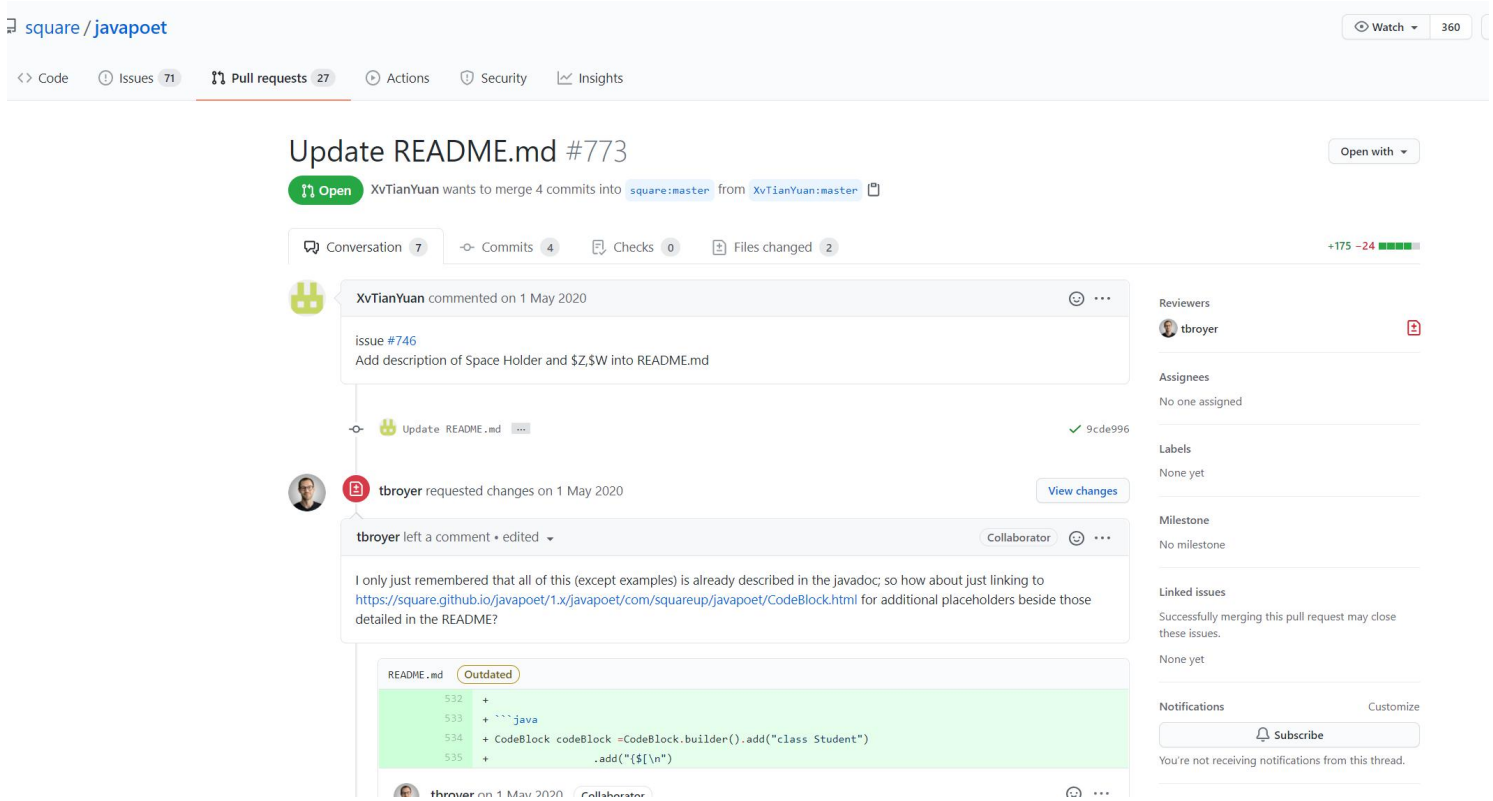
- Need to change the code to resolve conflicts so that developers can merge cleanly

Project Rules:
What not to do when making pull request (PR)?



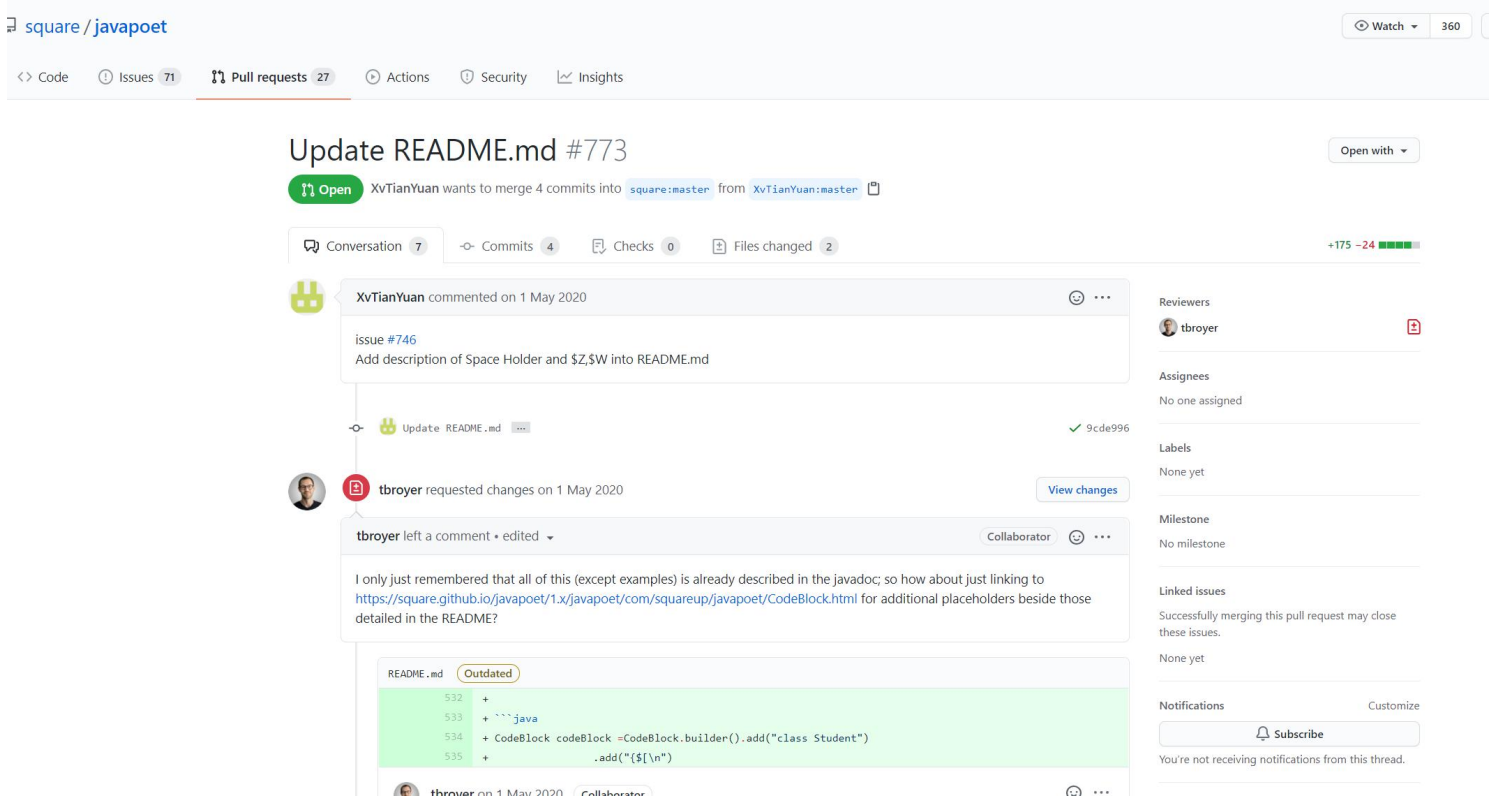
- No communication with the user and developer. Only fixing the way you like

Project Rules:
What not to do when making pull request (PR)?



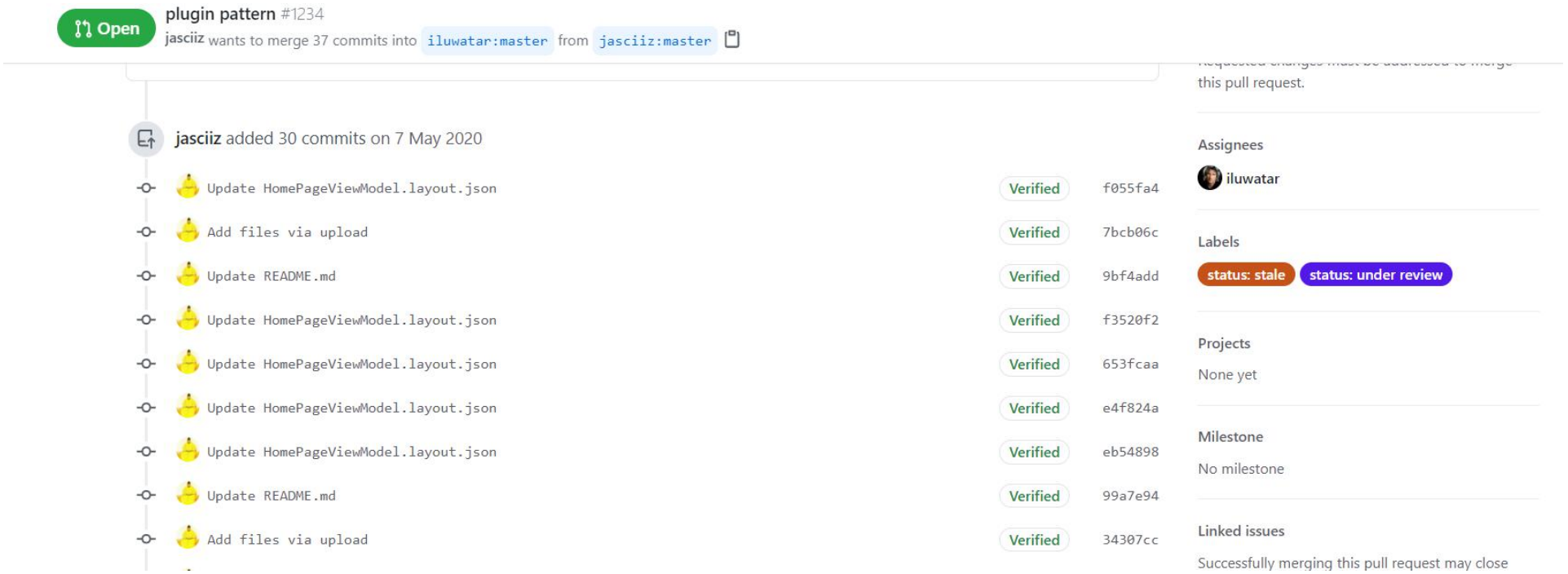
➢Do not choose documentation related issue.

Project Rules:
What not to do when making pull request (PR)?



➢Do not choose documentation related issue.

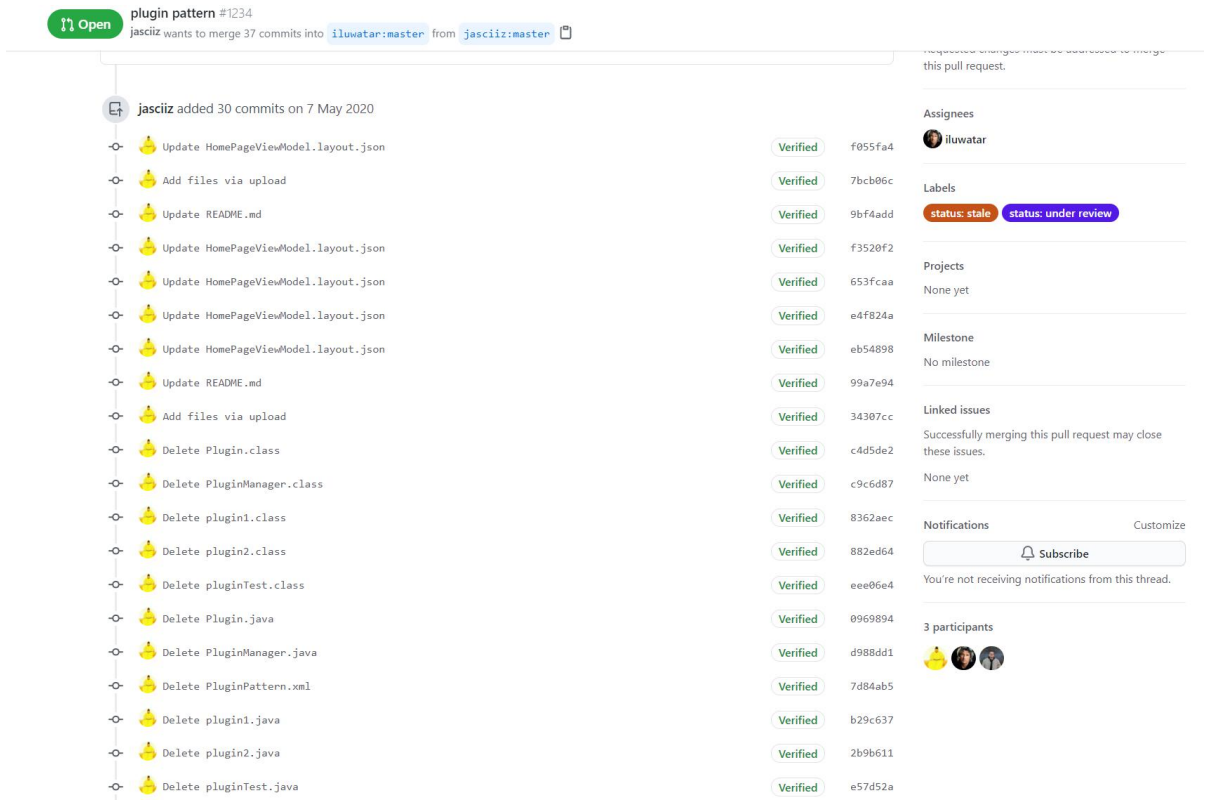
Project Rules:
What not to do when making pull request (PR)?



- Do not adding too many commits to your PR. You PR should be short
- Make minimal changes to current code. A PR that change less lines will be more likely to be accepted!

Example from: https://github.com/iluwatar/java-design-patterns/pull/1234

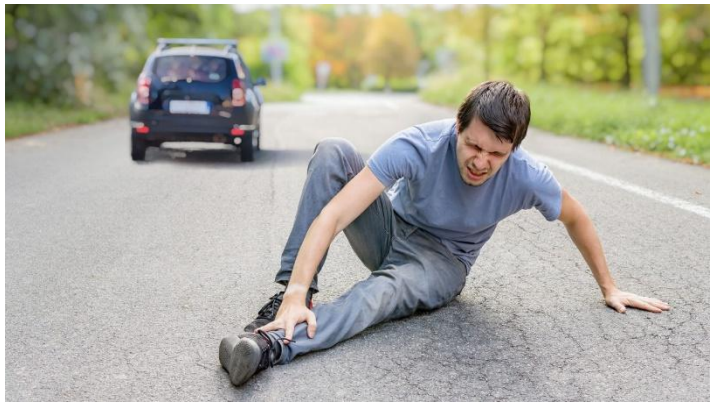
Project Rules:
What not to do when making pull request (PR)?

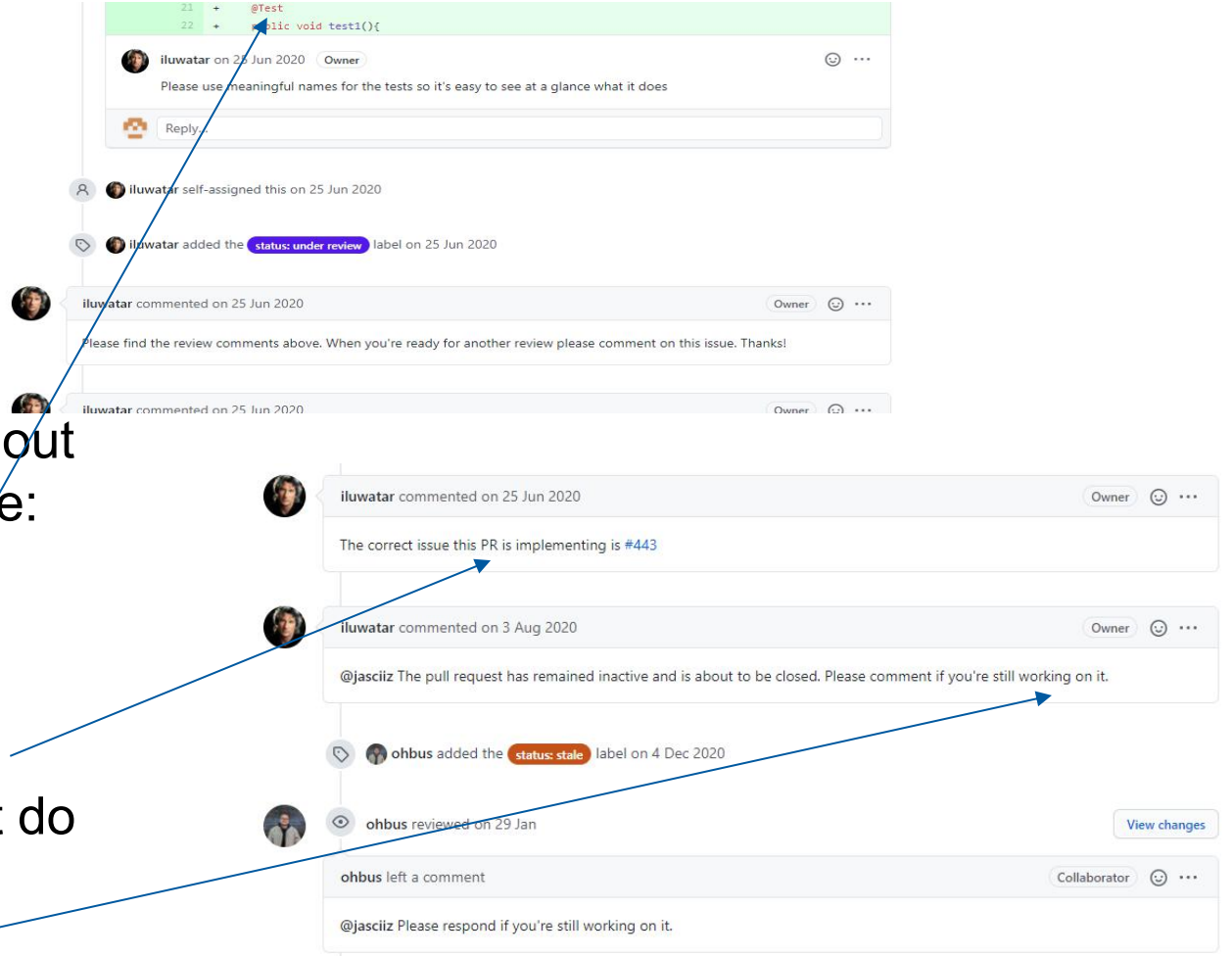


- Do not adding too many commits to your PR. You PR should be short

From: https://github.com/iluwatar/java-design-patterns/pull/1234

Project Rules:
What not to do when making pull request (PR)?





- **Hit-and-run PR**: Commits and run away without responding. This is very bad practice because:
  - Waste time of developer in code review
  - Developers need to correct your mistake because you didn't even check if you are implementing the correct issue
  - Developer mention you several times but do not respond. Leave a bad reputation in GitHub

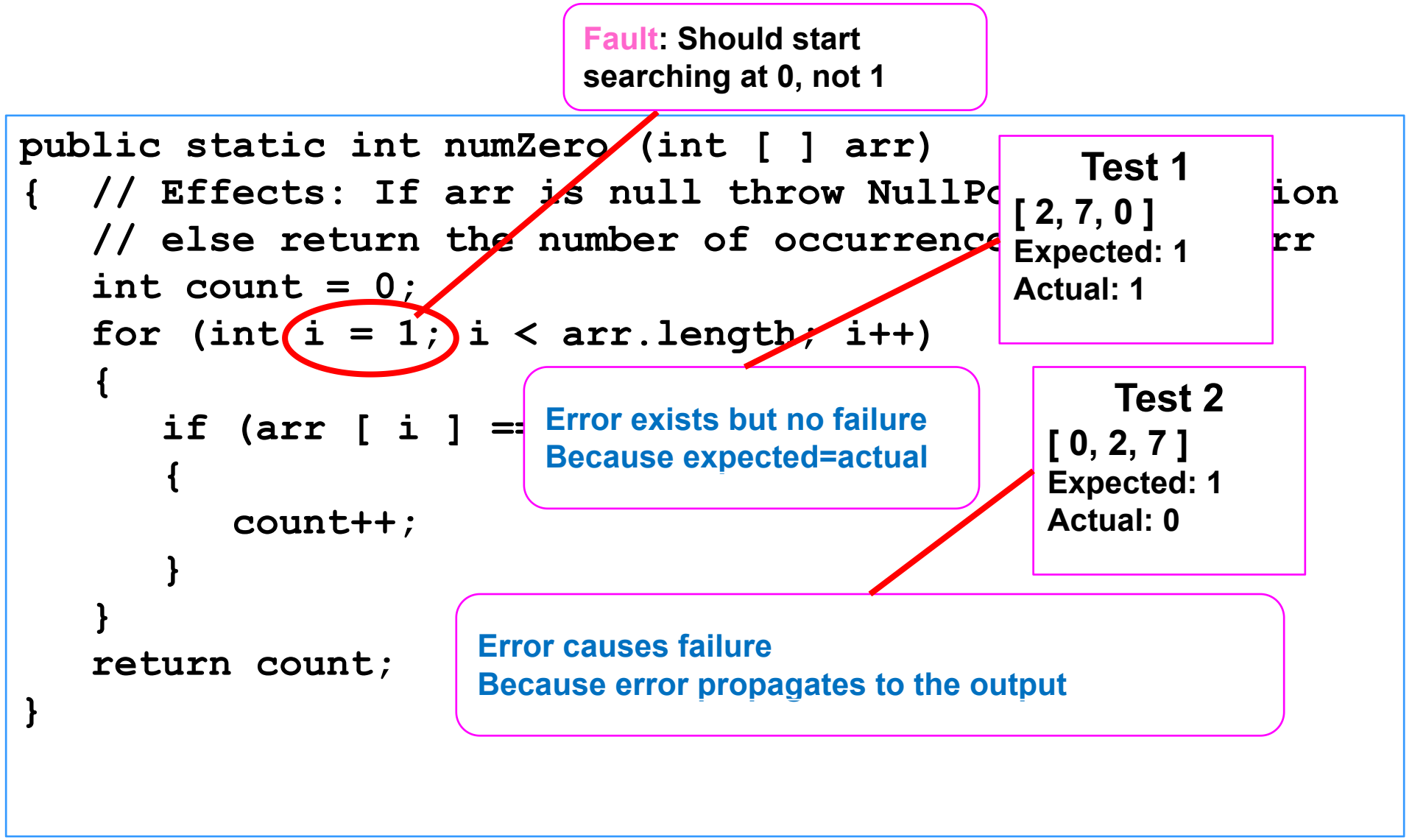
Example from: https://github.com/iluwatar/java-design-patterns/pull/1234

Recap: Failure and errors

- Q:Why is the difference between failure and errors?
- A:
  - Error：计算、观察或测量值或条件，与真实、规定或理论上正确的值或条件之间的差异（Discrepancy between a computed, observed or measured value or condition and the true, specified, or theoretically correct value or condition.），可译为“错误”。Error是能够导致系统出现Failure的系统内部状态。
  - Failure：当一个系统不能执行所要求的功能时，即为Failure，可译为“失效”。（Termination of the ability of an element or an item to perform a function as required.）

From: https://blog.csdn.net/yangxingpa/article/details/70754788

Recap Example



Test Driven Development (TDD)

One of the core practices in XP

Kent Beck's rules

- Beck's concept of test-driven development centers on two basic rules:
  - Never write a single line of code unless you have a failing automated test.
  - Eliminate duplication.

Informal Requirements

Maintenance: The Maintenance function records the history of items undergoing maintenance.

If the product is covered by warranty or maintenance contract, maintenance can be requested either by calling the maintenance toll free number, or through the web site, or by bringing the item to a designated maintenance station.

If the maintenance is requested by phone or web site and the customer is a US or EU resident, the item is picked up at the customer site, otherwise, the customer shall ship the item with an express courier.

If the maintenance contract number provided by the customer is not valid, the item follows the procedure for items not covered by warranty.

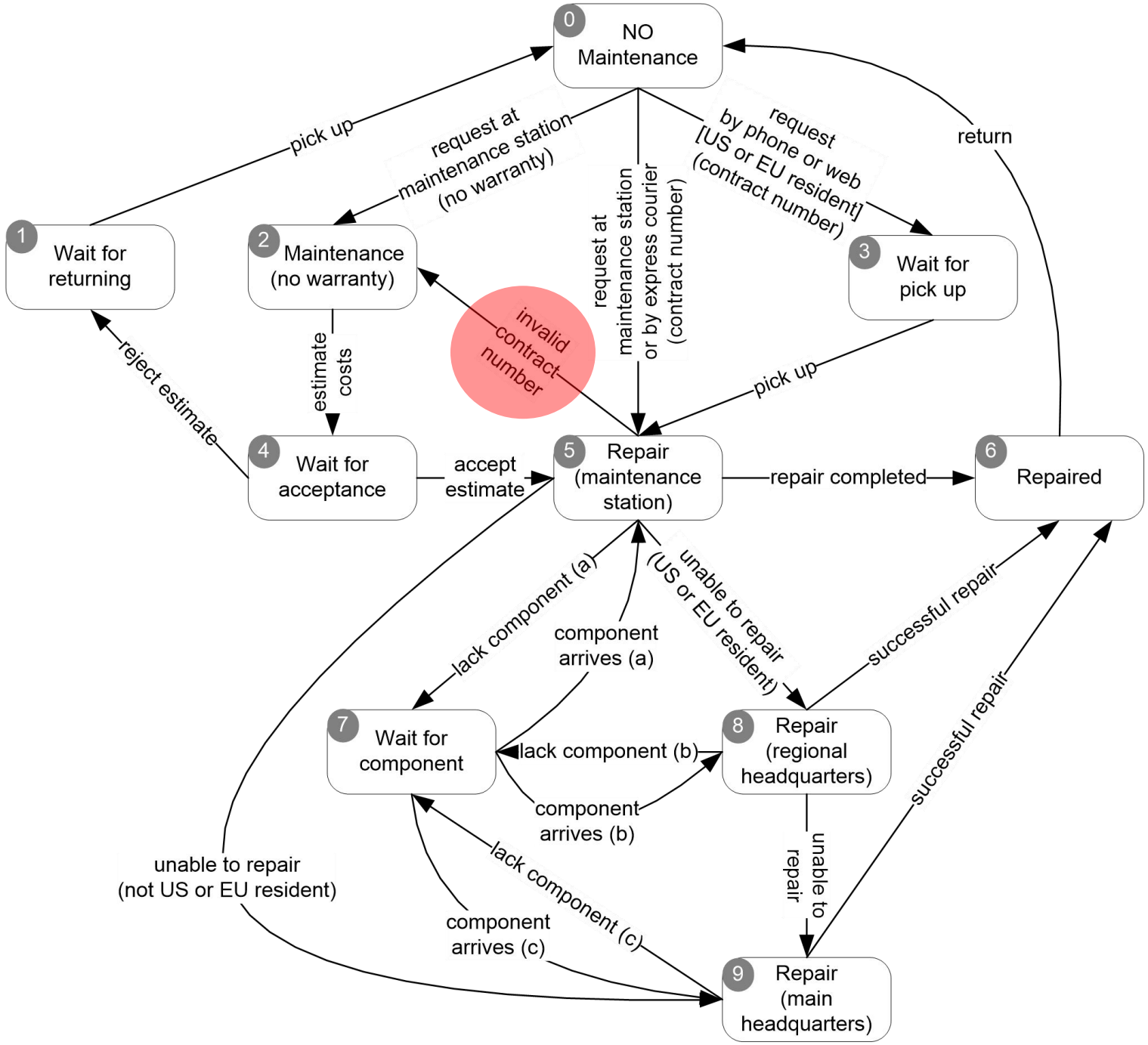
If the product is not covered by warranty or maintenance contract, maintenance can be requested

maintenance main headquarters.

Maintenance is suspended if some components are not available.

Once repaired, the product is returned to the customer.

If the maintenance contract number provided by the customer is not valid, the item follows the procedure for items not covered by warranty.


0
NO Maintenance
1
Wait for returning
2
Maintenance (no warranty)
3
Wait for pick up
4
Wait for acceptance
5
Repair (maintenance station)
6
Repaired
7
Wait for component
8
Repair (regional headquarters)
9
Repair (main headquarters)
pick up
request at maintenance station (no warranty)
request at maintenance station or by express courier (contract number)
request by phone or web [US or EU resident] (contract number)
return
reject estimate
estimate costs
invalid contract number
pick up
accept estimate
repair completed
lack component (a)
component arrives (a)
unable to repair (US or EU resident)
successful repair
lack component (b)
component arrives (b)
unable to repair (not US or EU resident)
lack component (c)
component arrives (c)
unable to repair
successful repair

Ambiguity in Informal Requirements

If the maintenance contract number provided by the customer is not valid

- Contract number cannot contain alphabets or special characters?
- Contract number must be 5 digits?
- Contract number cannot start with 0?

Requirements based on Test Cases

```
@Test
public void testContractNumberCorrectLength() {
        assertTrue(contract.isValidContractNumber("12345"));
}
```

```
@Test
public void testContractNumberTooLong() {
        assertFalse(contract.isValidContractNumber("53434434343"));
}
```

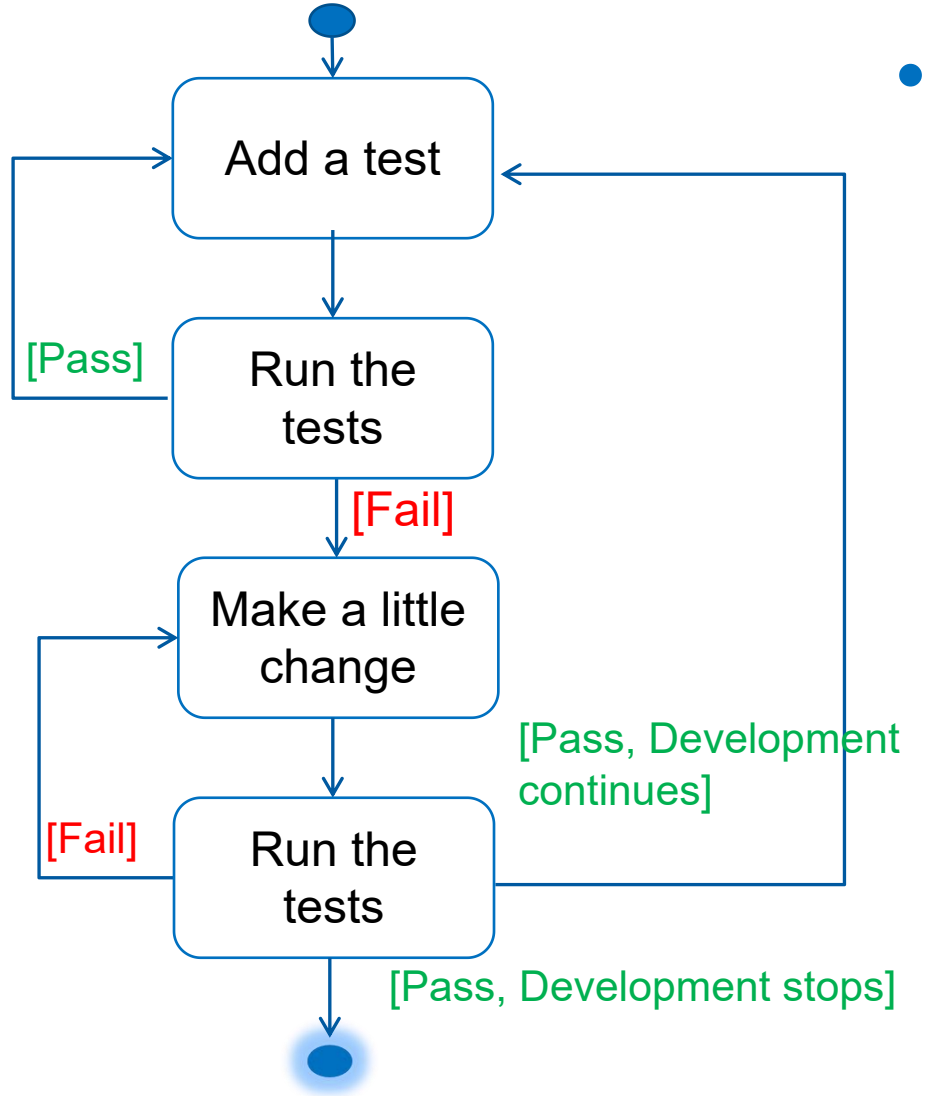
```
@Test
public void testContractNumberNoSpecialCharacter() {
        assertTrue(contract.isValidContractNumber("08067"));
}
```

```
@Test
public void testContractNumberWithSpecialCharacter() {
        assertFalse(contract.isValidContractNumber("98&67"));
}
```

Informal Requirements versus Test cases

- Test cases are more specific than requirements.
- But: How to develop code based on test cases?
  - Follow the steps in Test Driven Development(TDD)

Steps in Test Driven Development (TDD)



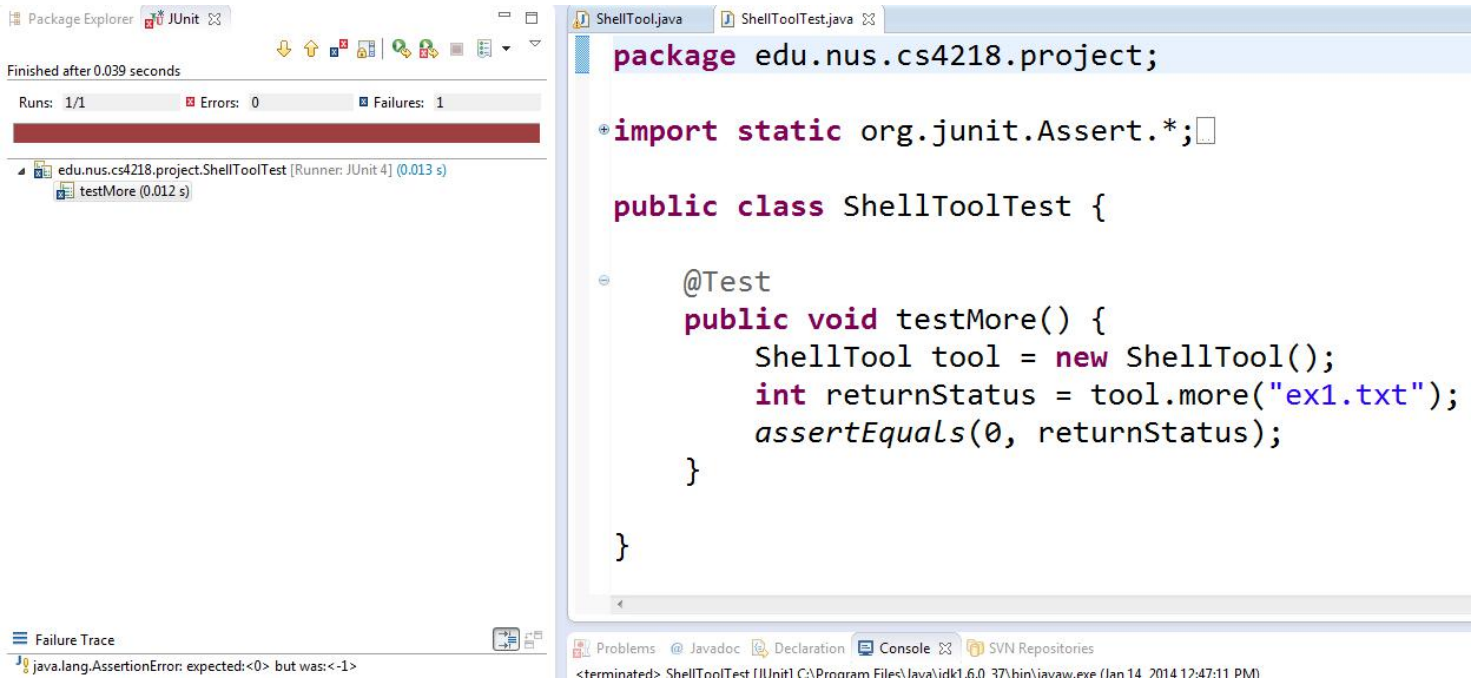
- The iterative process
  - Quickly add a test.
  - Run all tests and see the new one fail.
  - Make a little change to code.
  - Run all tests and see them all succeed.
  - Refactor to remove duplication.

Test First Scenario

- Write test for the newly added functionality
  - These test cases will serve as a specification for your implementation
  - These test cases should fail now because the corresponding methods are not implemented
  - Write minimal code to make the test pass
  - Add more tests

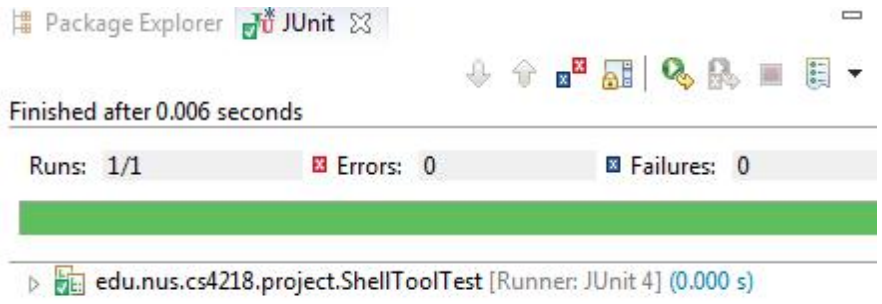
Run the tests

- Run the tests that your team gets to see the failing ones
  - Failing test cases indicates missing functionality



Make them Pass

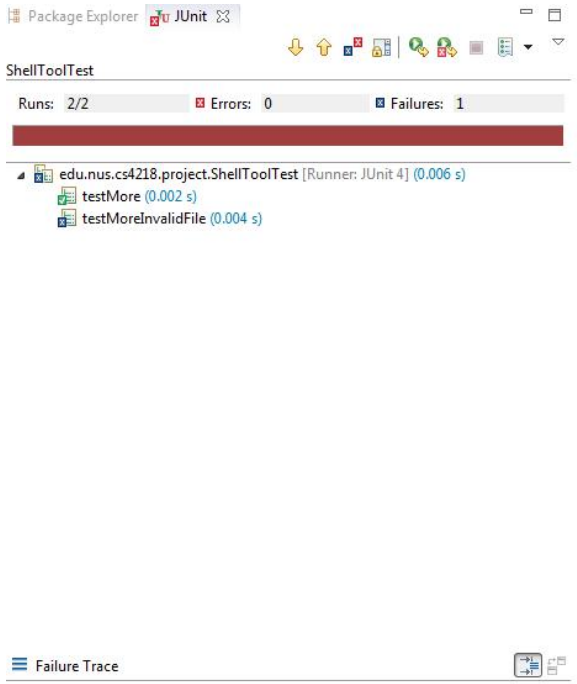
- Add code to make the failing tests pass
  - After implementing all the missing functionalities, all failing test cases should now pass




```
    /**
     * Shows the first part of a file
     *
     * @param path
     * @return
     */
    public int more(String path) {
        BufferedReader in = null;
        try {
            in = new BufferedReader(new FileReader(path));
            String line;
            try {
                while ((line = in.readLine()) != null) {
                    System.out.println(line);
                }
            } catch (IOException e) {
                e.printStackTrace();
            }
        } catch (FileNotFoundException e) {
            e.printStackTrace();
        }
        return 0;
    }
}
```

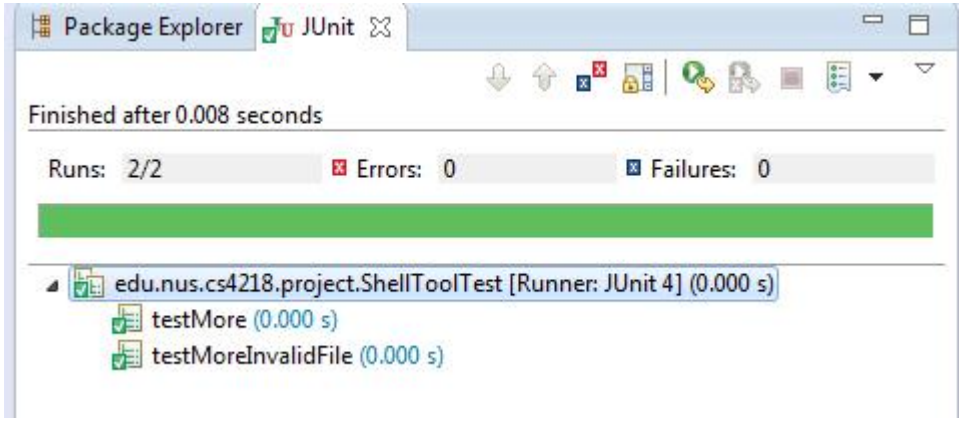
Add more tests

- Add more tests for
  - Newly added components or helper methods
  - Checking for corner cases
- Run the new set of tests to see the failing ones



Make them Pass

- Add more code to make the added tests pass
  - After implementing all the helper methods and checking for corner cases, all new failing test cases should now pass



Lesson Learned from Pair Testing

How to write good test?
How to create a good test suite?

Which example has better tests?

Example 1

```
@Test
public void popTest() {
 MyStack s = new MyStack ();
  s.push (314);
  assertEquals (314, s.pop ());
}
@Test
public void sizeTest() {
 MyStack s = new MyStack ();
 s.push (2);
 assertEquals (1, s.size ());
}
```

Example 2

```
MyStack s = new MyStack ();
@Test
public void popTest() {
 s.push (314);
 assertEquals (314, s.pop ());
}

@Test
public void sizeTest() {
 s.push (2);
 assertEquals (1, s.size ());
}
```

Which example has better tests?

Example 1

```
@Test
public void sizeTest() {
  MyStack s = new MyStack ();
  assertEquals (0, s.size ());
  s.push (2);
  assertEquals (1, s.size ());
}
```

Example 2

```
@Test
public void emptyTest() {
  MyStack s = new MyStack ();
  assertEquals (0, s.size ());
}

@Test
public void sizeTest() {
  MyStack s = new MyStack ();
  s.push (2);
  assertEquals (1, s.size ());
}
```

Which tests is correct?

Example 1

```
@Test
public void sizeTest() {
 MyStack s = new MyStack ();
 assertEquals (s.size (),0);
}
```

Example 2

```
@Test
public void emptyTest() {
  MyStack s = new MyStack ();
  assertEquals (0, s.size ());
}
```

Which tests is correct?

Example 1

```
@Test
public void sizeTest() {
 MyStack s = new MyStack ();
 s.push(“Hello”);
 assertEquals (“Hello”, s.pop());
}
```

Example 2

```
@Test
public void emptyTest() {
 MyStack s = new MyStack ();
 s.push(“Hello”);
 assertTrue (s.pop()==”Hello”);
}
```

Is there a standard measurement for test quality?

Yes, code coverage!

What is Code Coverage?

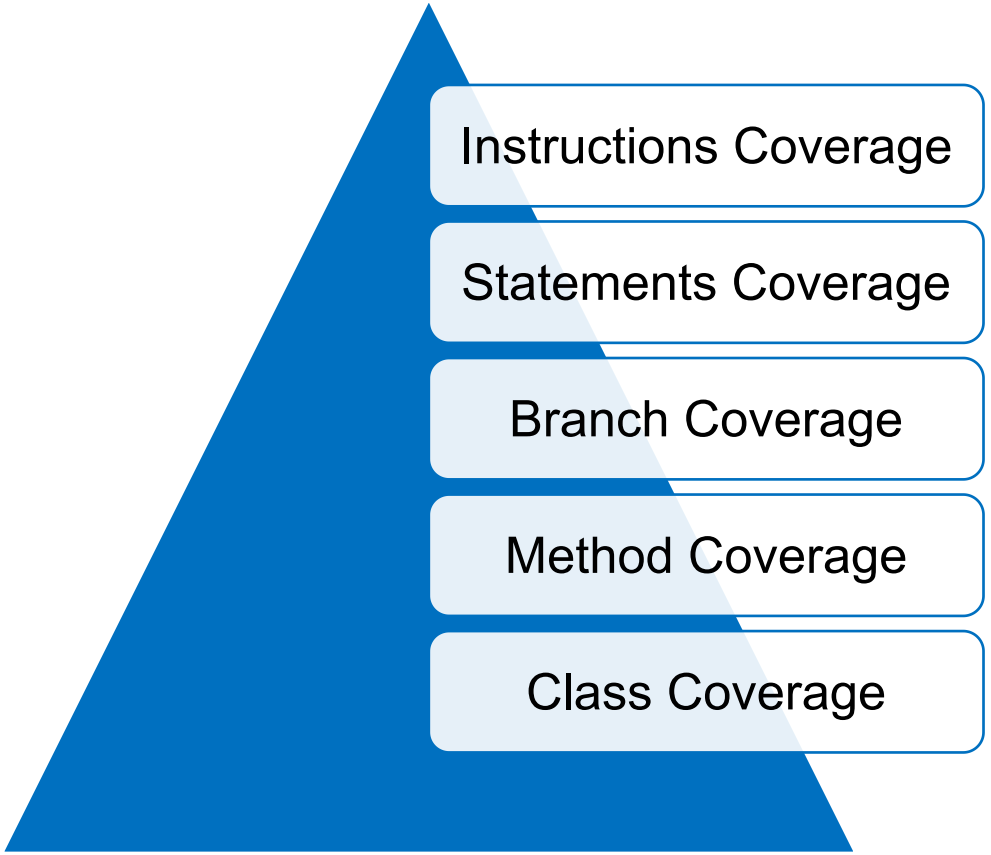
- Code coverage is a measure used to describe the degree to which the source code of a program is executed when a particular **test suite runs** ← A form of dynamic analysis:动态分析
- Code Coverage is classified as a White box testing
  - White Box testing: Testing where internal structure/ design/ implementation of the item being tested is known

Benefits of Code Coverage

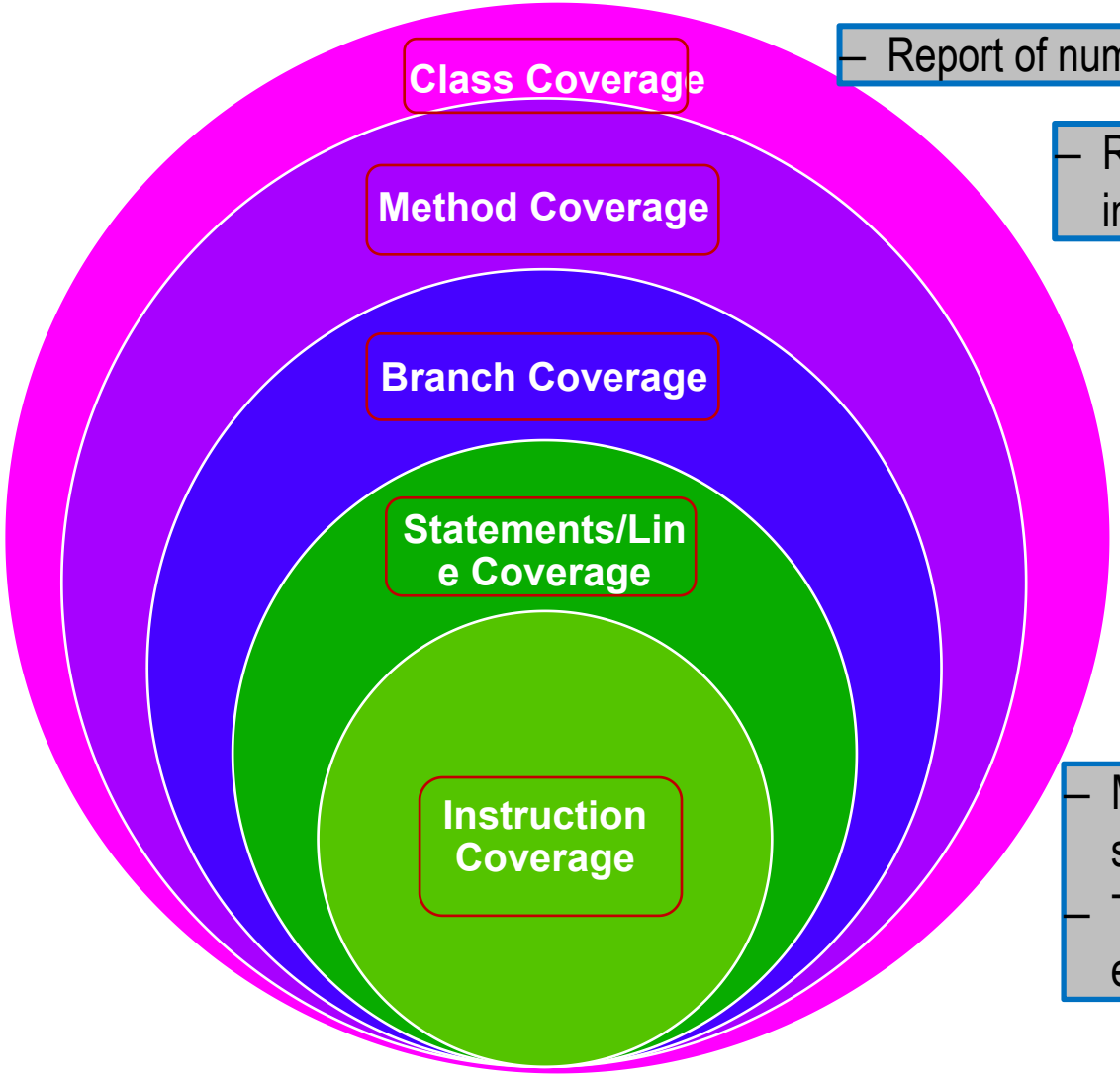
- Identify untested part of codebase
- Improve the Quality by improved test coverage
- Identify testing gaps or missing tests
- Identify the redundant/dead Code

Coverage Criteria

- To measure what percentage of code has been exercised by a test suite, one or more coverage criteria are used

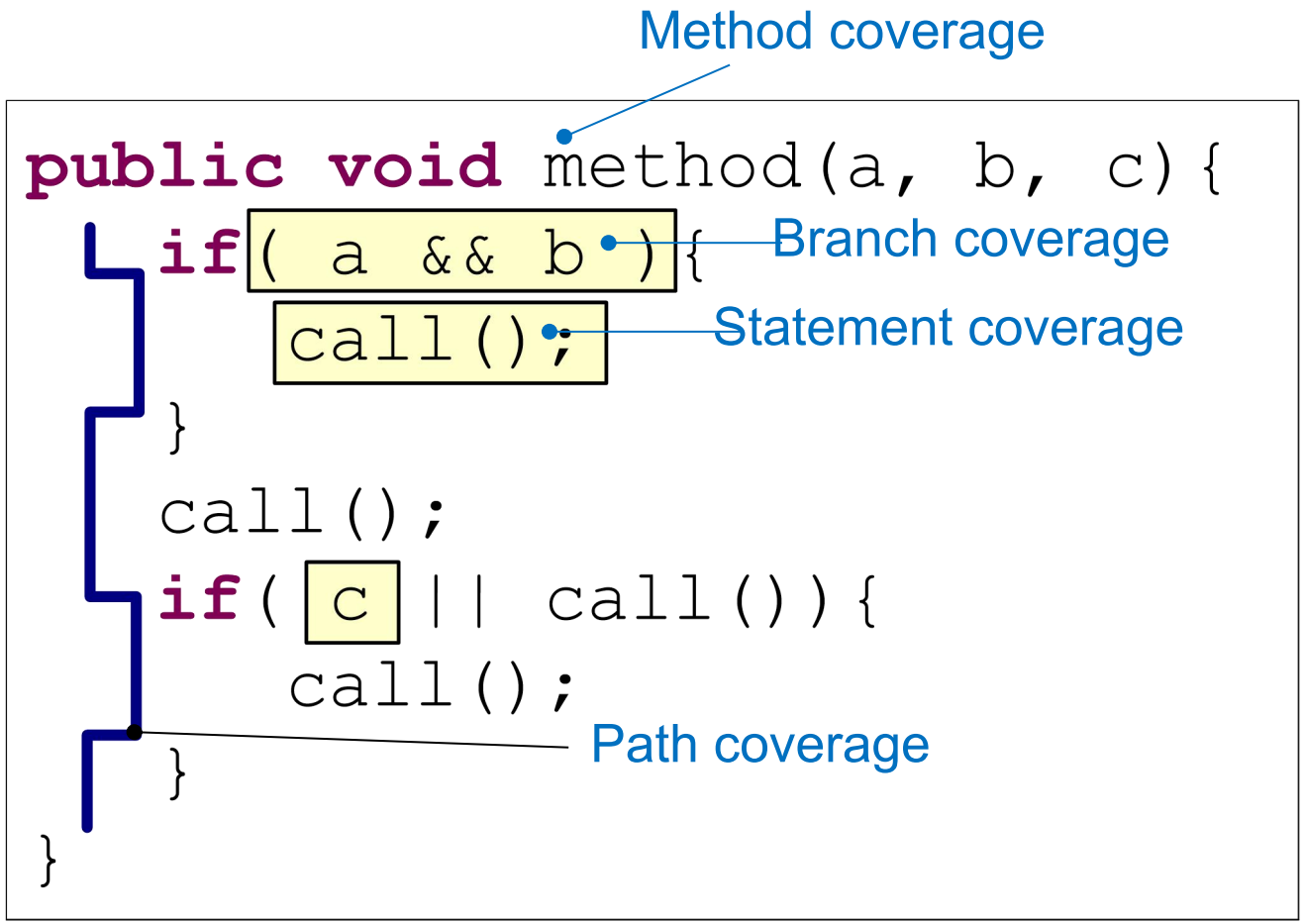


Basic Coverage Criteria



- Report of number of classes from the code base covered
- Reports whether a method (function) was invoked while testing the application
- Reports whether Boolean expressions evaluate to true AND false
- Reports whether each executable statement was executed
- Method's bytecode stream is a sequence of instructions for JVM
- The Bytecode for a method are executed when that method is invoked

Code Coverage



Equation for Computing Coverage

$$\textbf{Statement Coverage} = \frac{Number\ of\ executed\ statements}{Total\ number\ of\ statements} \times 100$$

$$Branch\ Coverage = \frac{Number\ of\ Executed\ Branches}{Total\ Number\ of\ Branches}$$

Code Coverage Analysis Process

Writing test cases and execute them

Finding areas of code not covered using Code Coverage Tool

Creating additional tests for identified gaps to increase test coverage

Determining a quantitative measure of code coverage

Code Coverage using JaCoCo

JaCoCo is an open source code coverage Tool for Java, which has been created by the EclEmma team

- Configure JaCoCo agent with JVM of your system to instrument java classes
- .EXEC file gets generated while the test cases are executed on the system
- Generate Code Coverage report using ant (in different formats)

The agent jacocoagent.jar is part of the JaCoCo distribution and includes all required dependencies. A Java agent can be activated with the following JVM option: -javaagent:*[yourpath/]*jacocoagent.jar=*[option1]=[value1]*,*[option2]=[value2]*

Example-
java -jar-<Jar> -javaagent:<Jacoco location path>/jacocoagent.jar=destfile=<Jacoco location>/jacoco.exec

Code Coverage using JaCoCo

- JaCoCo offers instructions, line, branch, class and package coverage

JaCoCo Ant Example

Element	Missed Instructions	Cov.	Missed Branches	Cov.	Missed	Cxty	Missed	Lines	Missed	Methods	Missed	Classes
com.abc.sss.services		0%		0%	313	313	738	738	90	90	4	4
com.abc.ppp.utils		0%		0%	181	181	488	488	37	37	2	2
com.abc.sss.export		23%		23%	169	214	645	856	33	55	0	2
com.abc.aa.import		63%		50%	294	491	449	1,388	33	131	5	19
com.abc.aa.export		39%		36%	179	239	451	742	34	60	3	5
com.abc.aa.view		16%		10%	186	203	331	409	27	41	5	9
com.abc.bb.code		34%		38%	142	217	332	559	11	61	0	9
com.abc.customize		42%		37%	114	160	278	515	15	34	2	4
com.abc. xx.utilities		37%		28%	141	181	295	489	20	43	2	5
com.abc.yy. search		56%		50%	229	410	322	766	121	265	4	16
com.abc. zz.ptl		40%		30%	95	144	214	374	12	39	0	5
com.abc.xx.result												

com.abc.customize

Element	Missed Instructions	Cov.	Missed Branches	Cov.	Missed	Cxty	Missed	Lines	Missed	Methods	Missed	Classes
FreeFormCust		76%		62%	71	121	62	391	0	18	0	1
WizardCustomize		64%		56%	51	87	65	203	1	9	0	1
VeiwCustomize		34%		10%	32	40	52	96	8	16	0	1
PageObject		64%		50%	31	55	45	161	2	16	0	1
PageCustService		63%		50%	20	32	39	116	0	4	0	1
ResultModelService		80%		63%	24	55	29	167	0	13	0	1
		0%		0%	11	11	47	47	10	10	1	1
		44%		38%	8	14	21	42	2	6	0	1

Other tools for code coverage

- Cobertura
- Atlassian Clover
- DevPartner
- JTest
- Bullseye for C++
- Sonar
- Kalistick

Cobertura

Atlassian Clover

MICRO FOCUS

PARASOFT

sonarsource

Other references

- https://en.wikipedia.org/wiki/Code_coverage
- https://en.wikipedia.org/wiki/White-box_testing
- http://www.eclemma.org/jacoco/
- http://www.jacoco.org/jacoco/trunk/doc/
- https://www.atlassian.com/software/clover

*Learn about how to use JaCoCo in the lab today!

Is Code Coverage a good measurement?

Achieving code coverage

```
@Test
public void add_should_add() {
        new Math().add(1, 1);
}
```

But, where is the assert?

As long as the Code Coverage is OK...

Code coverage as a measure of test quality

EVOSUITE

Any metric can be gamed! Code coverage is a metric…

⇒ Code coverage can be gamed

On purpose Or by accident

Code coverage as a measure of test quality

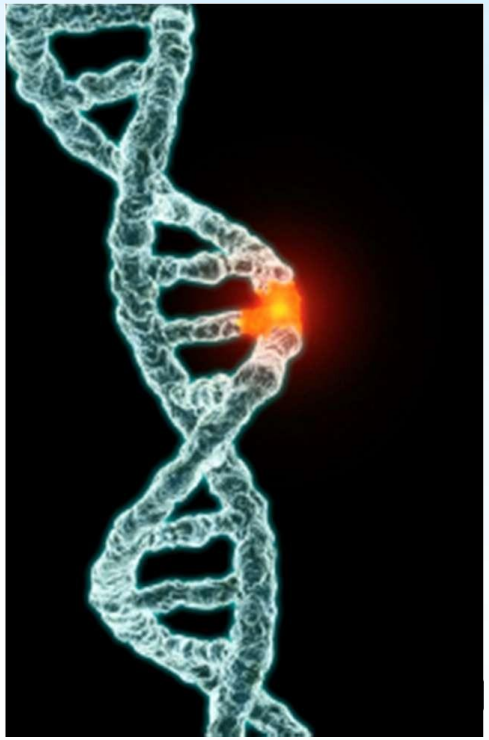
Code Coverage lulls you into a false sense of security (虚假的安全感)…

The problem still remains

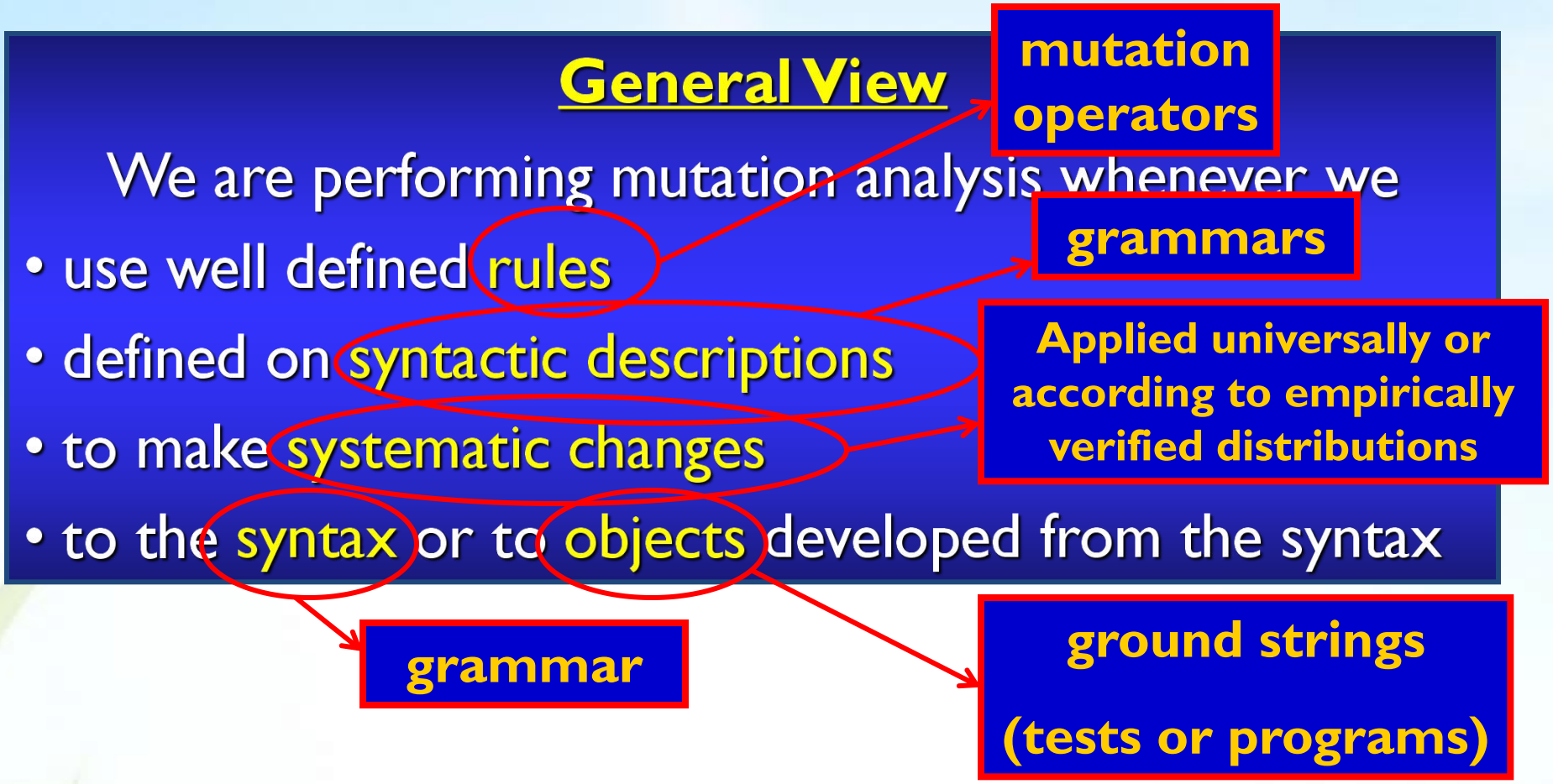
Code coverage cannot ensure test quality

- Is there another way?

Mutation Testing to the rescue!



What is Mutation ?



Why Mutation?

```
public int m1(int i1, int i2) {
    return i1 + i2;
    }
```

```
public int m1(int i1, int i2) {
    return i1 - i2;
    }
```

- What is the changes?
- Which is the correct program?
  - m1(1,0) Output: 1
  - m1(1,2) Output: 3

Why Mutation?

- Mutant processes are created to try to mimic typical syntactic errors made by programmers
- Many differing mutants are run against the specified tests to assess the quality of the tests
- The tests are attributed a score between 0 and 1, as to whether they can distinguish between the original and the mutants

How does it work?
1st Step: Create the Mutant

The Source Code

The "Mutant"

The Mutation "Operator"

Examples

```
DebitCard>>= anotherDebitCard
    ^(type = anotherDebitCard type)
      and: [ number = anotherDebitCard number ]
```

Operator: Change #and: by #or:

```
CreditCard>>= anotherDebitCard
    ^(type = anotherDebitCard type)
      or: [ number = anotherDebitCard number ]
```

How does it work?
2nd Step: Try to Kill the Mutant

The "Mutant"

A Killer
tries to kill the Mutant!

The Test Suite

All tests run → The Mutant Survives!!!

A test fails or errors → The Mutant Dies

Meaning…

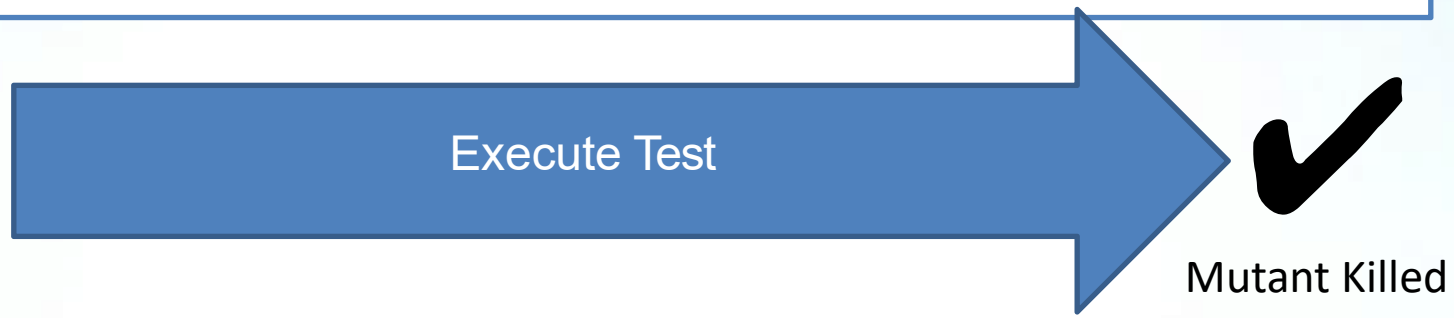
The Mutant Survives → The case generated by the mutant is not tested

The Mutant Dies → The case generated by the mutant is tested

Example: Killing mutants

```
@Test
public void add_should_add() {
    new Math().add(1, 1);
    Assert.assertEquals(sum, 2);

}
```



Mutation Testing in Java

PIT is a tool for Mutation testing

Available as

- Command-line tool

  Ant target
- Maven plugin

Mutators

Mutators are patterns applied to source code to produce mutations

PIT mutators sample

EVOSUITE

Name	Example source	Result
Conditionals Boundary	>	>=
Negate Conditionals	==	!=
Remove Conditionals	foo == bar	true
Math	+	-
Increments	foo++	foo--
Invert Negatives	-foo	foo
Inline Constant	static final FOO= 42	static final FOO = 43
Return Values	return true	return false
Void Method Call	System.out.println("foo")	
Non Void Method Call	long t = System.currentTimeMillis()	long t = 0
Constructor Call	Date d = new Date(d);	Date d = null;

Example of Important mutators

Conditionals Boundary

- Probably a potential serious bug smell

```
if (foo > bar)
```

Is there any tool that helps you increase coverage fast by generating JUnit tests automatically?

- Yes, there are several popular open-source test generations
  - Randoop
  - Evosuite

From:http://www.evosuite.org/evosuite-tutorials/

History

April 9, 2010

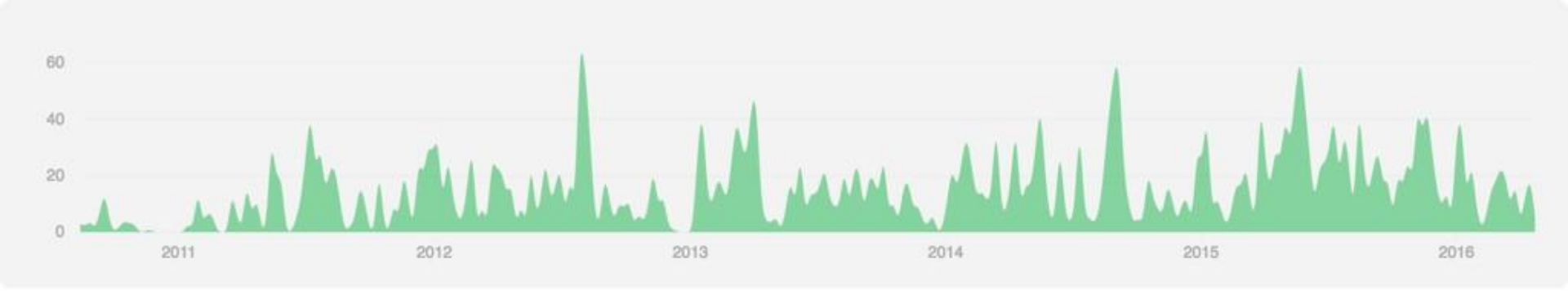
"Evolutionary Generation of Whole Test Suites"

11th International Conference on Software Quality (QSIC 2011)

Who is he?

- **Andrea Arcuri**
- **Prof. Xin Yao's previous students**

Stats of the Evosuite projects on GitHub



- **6,865 commits**
- **229,889 LOC**
- **2,420 tests**

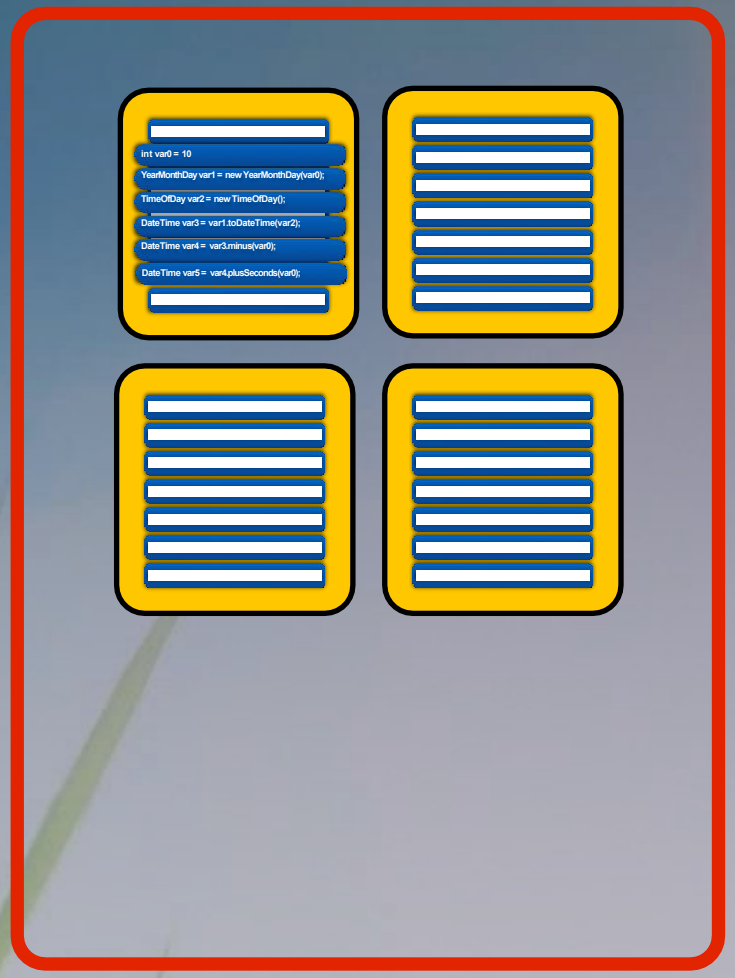
How does it works?

EvoSuite uses evolutionary algorithm to generate and optimize whole test suites towards satisfying a coverage criterion.

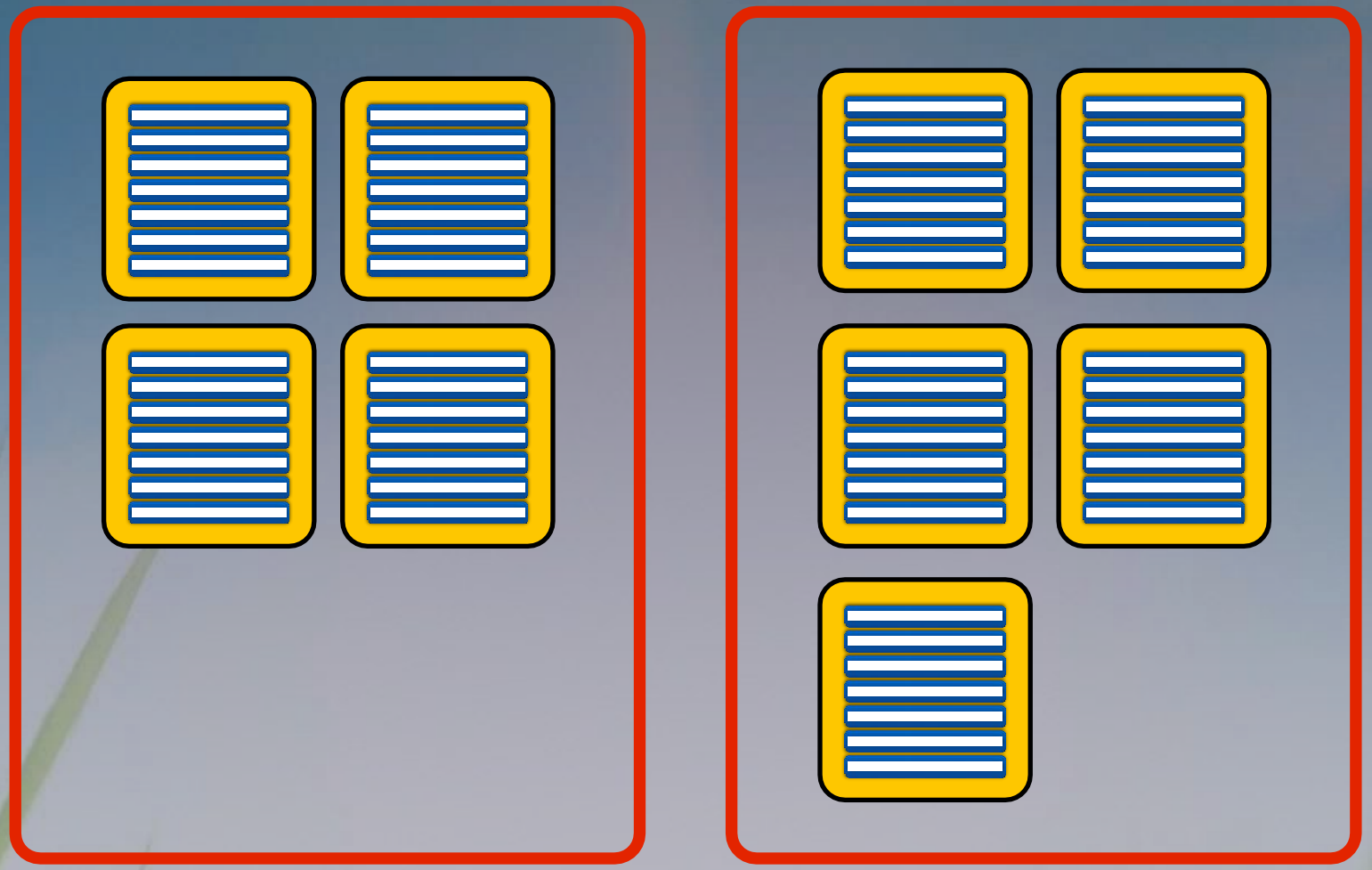
EVOSUITE

```
@Test
public void test()
{
    int var0 = 10
    YearMonthDay var1 = new
    YearMonthDay(var0); TimeOfDay var2 =
    new TimeOfDay(); DateTime var3 =
    var1.toDateTime(var2); DateTime var4 =
    var3.minus(var0);
}
```

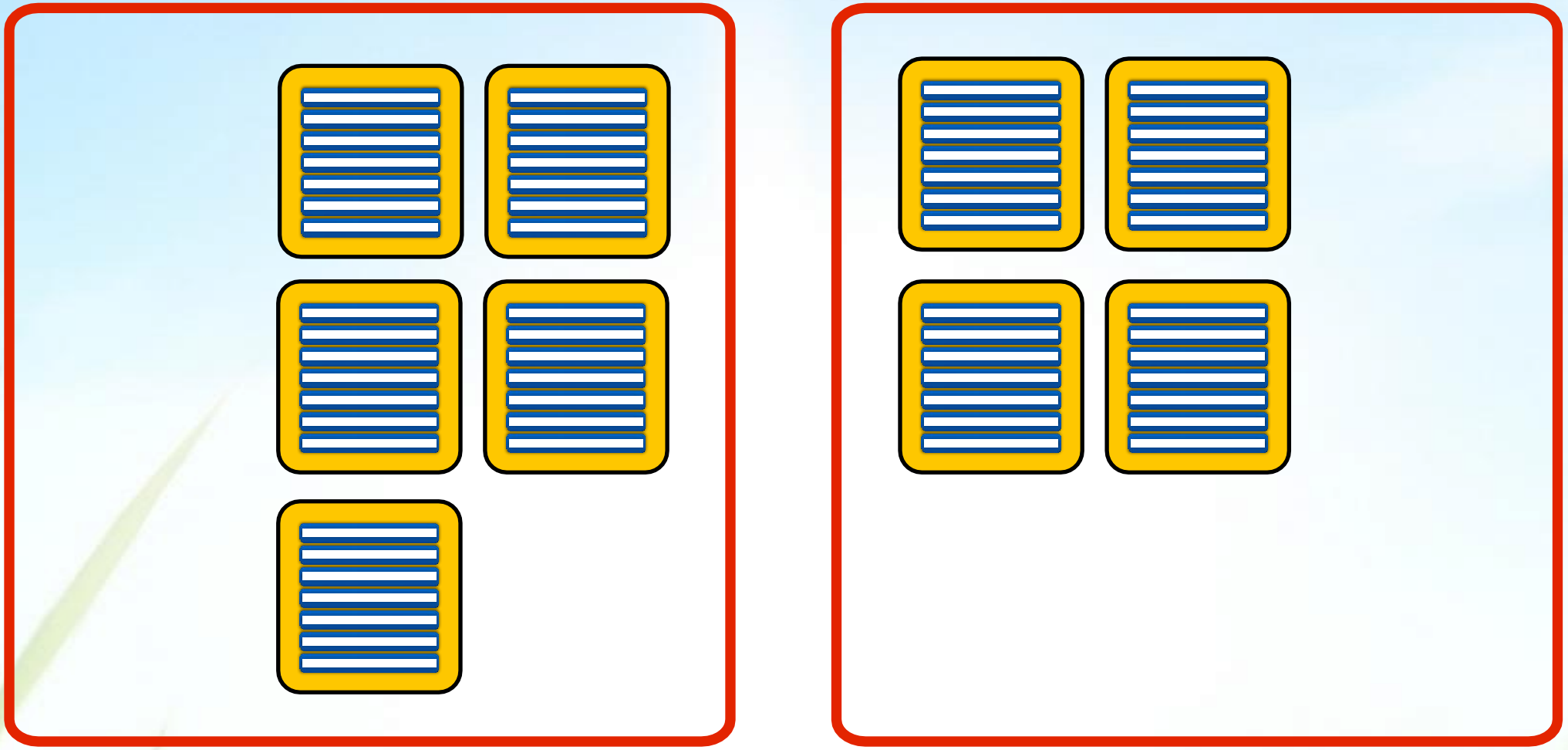
EVOSUITE


int var0 = 10
YearMonthDay var1 = new YearMonthDay(var0);
TimeOfDay var2 = new TimeOfDay();
DateTime var3 = var1.toDateTime(var2);
DateTime var4 = var3.minus(var0);
DateTime var5 = var4.plusSeconds(var0);

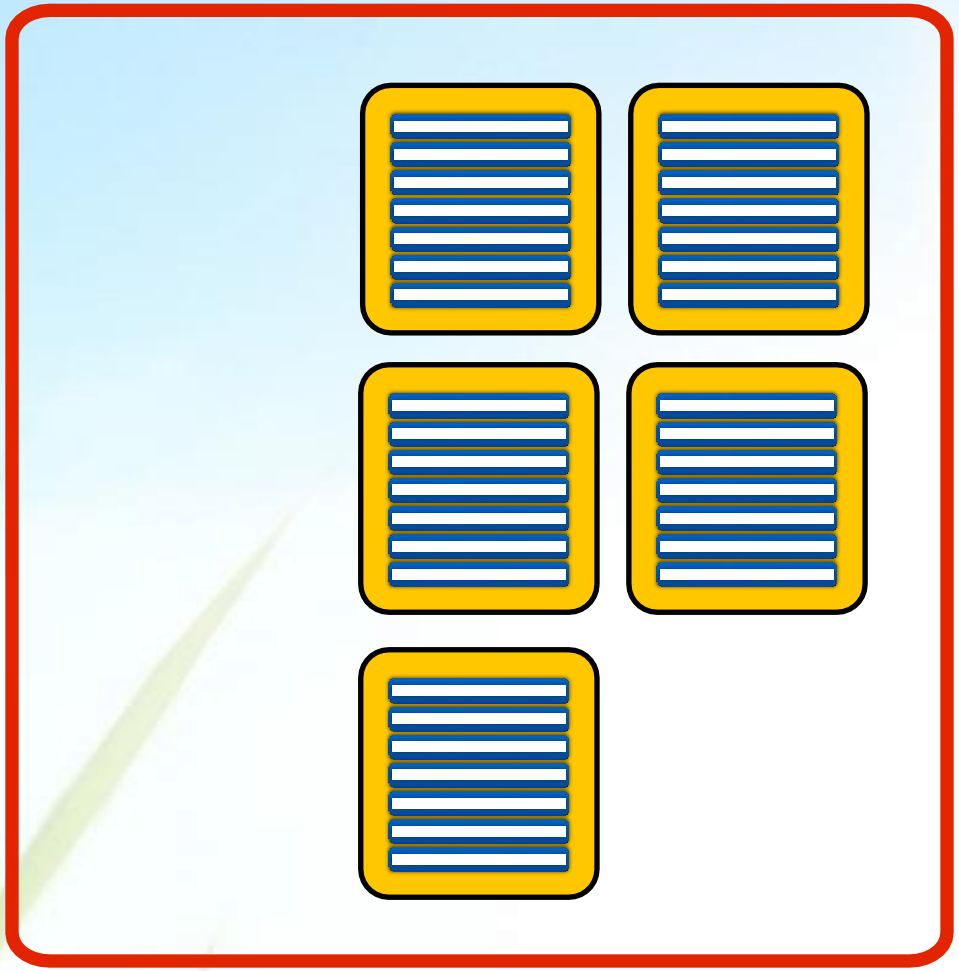
EVOSUITE

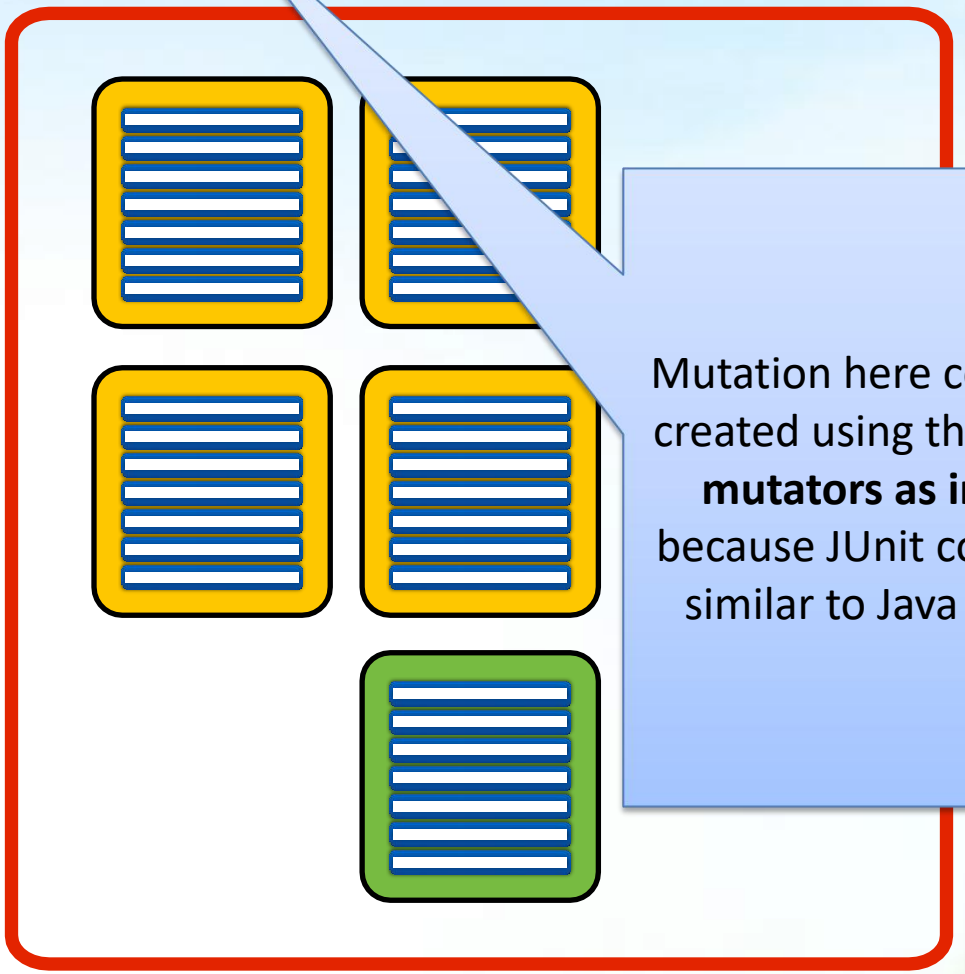


Crossover



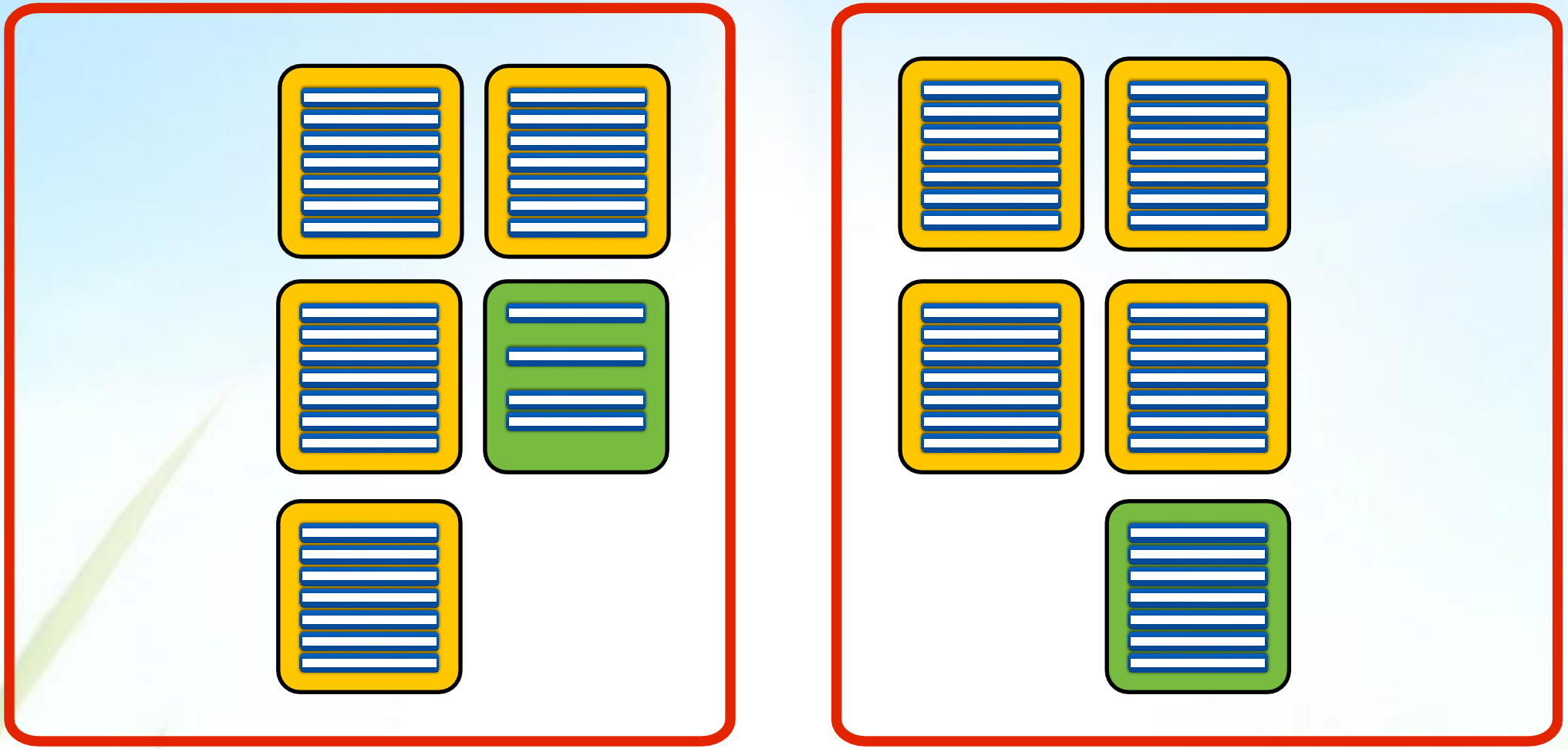
Mutation



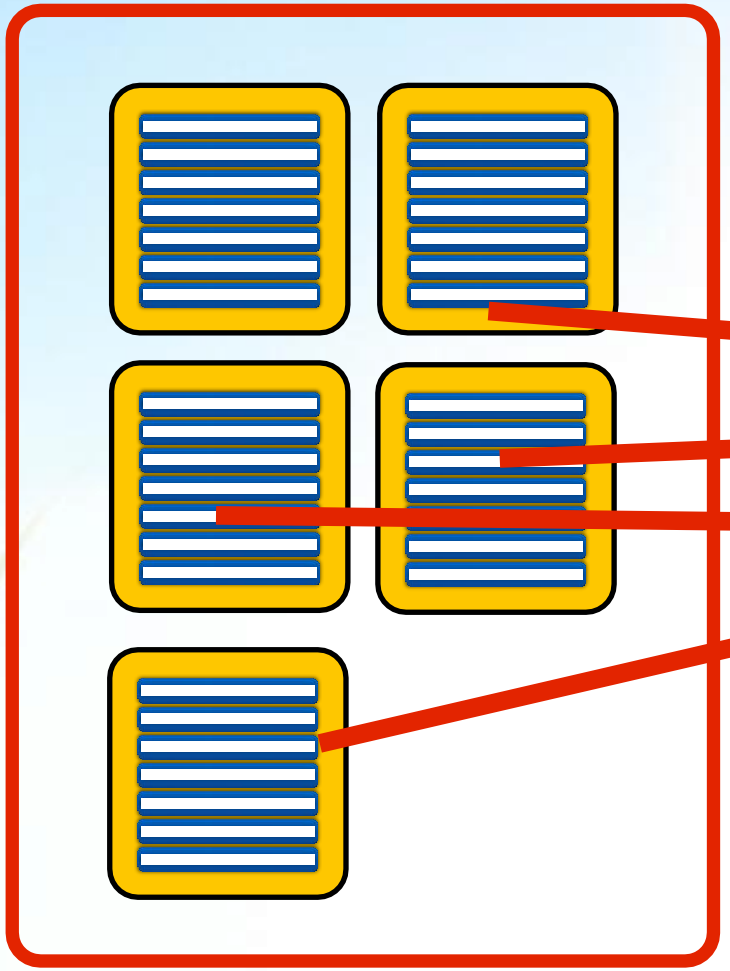


Mutation here could be created using the **same mutators as in PIT** because JUnit code are similar to Java code.

Mutation



Fitness



```
public int gcd(int x, int y) {
   int tmp;
   while (y
      != 0) {
      tmp = x
      % y;  x
      = y;
      y = tmp;
   }
   return x;
}
```

Getting EvoSuite

http://www.evosuite.org/downloads

- Jar release - for command line usage
- Maven plugin
- IntelliJ plugin
- Eclipse plugin
- Jenkins plugin

You will be using Evosuite during the lab this week

When to use and not to use Evosuite?

- **Should I use EvoSuite…**
- **…to test my own Javacode?**
- **Yes, of course**

When to use and not to use Evosuite?

- **Should I use EvoSuite…**
- **…to implement my ideas on unit test generation?**
- **Yes, of course**

When to use and not to use Evosuite?

- **Should I use EvoSuite…**
- **…to study developer behaviour?**
- **Yes, of course**

When to use and not to use Evosuite?

- **Should I use EvoSuite…**
- **…to generate unit tests for my experiment on X?**
- **Yes, of course**

When to use and not to use Evosuite?

- **Should I use EvoSuite…**
- **…to build a unit test generator for a different language ?**
- **Evosuite is 90% JVM Handling code**
- **Would need to reimplement representation, search operators, fitness functions, test execution, …**

When to use and not to use Evosuite?

- **Should I use EvoSuite…**
- **…to create an Android testing tool?**
- **Android uses Java Dalvik bytecode**
- **Can also compile to Java bytecode**
- **How to handle Android dependencies?**

When to use and not to use Evosuite?

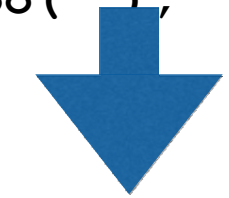
- **Should I use EvoSuite…**
- **…to create a GUI testing tool?**
- **If you want to test Java/Swing applications …**
- **But whole test suite optimisation many not be the right choice**

Research they are working on...

- Increasing coverage…
- Readability optimisation
- Better environment handling
- Mocking and private reflection
- Finding out how developers benefit most from using test generation
- User studies, replications

Method Names

```
@Test(timeout = 4000)
public voi  test3()  throws Throwable
                                {
      StringExample stringExample0 = new
      StringExample();  boolean boolean0 =
      stringExample0.foo("");  assertFalse(boolean0);
}
```

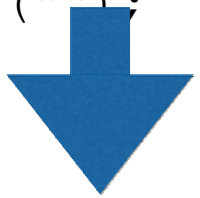


```
@Test(timeout = 4000)
public voi  testFooReturningFalse()  hrows Throwable  {
      StringExample stringExample0 = new
      StringExample();  boolean boolean0 =
      stringExample0.foo("");  assertFalse(boolean0);
}
```

Variable Names

```
@Test(timeout = 4000)
public void testFooReturningFalse()throws Throwable

      {  StringExample stringExample0 = new
      StringExample();  boolean boolean0 =
      stringExample0.foo("");  assertFalse(boolean0);
}
```



```
@Test(timeout = 4000)
public void testFooReturningFalse()throws Throwable

      {  StringExample invokesFoo = new StringExample();
      boolean resultFromFoo = invokesFoo.foo("");
      assertFalse(resultFromFoo);
}
```

Online Tutorials

- Using EvoSuite on the command line:
  http://www.evosuite.org/documentation/tutorial-part-1/
- Using EvoSuite with Maven:
  http://www.evosuite.org/documentation/tutorial-part-2/
- Running experiments with EvoSuite:
  http://www.evosuite.org/documentation/tutorial-part-3/
- Extending EvoSuite:
  http://www.evosuite.org/documentation/tutorial-part-4/