Ch2 Code Unit Testing

Write Code to Test Code(1)

JUnit 5 mockito JaCoCo

Instructor: Haiying SUN

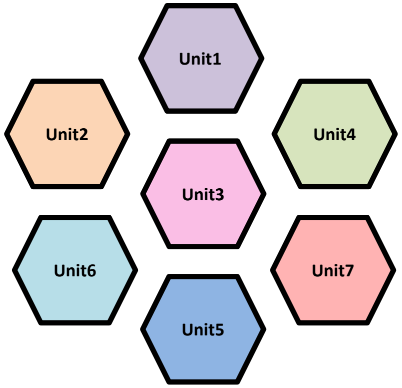
E-mail: hysun@sei.ecnu.edu.cn

Office: ECNU Science Build B1104

Available Time: Wednesday 8:00 -12:00 a.m.

Dynamic Code Test

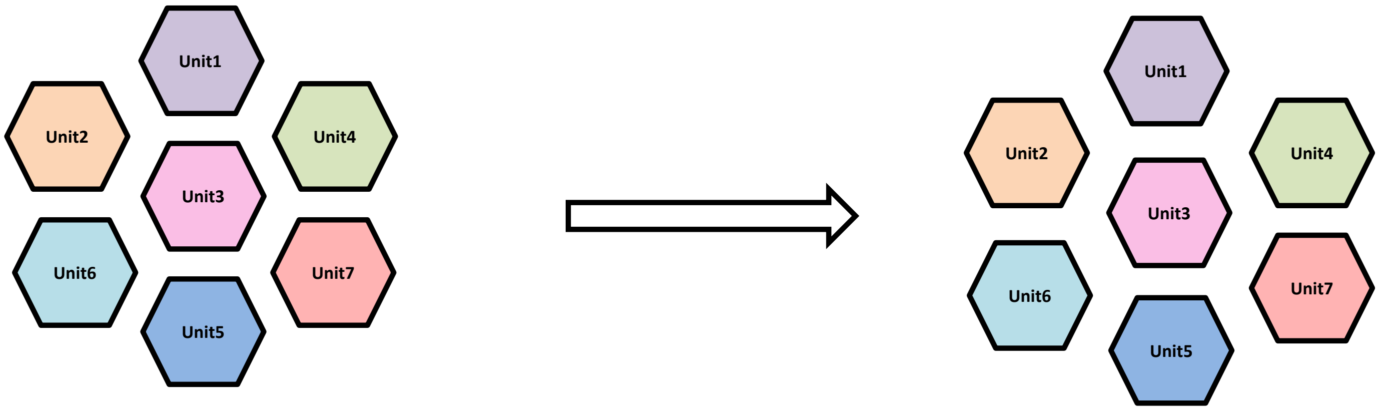
- 在开发环境中，通过运行被测代码以验证其是否满足期望目标而进行的一种测试活动以尽早尽可能发现与目标不一致的缺陷



Unit Test

Dynamic Code Test

- 在开发环境中，通过运行被测代码以验证其是否满足期望目标而进行的一种测试活动以尽早尽可能发现与目标不一致的缺陷



Unit Test

Integration Test

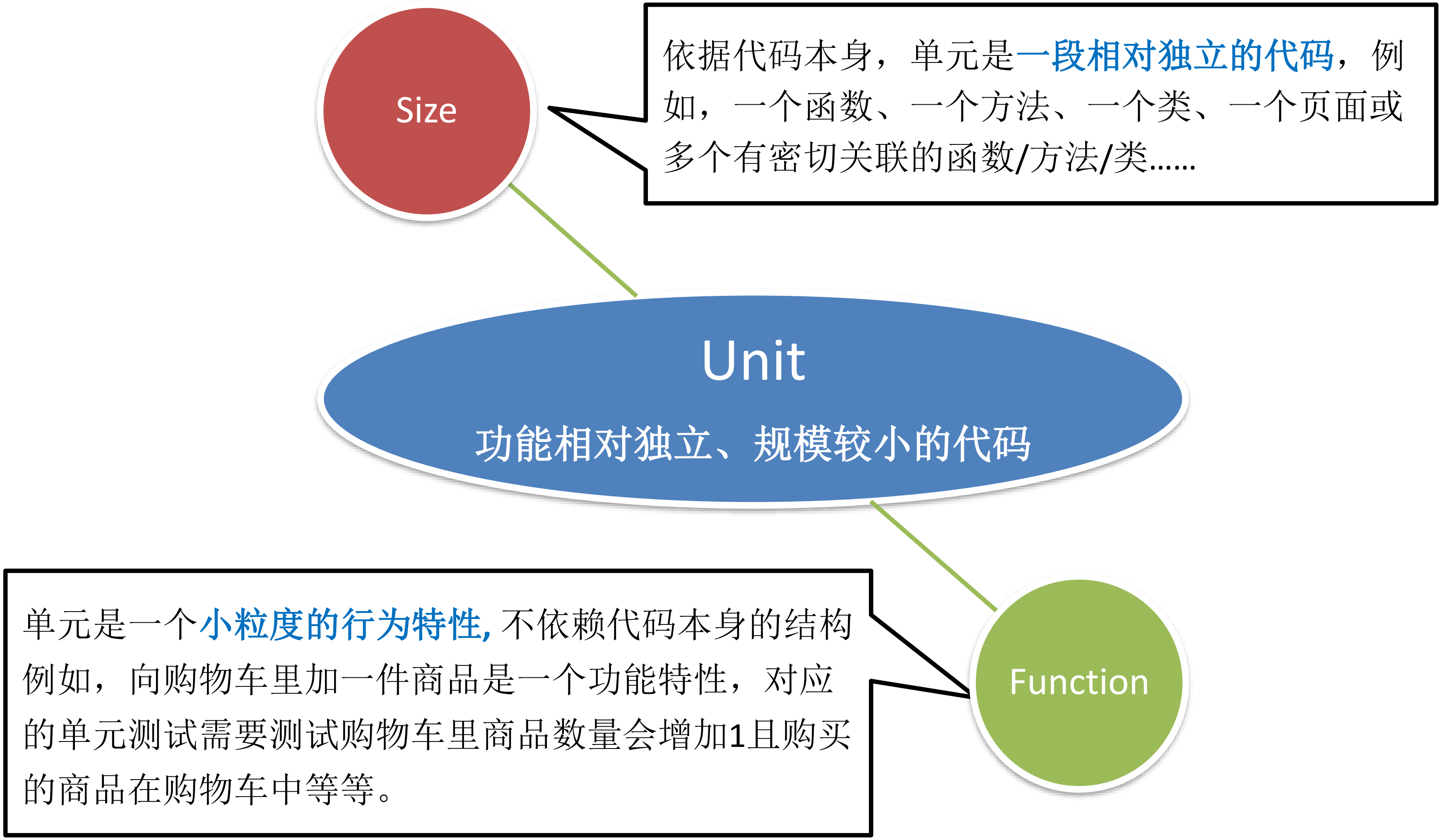
Unit Test vs Integration Test

测试执行是否经济快捷界定单元测试，运行要**快,** **不超过0.1秒**
不是单元测试：

① 跟数据库交互
② 进行了网络间通信
③ 调用了文件系统
④ 需要对环境做特定的准备（如编辑配置文件）才能运行起来

测试替身（Test Double）
替代真实代码中依赖于数据库、网络和文件系统的代码

What is a Unit



Size

依据代码本身，单元是**一段相对独立的代码**，例如，一个函数、一个方法、一个类、一个页面或多个有密切关联的函数/方法/类......

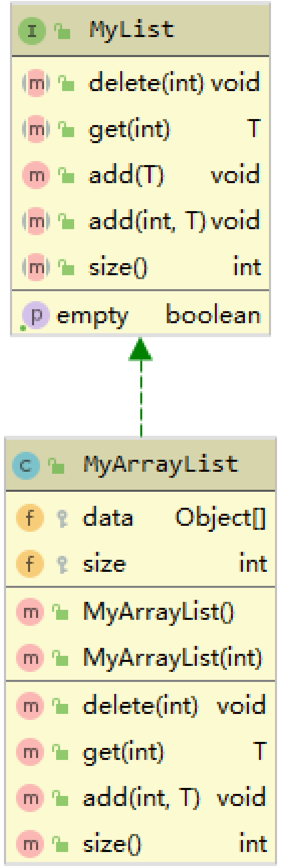
Unit

功能相对独立、规模较小的代码

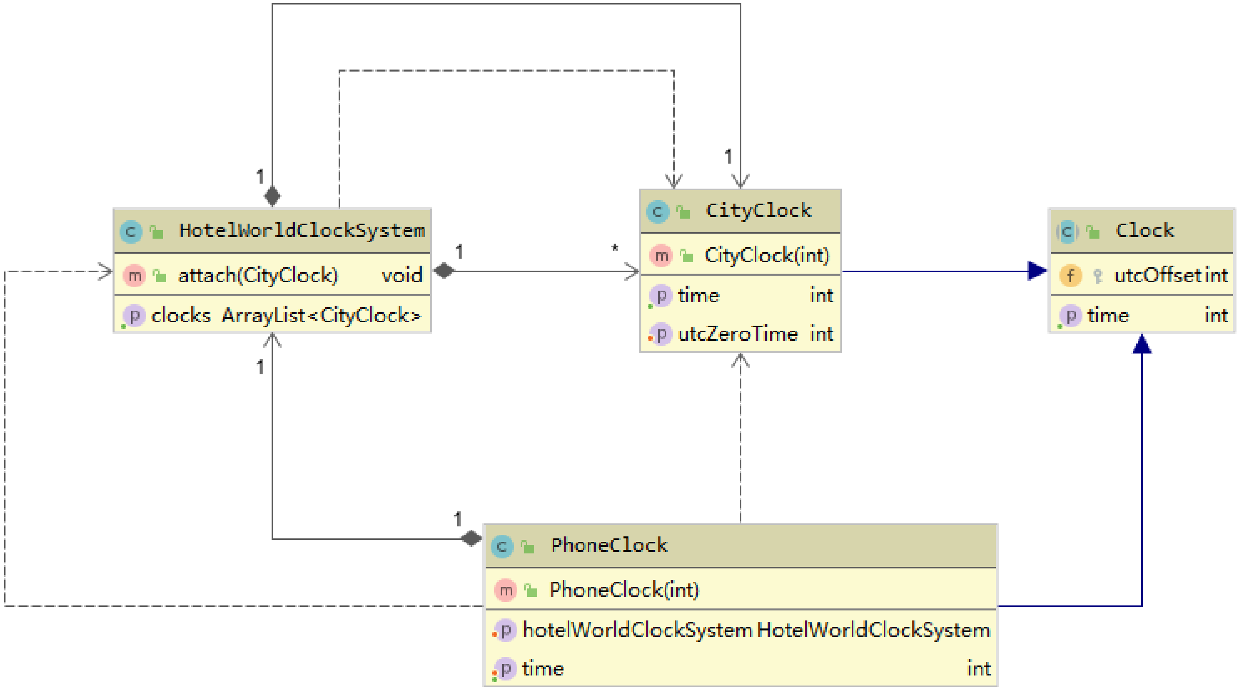
Function

单元是一个**小粒度的行为特性,** 不依赖代码本身的结构例如，向购物车里加一件商品是一个功能特性，对应的单元测试需要测试购物车里商品数量会增加1且购买的商品在购物车中等等。

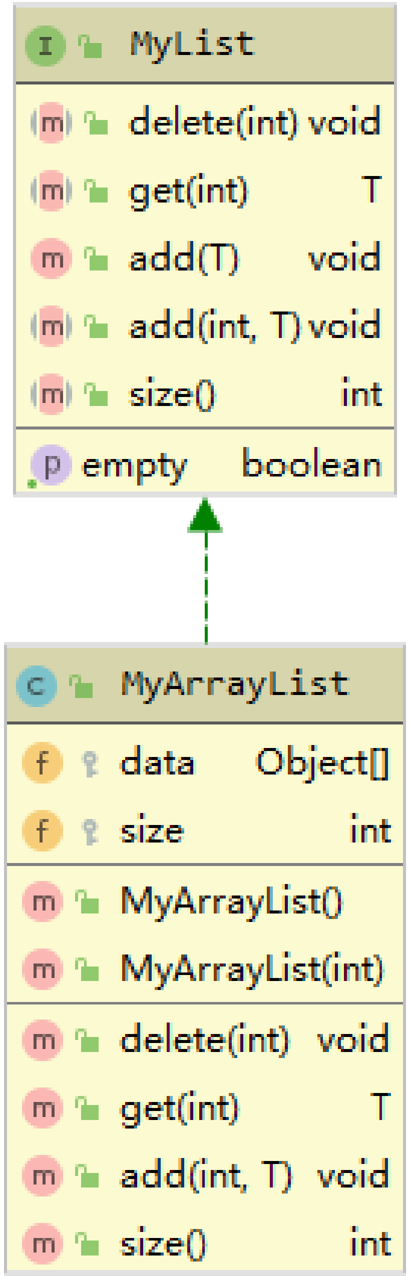
Example



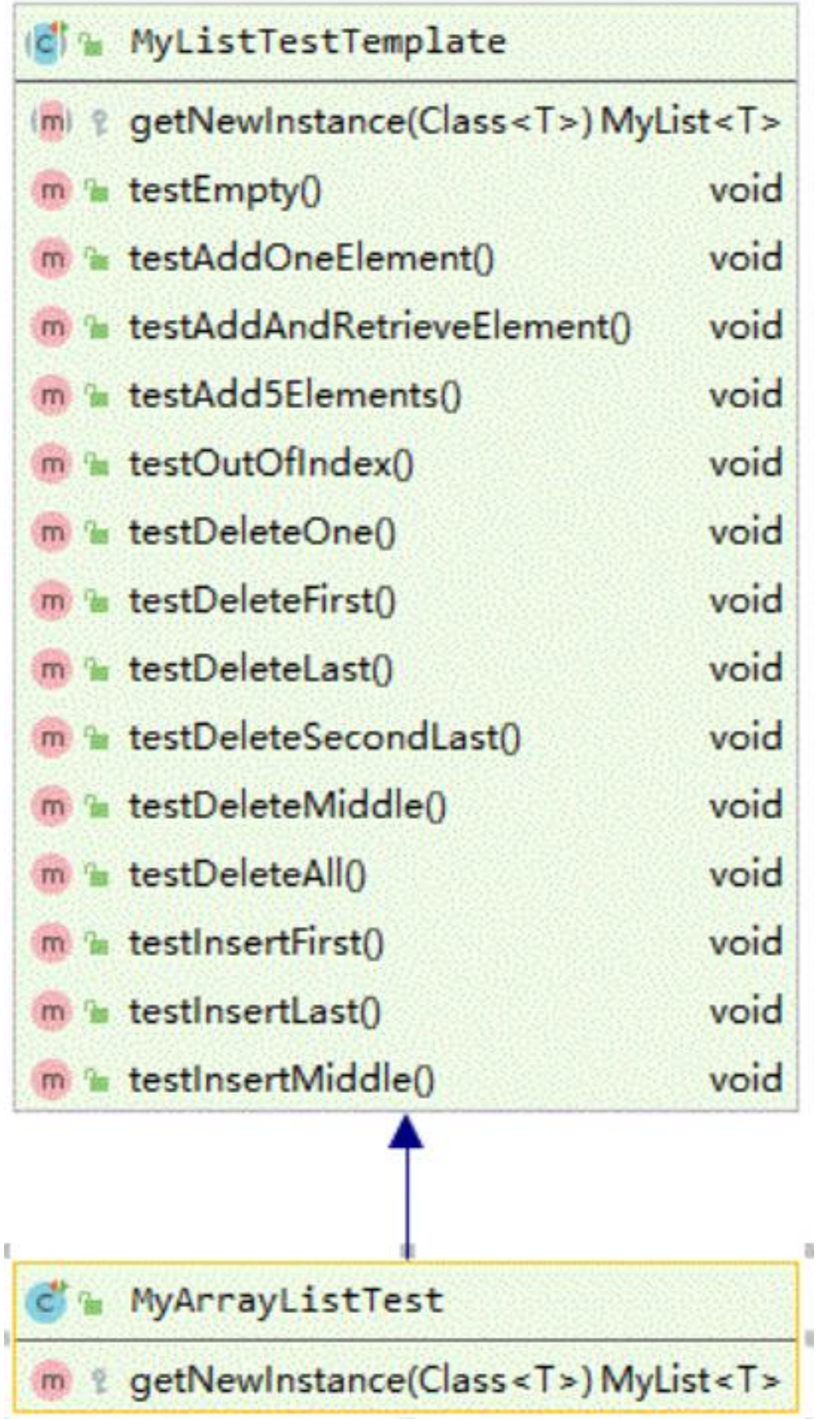
ArrayList

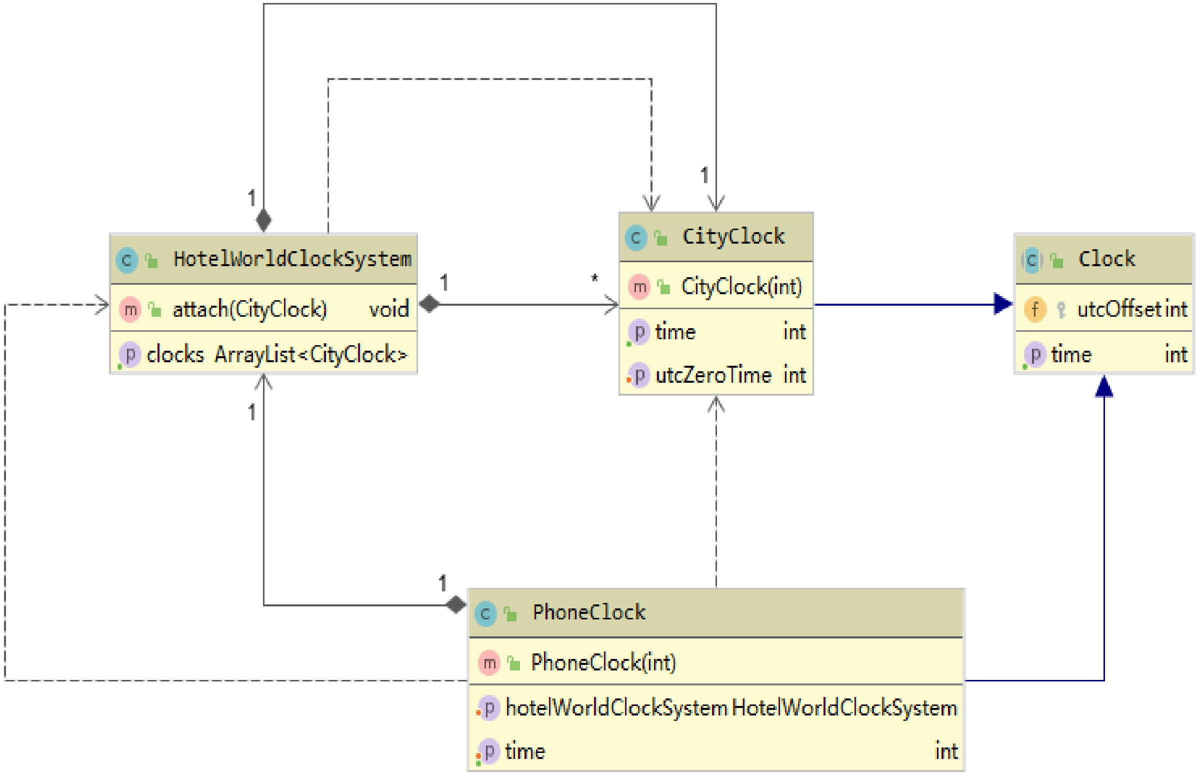


酒店时钟系统：根据手机时间调整酒店前台上展示的世界各地时钟的时间

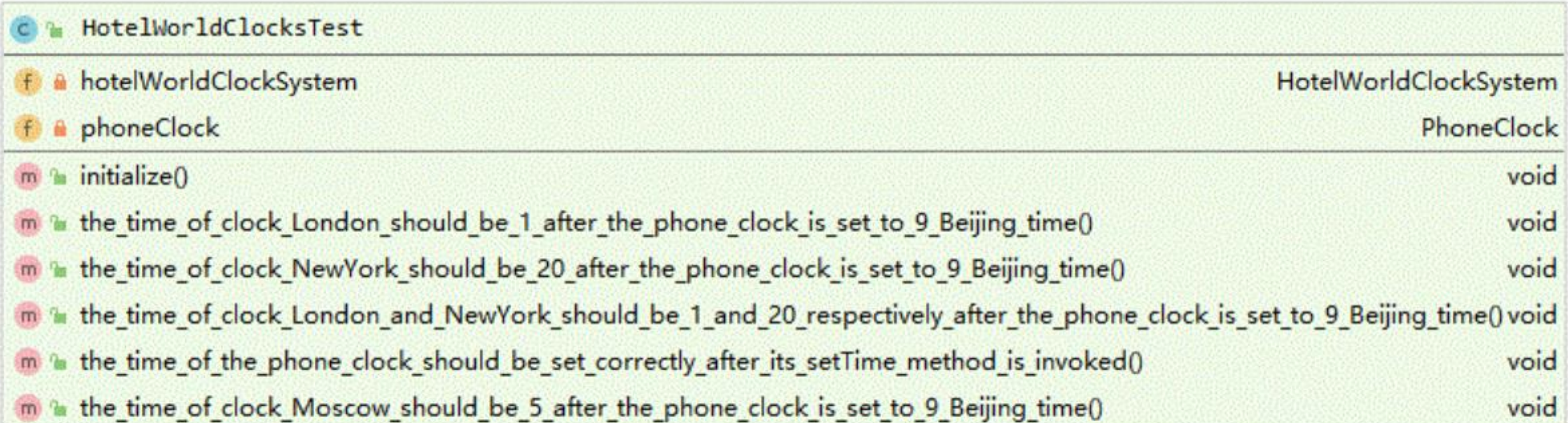


Unit Test Code





Unit Test Code



```
HotelWorldClocksTest
hotelWorldClockSystem                                                                   HotelWorldClockSystem
phoneClock                                                                              PhoneClock
initialize()                                                                            void
the_time_of_clock_London_should_be_1_after_the_phone_clock_is_set_to_9_Beijing_time()   void
the_time_of_clock_NewYork_should_be_20_after_the_phone_clock_is_set_to_9_Beijing_time() void
the_time_of_clock_London_and_NewYork_should_be_1_and_20_respectively_after_the_phone_clock_is_set_to_9_Beijing_time() void
the_time_of_the_phone_clock_should_be_set_correctly_after_its_setTime_method_is_invoked() void
the_time_of_clock_Moscow_should_be_5_after_the_phone_clock_is_set_to_9_Beijing_time()   void
```

Benefits of Code Testing

Quick feedback

Automated Regression Checking

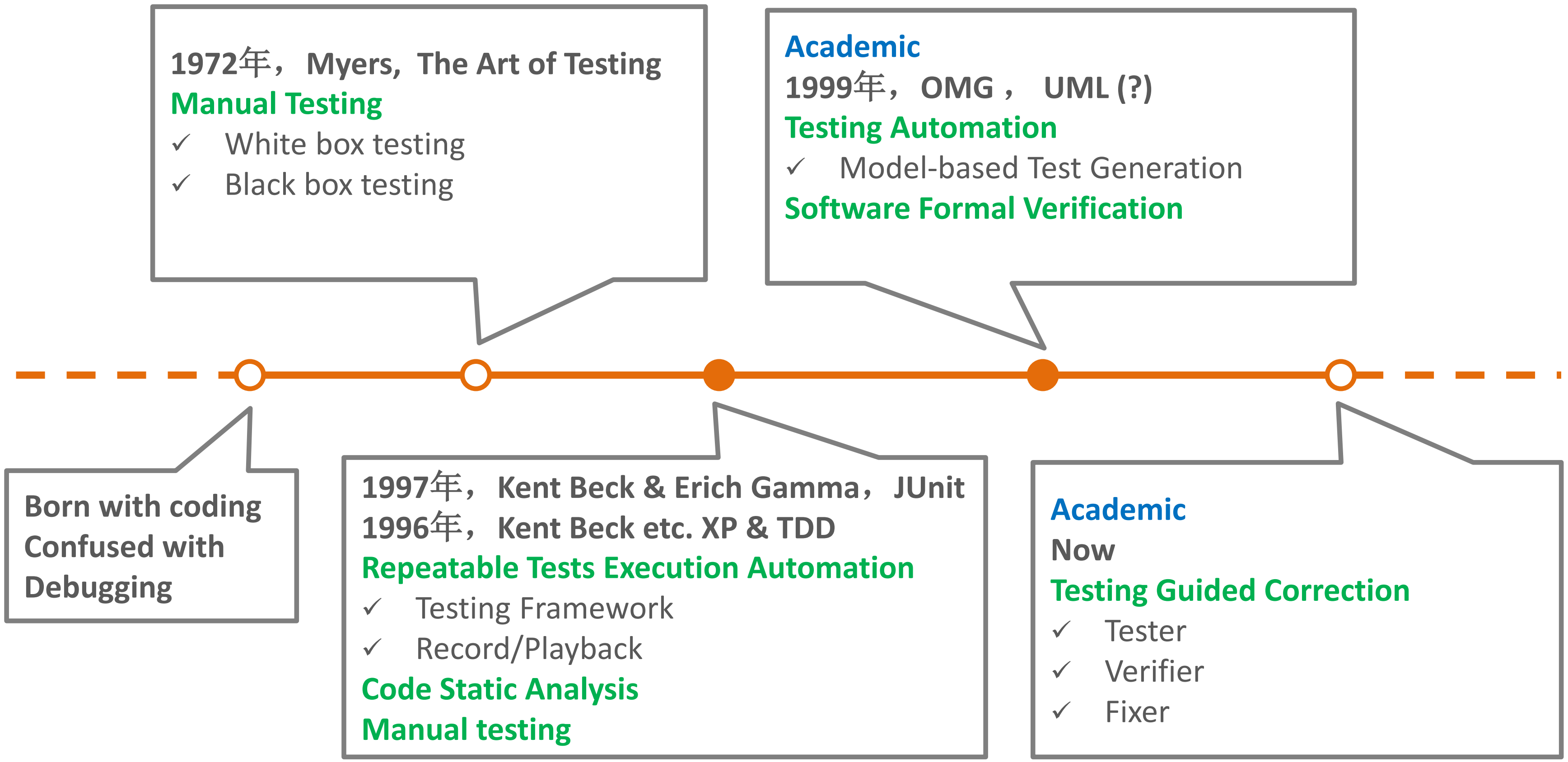
Design Aid

Improve confidence

Document

Documentation

Testing Methodology Timeline



Born with coding
Confused with
Debugging

1972年， Myers，The Art of Testing
Manual Testing
- ✓ White box testing
- ✓ Black box testing

1997年， Kent Beck & Erich Gamma， JUnit
1996年， Kent Beck etc. XP & TDD
Repeatable Tests Execution Automation
- ✓ Testing Framework
- ✓ Record/Playback

Code Static Analysis
Manual testing

Academic
1999年， OMG ， UML (?)
Testing Automation
- ✓ Model-based Test Generation

Software Formal Verification

Academic
Now
Testing Guided Correction
- ✓ Tester
- ✓ Verifier
- ✓ Fixer

Agenda

- Code Test Techniques
  - Logical Testing & Tools
  - Heuristic Rules
  - Junit & Qualified test scripts
- Code Test Generation
  - Control flow based
  - Data flow based
  - Mutation Based
  - Test Automation Tool Development

Agenda

- Code Test Techniques
  - Logical Testing & Tools
  - Heuristic Rules
  - Junit & Qualified test scripts
- Code Test Generation
  - Control flow based
  - Data flow based
  - Mutation Based
  - Test Automation Tool Development

Logical Testing & Tools

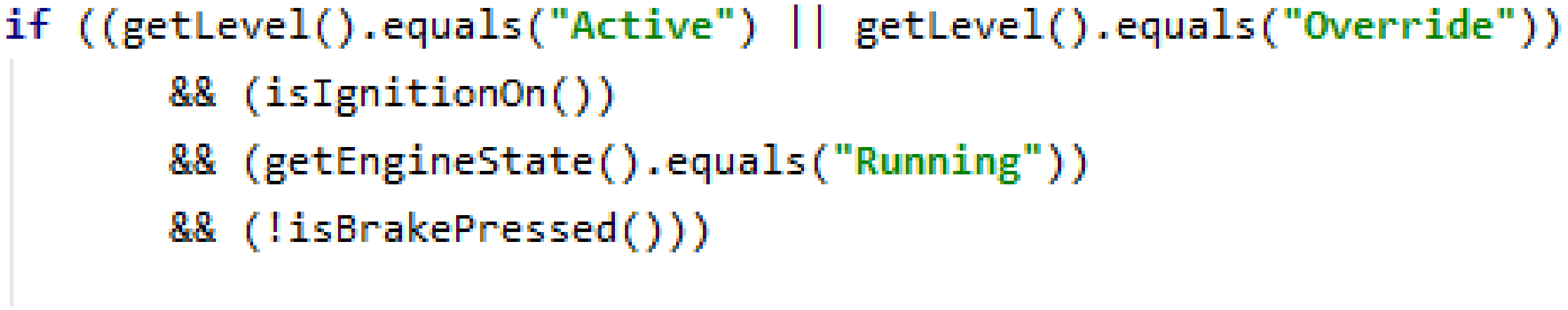
- Logical Coverage Criteria
  - Statement Coverage
  - Decision Coverage
  - Condition Coverage
  - Decision-Condition Coverage
  - Modified Decision-Condition Coverage
  - Multiple Condition Coverage
- Logical Coverage Criteria Tools

Logical Testing

- 逻辑表达式是实现代码特性的核心成份
- 逻辑测试
  - 以代码中逻辑表达式结构为对象的测试，以期发现代码逻辑结构缺陷（不是所有的缺陷类型都可以发现）
  - 逻辑结构缺陷
    1. 写代码时所犯错误在逻辑表达式上的可视化体现
    2. 逻辑表达式写错了，程序行为不正确
- 逻辑测试技术
  - 基于逻辑覆盖准则的测试（Logical Coverage Criteria）
  - 满足逻辑覆盖准则 ≠ 高质量测试

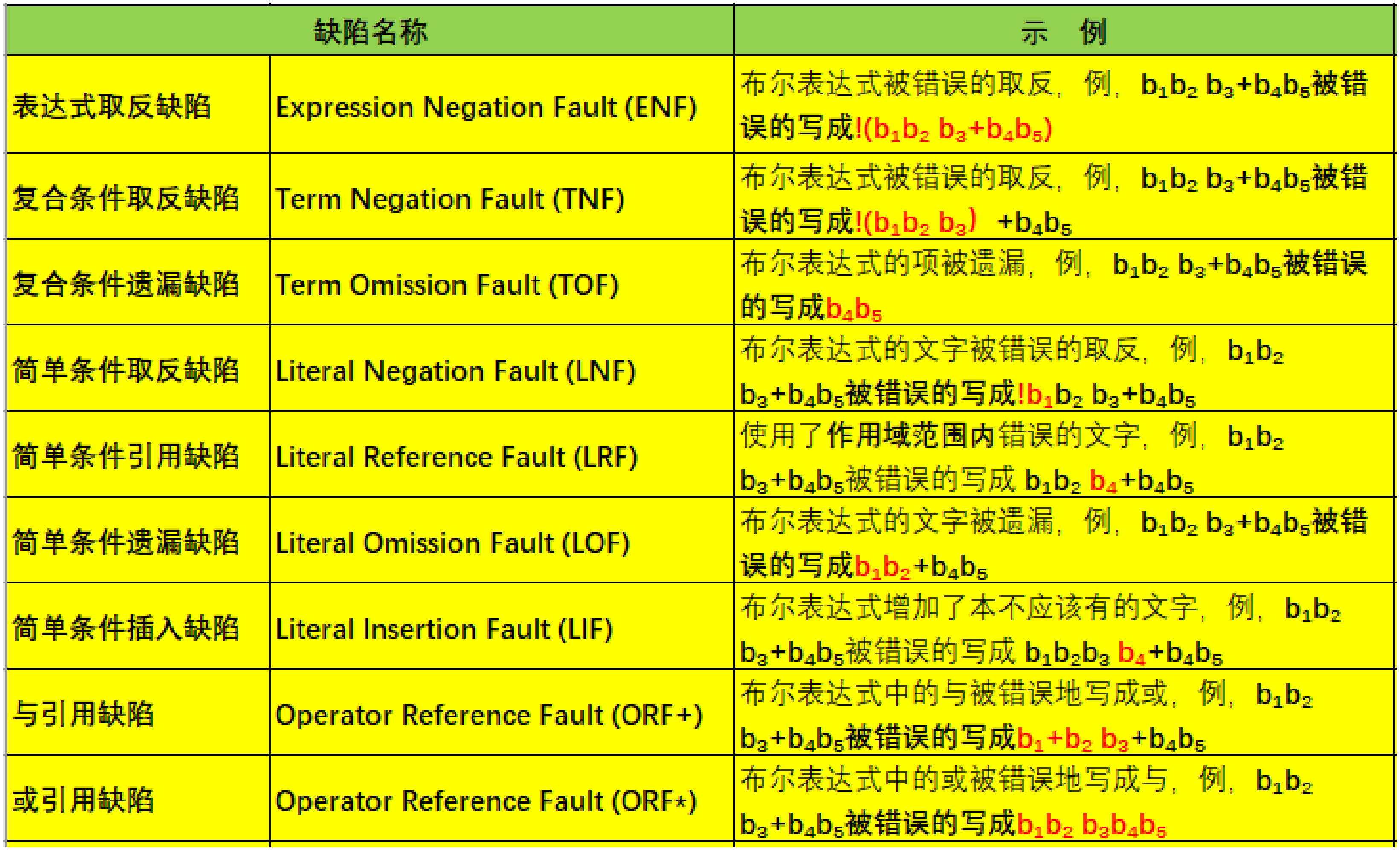
Logical Expression

- [**Specification**]: When the cruise control level is set at Activate or Override, meanwhile the automobile's ignition is on, the engine is running, the current car speed is more than 40km/h but less than 70km/h and the brake pedal is not being pressed, the cruise function begins to work.
- [**Implementation**]:



```
if ((getLevel().equals("Active") || getLevel().equals("Override"))
        && (isIgnitionOn())
        && (getEngineState().equals("Running"))
        && (!isBrakePressed()))
```

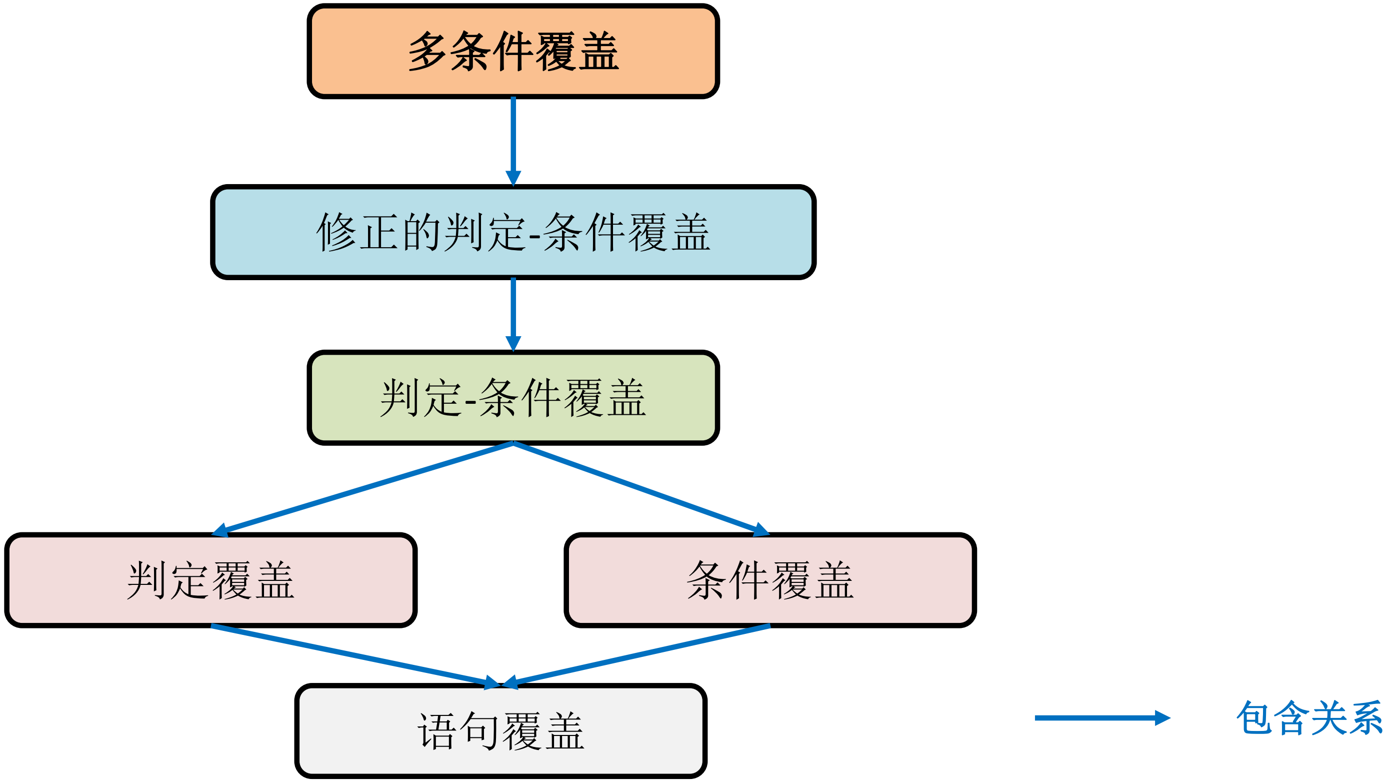
逻辑表达式缺陷类型(DNF)



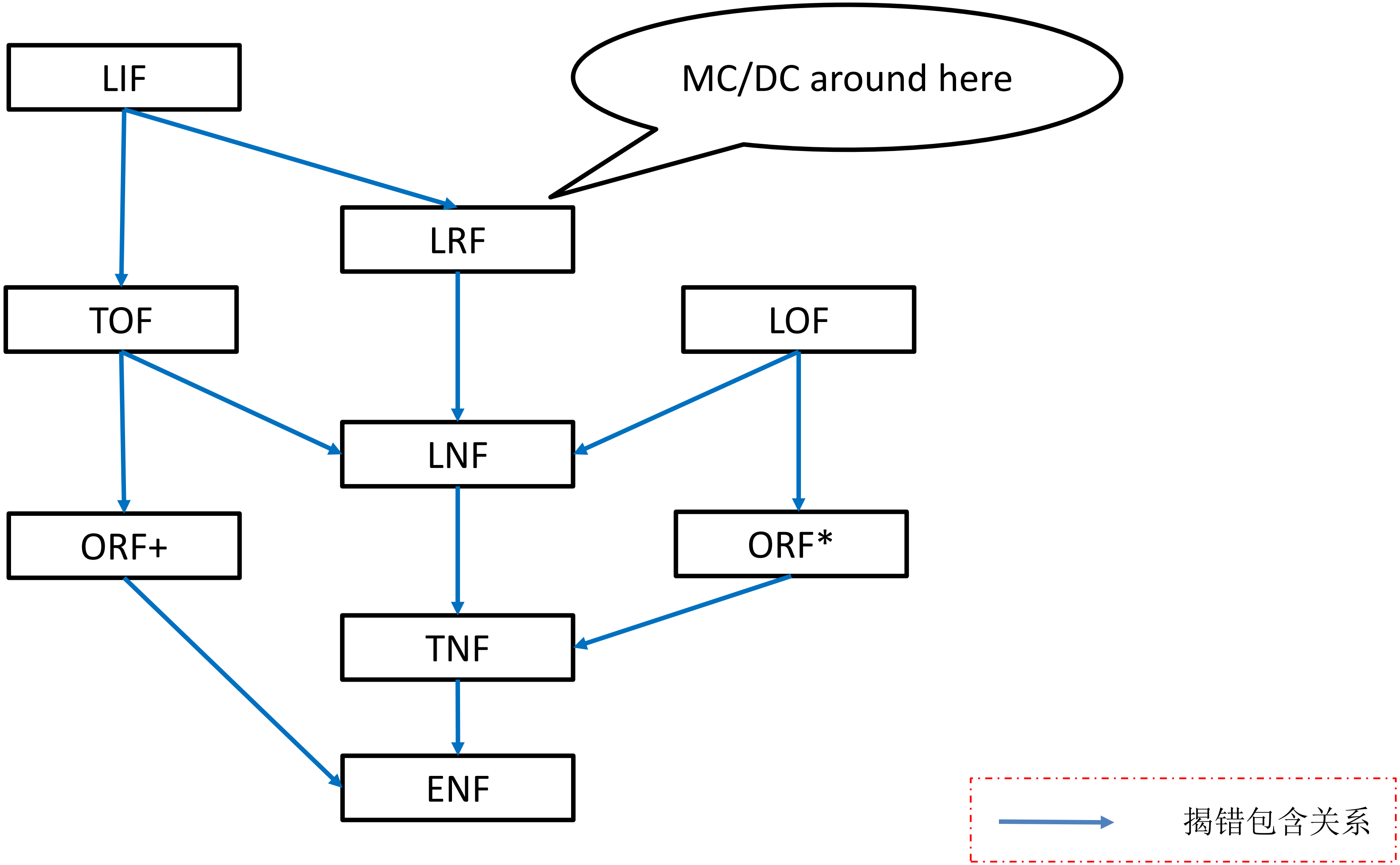
缺陷名称		示　例
表达式取反缺陷	Expression Negation Fault (ENF)	布尔表达式被错误的取反，例，$b_1b_2\ b_3+b_4b_5$被错误的写成!($b_1b_2\ b_3+b_4b_5$)
复合条件取反缺陷	Term Negation Fault (TNF)	布尔表达式被错误的取反，例，$b_1b_2\ b_3+b_4b_5$被错误的写成!($b_1b_2\ b_3$） $+b_4b_5$
复合条件遗漏缺陷	Term Omission Fault (TOF)	布尔表达式的项被遗漏，例，$b_1b_2\ b_3+b_4b_5$被错误的写成b_4b_5
简单条件取反缺陷	Literal Negation Fault (LNF)	布尔表达式的文字被错误的取反，例，$b_1b_2\ b_3+b_4b_5$被错误的写成!$b_1b_2\ b_3+b_4b_5$
简单条件引用缺陷	Literal Reference Fault (LRF)	使用了作用域范围内错误的文字，例，$b_1b_2\ b_3+b_4b_5$被错误的写成 $b_1b_2\ b_4+b_4b_5$
简单条件遗漏缺陷	Literal Omission Fault (LOF)	布尔表达式的文字被遗漏，例，$b_1b_2\ b_3+b_4b_5$被错误的写成$b_1b_2+b_4b_5$
简单条件插入缺陷	Literal Insertion Fault (LIF)	布尔表达式增加了本不应该有的文字，例，$b_1b_2\ b_3+b_4b_5$被错误的写成 $b_1b_2b_3\ b_4+b_4b_5$
与引用缺陷	Operator Reference Fault (ORF+)	布尔表达式中的与被错误地写成或，例，$b_1b_2\ b_3+b_4b_5$被错误的写成$b_1+b_2\ b_3+b_4b_5$
或引用缺陷	Operator Reference Fault (ORF*)	布尔表达式中的或被错误地写成与，例，$b_1b_2\ b_3+b_4b_5$被错误的写成$b_1b_2\ b_3b_4b_5$

Logical Coverage Criteria

- 用于衡量代码中逻辑表达式被测试的充分程度
- A包含B (A subsume B)：B能够发现的缺陷一定可以被A发现



逻辑测试揭错能力



Statement Coverage

- 语句覆盖（Statement Coverage）
  - 衡量被测代码中的语句得到执行的程度。
  - 如果测试集合能够使得被测代码中的每条语句至少被执行一次，那么则说该测试集合满足了语句覆盖。
- 语句覆盖度

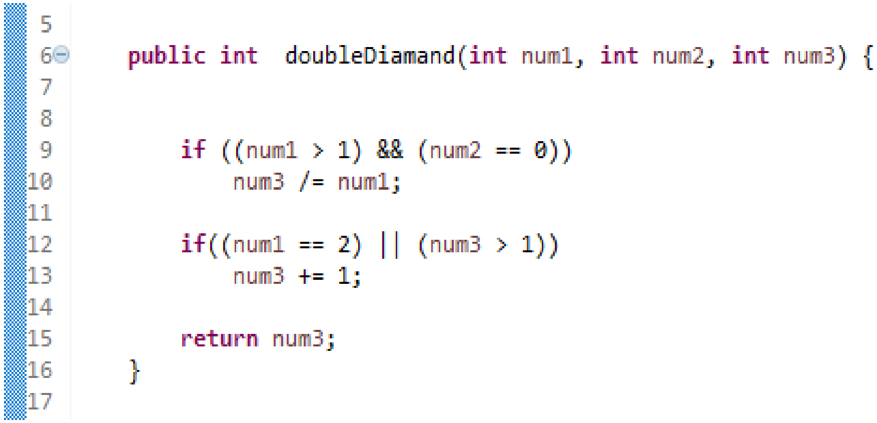
$$\text{语句覆盖度} = \frac{\text{得到执行的语句数}}{\text{语句总数}} * 100\%$$

Statement Coverage

- 测试集合1
  - ① 测试用例1

    [(num1=2, num2=0, num3=4) , 3]
  - 语句覆盖度 = 3/3 = 100%, 满足语句覆盖
- 测试集合2
  - ① 测试用例1

    [(num1=-2, num2=0, num3=2), 3]
  - 语句覆盖度 = 2/3 = 66.7%, 不满足语句覆盖

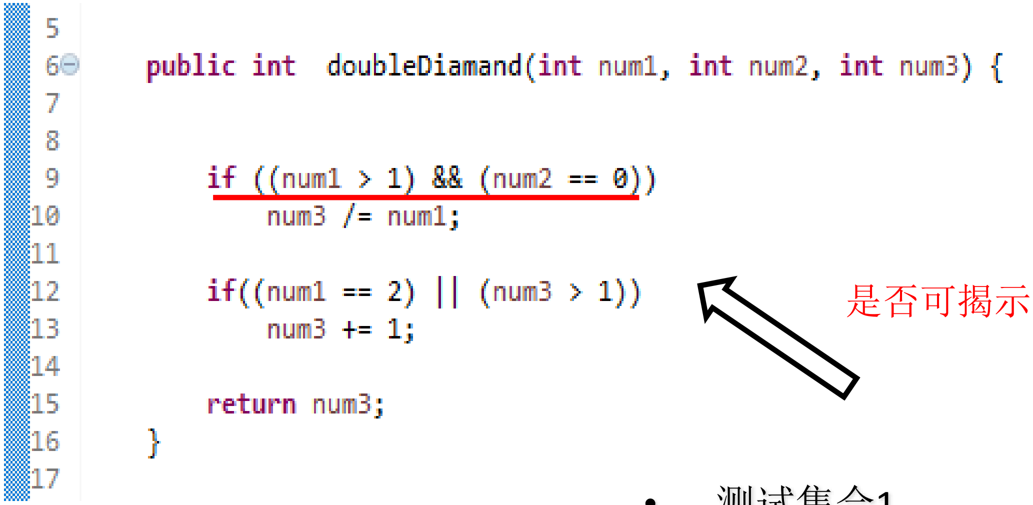


```
public int  doubleDiamand(int num1, int num2, int num3) {


    if ((num1 > 1) && (num2 == 0))
        num3 /= num1;

    if((num1 == 2) || (num3 > 1))
        num3 += 1;

    return num3;
}
```

- 测试集合3
  - ① 测试用例1： [(num1=-2, num2=0, num3=2), 3]
  - ② 测试用例2： [(num1=2, num2=0, num3=2), 2]
  - 语句覆盖度 = 3/3 = 100%, 满足语句覆盖

Statement Coverage

- 语句覆盖（Statement Coverage）
    - 逻辑测试最弱的标准



```
public int  doubleDiamand(int num1, int num2, int num3) {


    if ((num1 > 1) && (num2 == 0))
        num3 /= num1;

    if((num1 == 2) || (num3 > 1))
        num3 += 1;

    return num3;
}
```

是否可揭示&&”错写成 “||”？？？

- 测试集合1

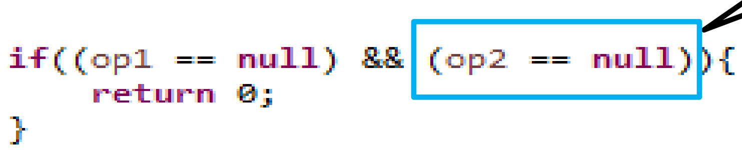
  ① 测试用例1

  [(num1=2, num2=0, num3=4) , 3]

- 语句覆盖度 = 3/3 = 100%, 满足语句覆盖

Decision Coverage

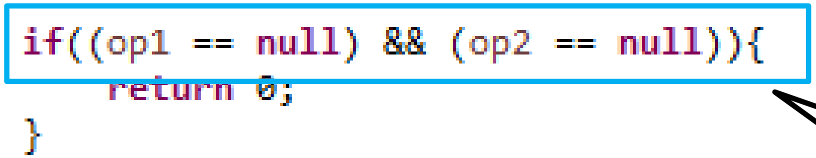
- 条件(Condition)
  - 不含布尔算子的逻辑表达式



```
if((op1 == null) && (op2 == null)){
     return 0;
}
```

条件

- 判定(Decision)
  - 由条件通过1个或多个布尔算子连接起来的逻辑表达式



```
if((op1 == null) && (op2 == null)){
     return 0;
}
```

判定

Decision Coverage

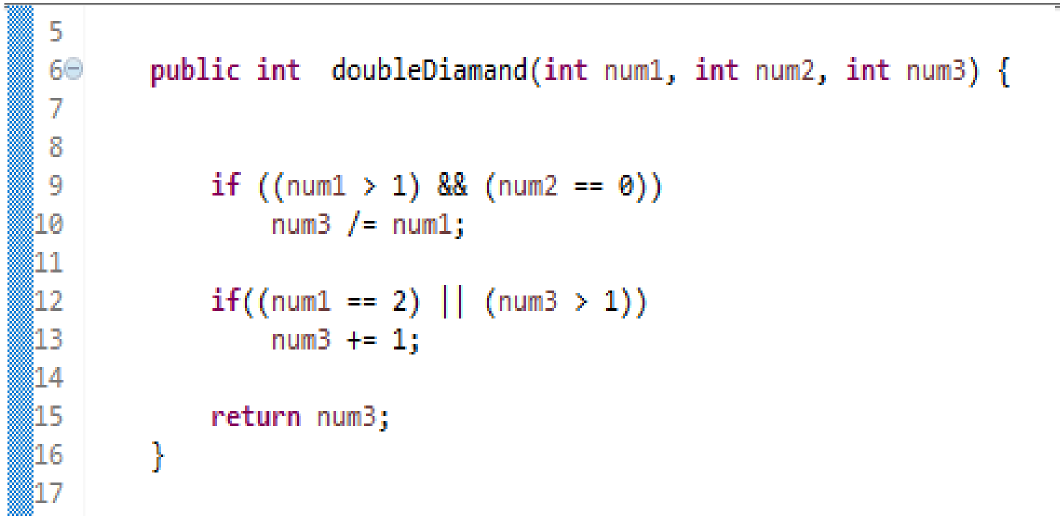
- 判定覆盖（Decision Coverage）
  - 衡量代码中的判定得到执行的程度，期望发现逻辑运算符相关缺陷
  - 如果测试集合能够使得被测代码中的每个判定至少被执行一次,那么则说该测试集合满足了判定覆盖。
  - 注意，每个判定被执行一次的含义是指每个判定的所有可能结果都至少出现一次。
  - 例 if((num1 >1）&& (num2==0))的真假结果都得到执行，才认为该判定被执行。

Decision Coverage

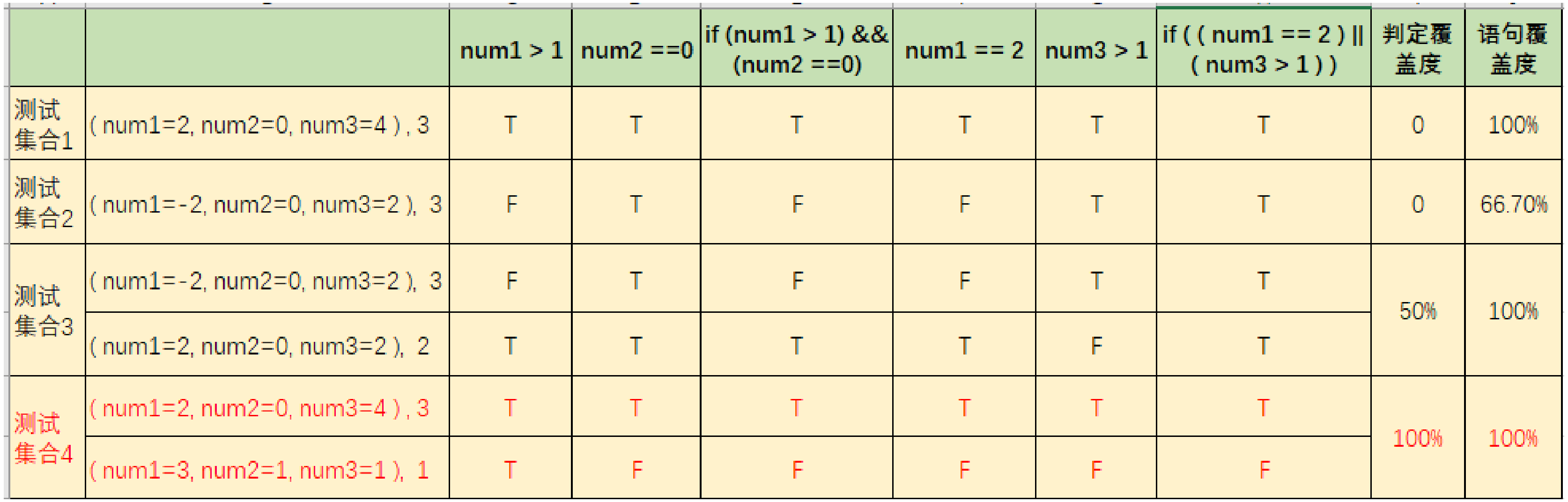
- 判定覆盖（Decision Coverage）
  - 判定覆盖度

$$判定覆盖度 = \frac{得到执行的判定数}{判定总数} * 100\%$$

Decision Coverage

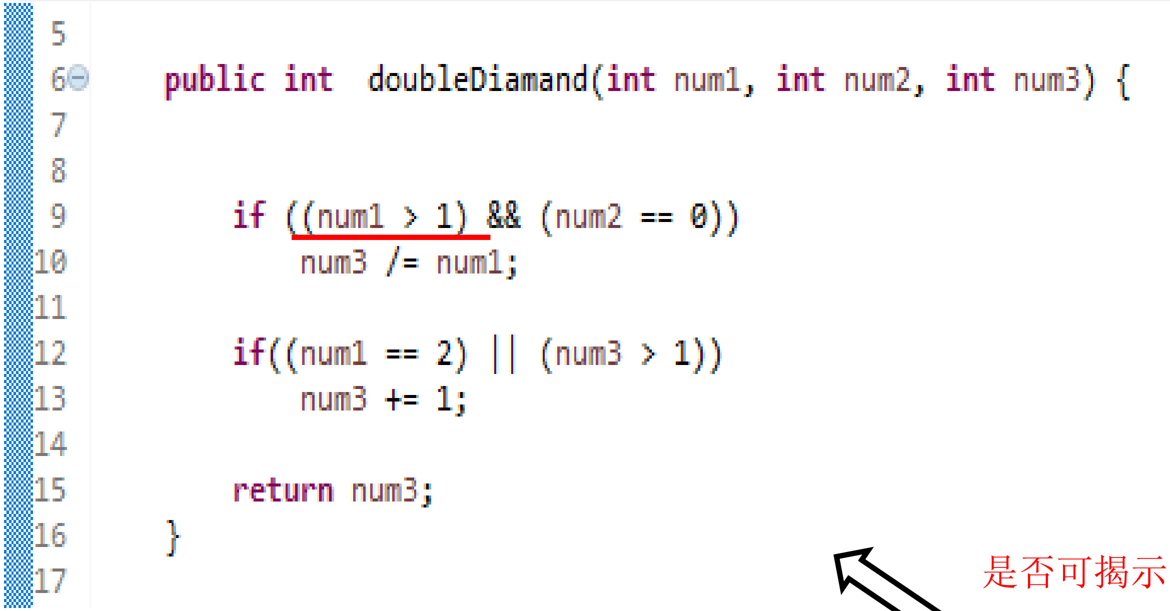


```
public int  doubleDiamand(int num1, int num2, int num3) {


    if ((num1 > 1) && (num2 == 0))
        num3 /= num1;

    if((num1 == 2) || (num3 > 1))
        num3 += 1;

    return num3;
}
```



		num1 > 1	num2 ==0	if (num1 > 1) && (num2 ==0)	num1 == 2	num3 > 1	if ((num1 == 2) || (num3 > 1))	判定覆盖度	语句覆盖度
测试集合1	(num1=2, num2=0, num3=4) , 3	T	T	T	T	T	T	0	100%
测试集合2	(num1=-2, num2=0, num3=2), 3	F	T	F	F	T	T	0	66.70%
测试集合3	(num1=-2, num2=0, num3=2), 3	F	T	F	F	T	T	50%	100%
	(num1=2, num2=0, num3=2), 2	T	T	T	T	F	T		
测试集合4	(num1=2, num2=0, num3=4) , 3	T	T	T	T	T	T	100%	100%
	(num1=3, num2=1, num3=1), 1	T	F	F	F	F	F		

Decision Coverage

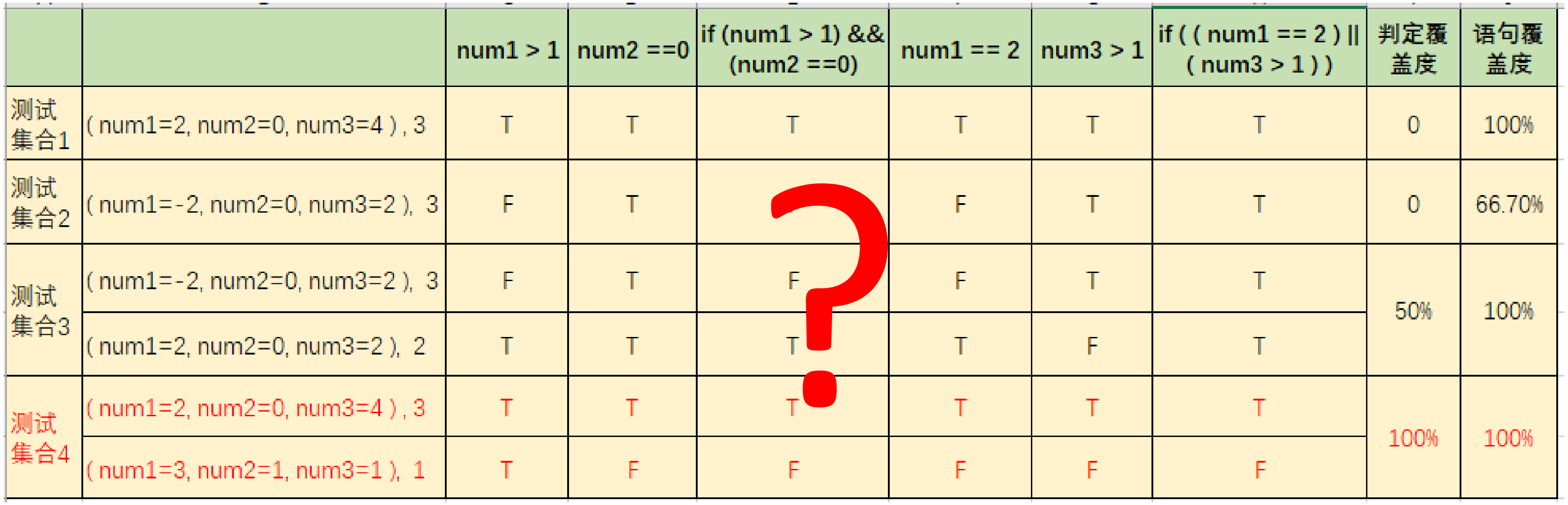


```
public int  doubleDiamand(int num1, int num2, int num3) {


    if ((num1 > 1) && (num2 == 0))
        num3 /= num1;

    if((num1 == 2) || (num3 > 1))
        num3 += 1;

    return num3;
}
```

是否可揭示num1>1错写成num1 > -1？ ？ ？

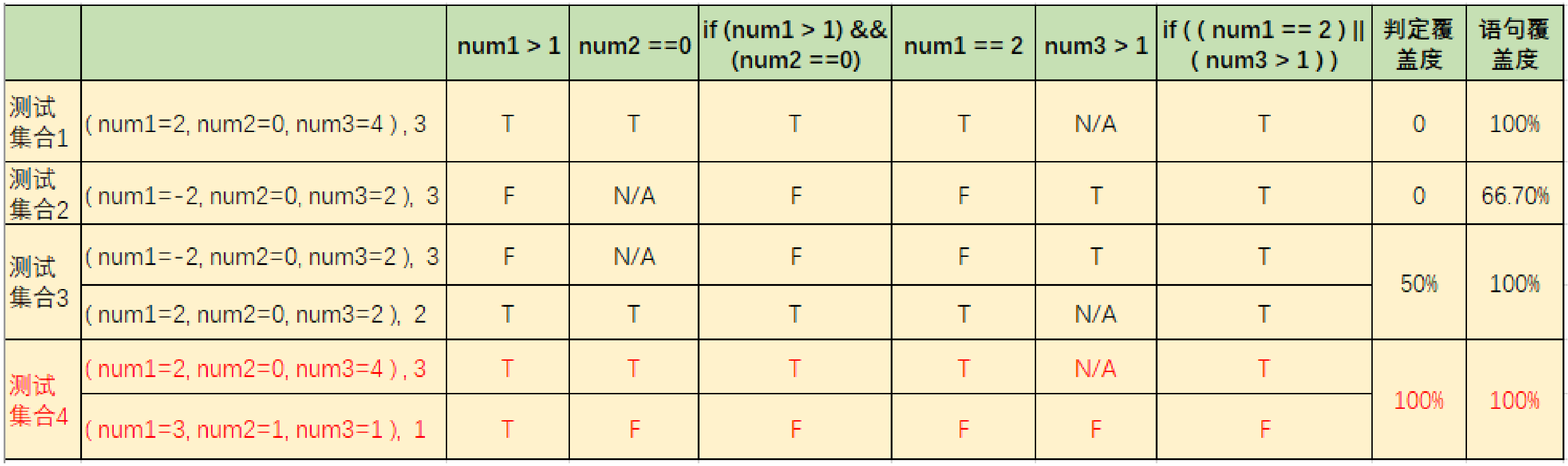
测试 集合4	(num1=2, num2=0, num3=4) , 3	T	T	T	T	T	T	100%	100%
	(num1=3, num2=1, num3=1), 1	T	F	F	F	F	F		

Decision Coverage



		num1 > 1	num2 ==0	if (num1 > 1) && (num2 ==0)	num1 == 2	num3 > 1	if ((num1 == 2) || (num3 > 1))	判定覆盖度	语句覆盖度
测试集合1	(num1=2, num2=0, num3=4) , 3	T	T	T	T	T	T	0	100%
测试集合2	(num1=-2, num2=0, num3=2), 3	F	T	[illegible]	F	T	T	0	66.70%
测试集合3	(num1=-2, num2=0, num3=2), 3	F	T	F	F	T	T	50%	100%
	(num1=2, num2=0, num3=2), 2	T	T	T	T	F	T		
测试集合4	(num1=2, num2=0, num3=4) , 3	T	T	T	T	T	T	100%	100%
	(num1=3, num2=1, num3=1), 1	T	F	F	F	F	F		

Decision Coverage



		num1 > 1	num2 ==0	if (num1 > 1) && (num2 ==0)	num1 == 2	num3 > 1	if ((num1 == 2) || (num3 > 1))	判定覆盖度	语句覆盖度
测试集合1	(num1=2, num2=0, num3=4) , 3	T	T	T	T	N/A	T	0	100%
测试集合2	(num1=-2, num2=0, num3=2), 3	F	N/A	F	F	T	T	0	66.70%
测试集合3	(num1=-2, num2=0, num3=2), 3	F	N/A	F	F	T	T	50%	100%
	(num1=2, num2=0, num3=2), 2	T	T	T	T	N/A	T		
测试集合4	(num1=2, num2=0, num3=4) , 3	T	T	T	T	N/A	T	100%	100%
	(num1=3, num2=1, num3=1), 1	T	F	F	F	F	F		

&&, | | 短路操作符！！！

⇩

MC/DC准则的产生

Condition Coverage

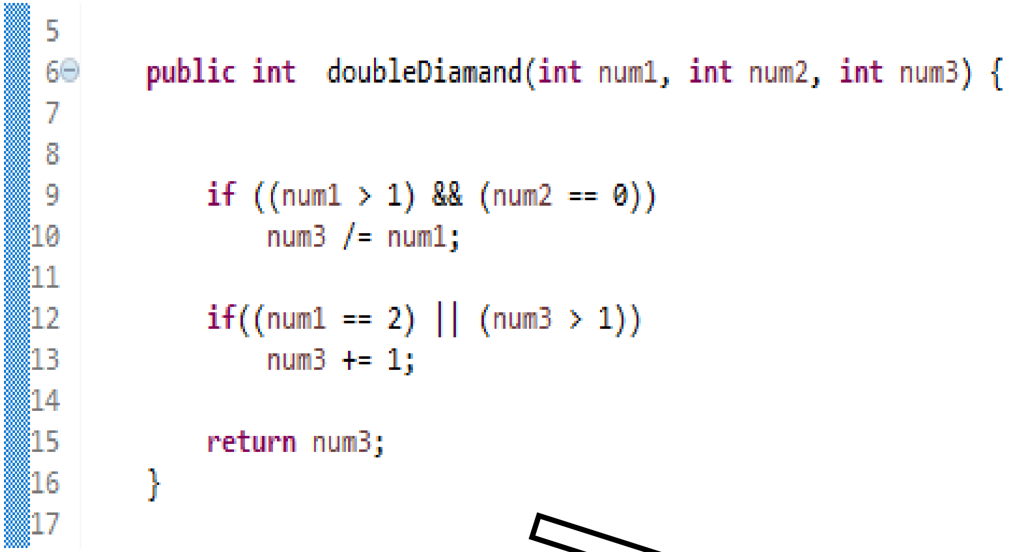
- 条件覆盖（Condition Coverage）
  - 衡量代码中构成判定的各个条件得到执行的程度，期望发现算术运算符相关缺陷
  - 如果测试集合能够使得被测代码中的每个条件至少被执行一次, 那么则说该测试集合满足了条件覆盖。
  - 每个条件被执行一次的含义：每个条件的所有可能结果都至少出现一次。

Condition Coverage

- 条件覆盖（Condition Coverage）
    - 条件覆盖度

$$条件覆盖度 = \frac{得到执行的条件数}{条件总数} * 100\%$$

Condition Coverage



```
public int  doubleDiamand(int num1, int num2, int num3) {


    if ((num1 > 1) && (num2 == 0))
        num3 /= num1;

    if((num1 == 2) || (num3 > 1))
        num3 += 1;

    return num3;
}
```

测试需求

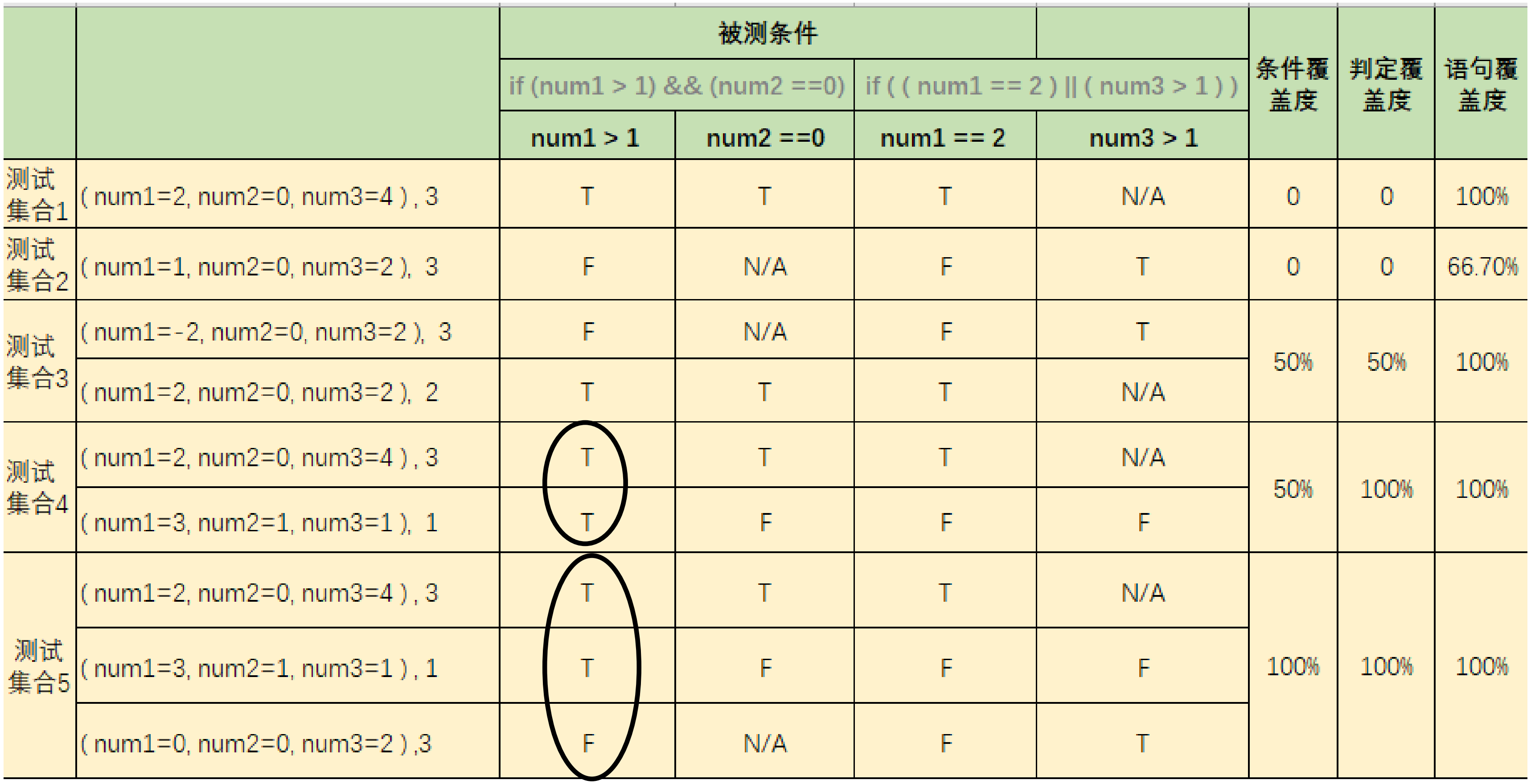
① num1 > 1 取真值和取假值的情况↵

② num2 == 0 取真值和取假值的情况

③ num1 == 2 取真值和取假值的情况

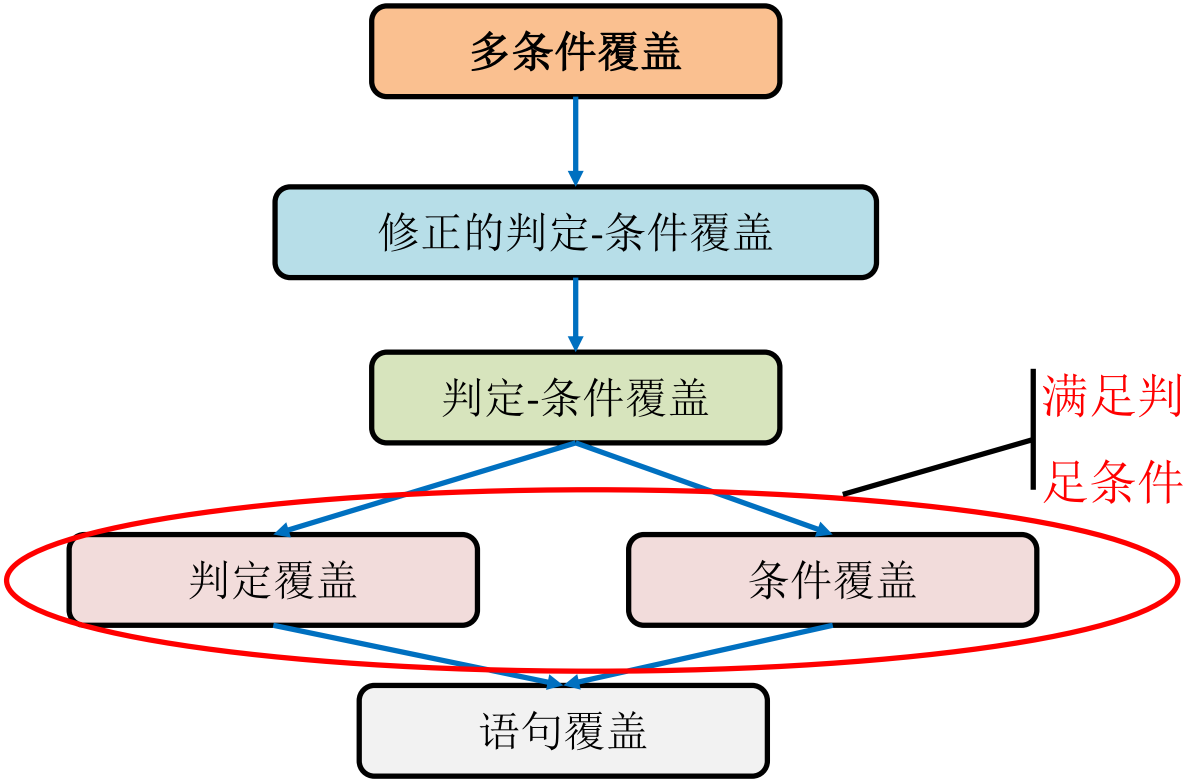
④ num3 > 1 取真值和取假值的情况↵

Condition Coverage



		被测条件				条件覆盖度	判定覆盖度	语句覆盖度
		if (num1 > 1) && (num2 ==0)		if ((num1 == 2) || (num3 > 1))				
		num1 > 1	num2 ==0	num1 == 2	num3 > 1			
测试集合1	(num1=2, num2=0, num3=4) , 3	T	T	T	N/A	0	0	100%
测试集合2	(num1=1, num2=0, num3=2), 3	F	N/A	F	T	0	0	66.70%
测试集合3	(num1=-2, num2=0, num3=2), 3	F	N/A	F	T	50%	50%	100%
	(num1=2, num2=0, num3=2), 2	T	T	T	N/A			
测试集合4	(num1=2, num2=0, num3=4) , 3	T	T	T	N/A	50%	100%	100%
	(num1=3, num2=1, num3=1), 1	T	F	F	F			
测试集合5	(num1=2, num2=0, num3=4) , 3	T	T	T	N/A	100%	100%	100%
	(num1=3, num2=1, num3=1) , 1	T	F	F	F			
	(num1=0, num2=0, num3=2) ,3	F	N/A	F	T			

Notice

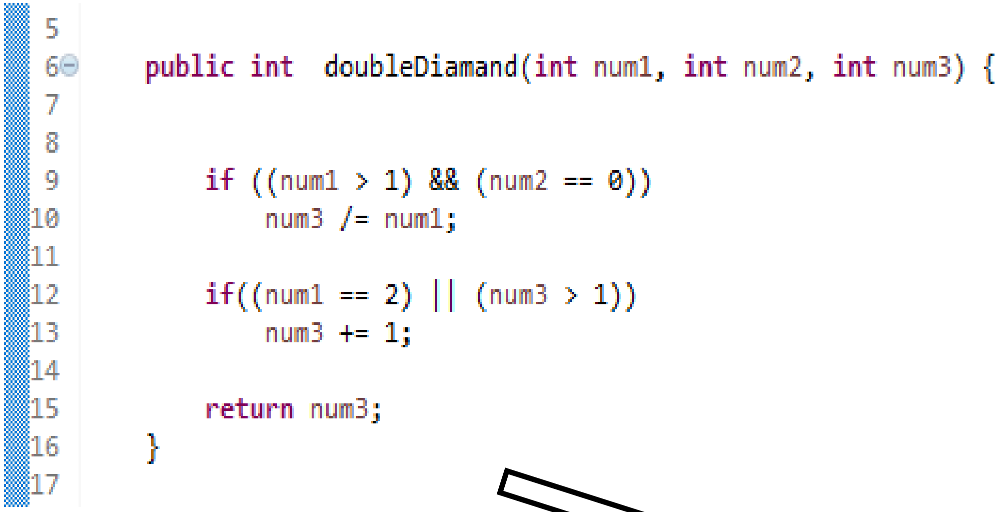


满足判定不一定满足条件，满足条件不一定满足判定

Decision-Condition Coverage

- 判定-条件覆盖（Decision-Condition Coverage）
  - 衡量代码中每个判定以及构成判定的每个条件得到执行的程度。
  - 如果测试集合能够使得被测代码中的每个判定至少被执行一次并且构成判定的每个条件至少被执行一次, 那么则说该测试集合满足了判定-条件覆盖。
  - 执行的含义同样指所有可能结果都至少出现一次

Decision-Condition Coverage



```
public int  doubleDiamand(int num1, int num2, int num3) {


    if ((num1 > 1) && (num2 == 0))
        num3 /= num1;

    if((num1 == 2) || (num3 > 1))
        num3 += 1;

    return num3;
}
```

测试需求

① if((num1>1) && (num2==0))取真值和取假值

② if((num1==2) || (num3 > 1))取真值和取假值↵

③ num1 > 1 取真值和取假值↵

④ num2 == 0 取真值和取假值↵

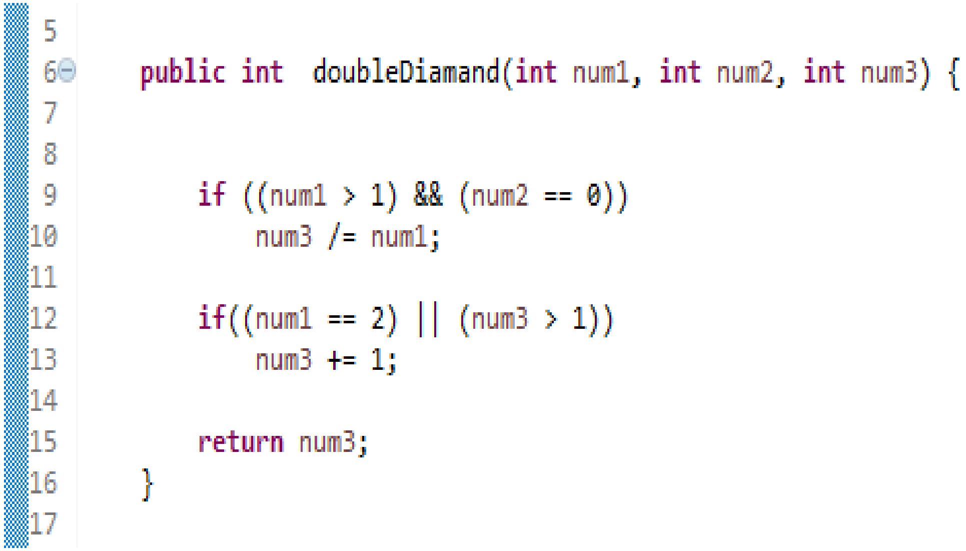
⑤ num1 == 2 取真值和取假值↵

⑥ num3 > 1 取真值和取假值↵

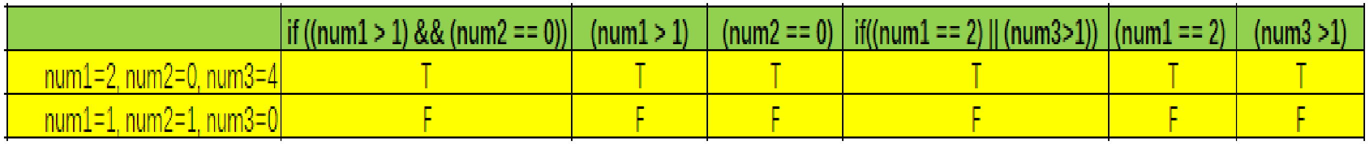
测试集合5满足Decision-Condition Coverage

Modified Decision-Condition Coverage

- 判定-条件覆盖存在的问题
  - 对于某些满足判定-条件覆盖的测试集合而言，其揭错能力并不高
  - 短路运算符



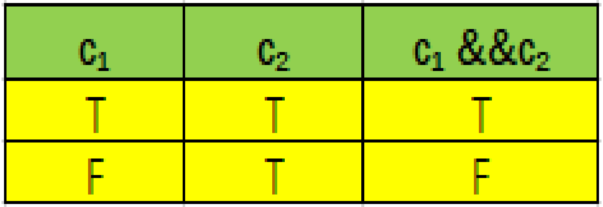
```java

    public int  doubleDiamand(int num1, int num2, int num3) {


        if ((num1 > 1) && (num2 == 0))
            num3 /= num1;

        if((num1 == 2) || (num3 > 1))
            num3 += 1;

        return num3;
    }

```



	if ((num1 > 1) && (num2 == 0))	(num1 > 1)	(num2 == 0)	if((num1 == 2) || (num3>1))	(num1 == 2)	(num3 >1)
num1=2, num2=0, num3=4	T	T	T	T	T	T
num1=1, num2=1, num3=0	F	F	F	F	F	F

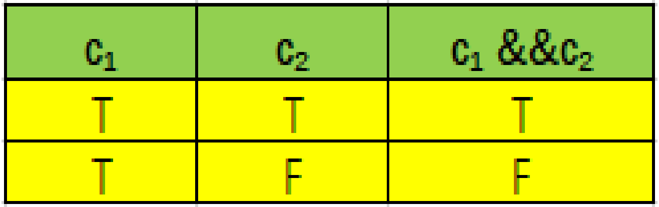
Modified Decision-Condition Coverage

- 修正的判定-条件覆盖（Modified Decision-Condition Coverage，MC/DC）
  - 期望构成每个判定的每个条件能独立地影响整个判定的结果。
  - 在这里独立地影响整个判定的结果是指在其它条件取值不变的情况下，只改变当前条件的取值就能使得整个判定的结果发生变化。



c_1	c_2	c_1 &&c_2
T	T	T
F	T	F

c_1独立影响**c_1 && c_2**的结果



c_1	c_2	c_1 &&c_2
T	T	T
T	F	F

c_2独立影响**c_1 && c_2**的结果

Modified Decision-Condition Coverage

- 确定某条件独立影响判定结果
  - 若使用D表示判定，c_i表示D的第i个条件，$D_{ci=true}$表示将D中所有c_i使用true替换之后的判定表达式，$D_{ci=false}$表示将D中所有c_i使用false替换之后的判定表达式，那么逻辑表达式$Dc_i= D_{ci=true} \oplus D_{ci=false}$可以用于计算$c_i$独立影响判定时，其它条件的测试输入值

Modified Decision-Condition Coverage

$D = c_1 \&\& c_2$

$Dc_1 = D_{c1=true} \oplus D_{c1=false}$

$= (true \&\& c_2) \oplus (false \&\& c_2)$

$= c_2 \oplus false$

$= c_2$

$Dc_2 = D_{c2=true} \oplus D_{c2=false}$

$= (c_1 \&\& true) \oplus (c_1 \&\& false)$

$= c_1 \oplus false$

$= c_1$

c_2=true时，c_1将独立影响整个表达式的结果，

Modified Decision-Condition Coverage

$D = c_1 \ \&\& \ (c_2 || c_3)$

$Dc_1 = D_{c1=true} \oplus D_{c1=false}$

$= (true \ \&\& \ (c_2||c_3)\ ) \oplus (false \ \&\& \ (c_2||c_3)\ )$

$= (c_2||c_3) \oplus false$

$= (c_2||c_3)$

有3种c2，c3的取值可以使得c1独立影响整个表达式结果

3种测试输入：

① { [c1 = true, c2=true, c3=true], [c1 = false, c2=true, c3=true] }

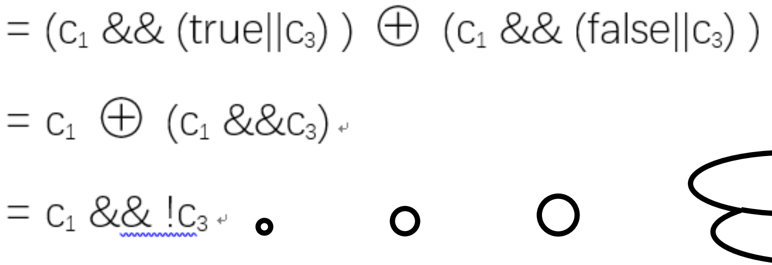
② { [c_1 = true, c_2=true, c_3=false], [c_1 = false, c_2=true, c_3=false] }

③ { [c_1 = true, c_2=false, c_3=true], [c_1 = false, c_2=false, c_3=true] }

Modified Decision-Condition Coverage

$D = c_1 \&\&$ （$c_2||c_3$）

$Dc_2 = D_{c2=true} \oplus D_{c2=false}$



$= (c_1 \&\& (true||c_3)) \oplus (c_1 \&\& (false||c_3))$

$= c_1 \oplus (c_1 \&\& c_3)$

$= c_1 \&\& !c_3$

有1种c1，c3的取值可以使得c2独立影响整个表达式结果

1种测试输入：

① { [c_1 = true, c_2=true, c_3=false], [c_1 = true, c_2=false, c_3=false] }

Modified Decision-Condition Coverage

$B = c_1 \ \&\& \ (c_2 || c_3)$

$Dc_3 = D_{c3=true} \oplus D_{c3=false}$

$= (c_1 \ \&\& \ (c_2 || true)) \oplus (c_1 \ \&\& \ (c_2 || false))$

$= c_1 \oplus (c_1 \ \&\& c_2)$

$= c_1 \ \&\& \ !c_2$

有1种c1，c2的取值可以使得c3独立影响整个表达式结果

1种测试输入：

① { [c1 = true, c2 = false, c3 = true], [c1 = true, c2 = false, c3 = false] }

Modified Decision-Condition Coverage

$D = c_1$ && （c_2||c_3）

c1: 3种测试输入：

① { [c1 = true, c2=true, c3=true], [c1 = false, c2=true, c3=true] }

② { [c_1 = true, c_2=true, c_3=false], [c_1 = false, c_2=true, c_3=false] }

③ { [c_1 = true, c_2=false, c_3=true], [c_1 = false, c_2=false, c_3=true] }

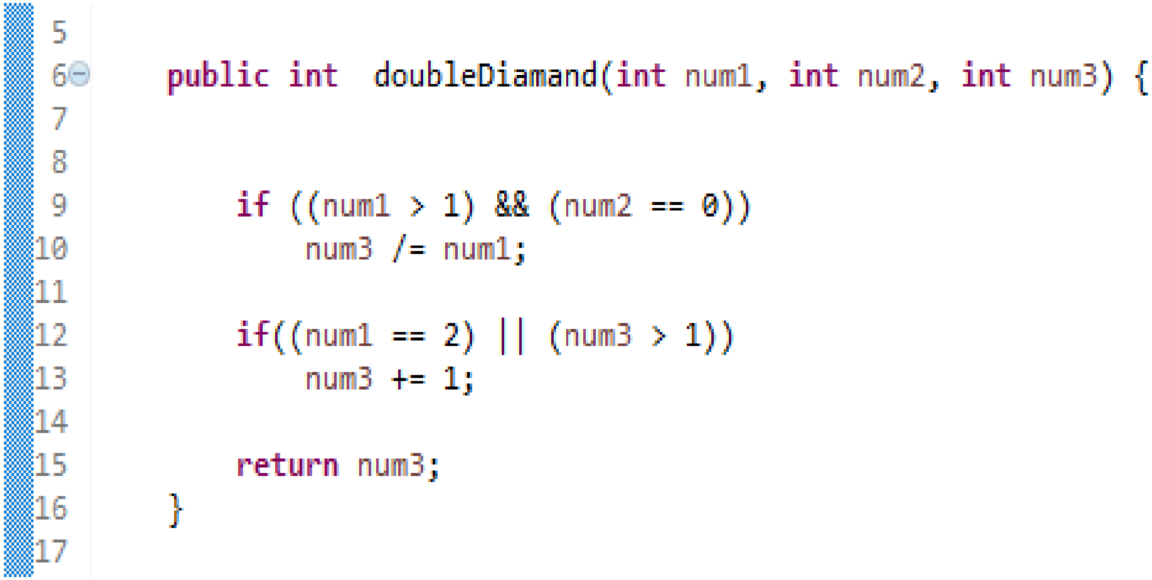
c2: 1种测试输入：

① { [c_1 = true, c_2=true, c_3=false], [c_1 = true, c_2=false, c_3=false] }

c3: 1种测试输入：

① { [c1 = true, c2 = false, c3 = true], [c1 = true, c2 = false, c3 = false] }

Modified Decision-Condition Coverage



```
    public int  doubleDiamand(int num1, int num2, int num3) {


        if ((num1 > 1) && (num2 == 0))
            num3 /= num1;

        if((num1 == 2) || (num3 > 1))
            num3 += 1;

        return num3;
    }

```

num1 > 1 独立影响(num1>1) &&(num2 == 0)

if ((num1 > 1) && (num2 == 0))	
(num1 > 1)	(num2 == 0)
T	T
F	T

num1== 2 独立影响 (num1 == 2) | | (num3 > 1)

if((num1 == 2) || (num3 >1))	
(num1 == 2)	(num3 >1)
T	F
F	F

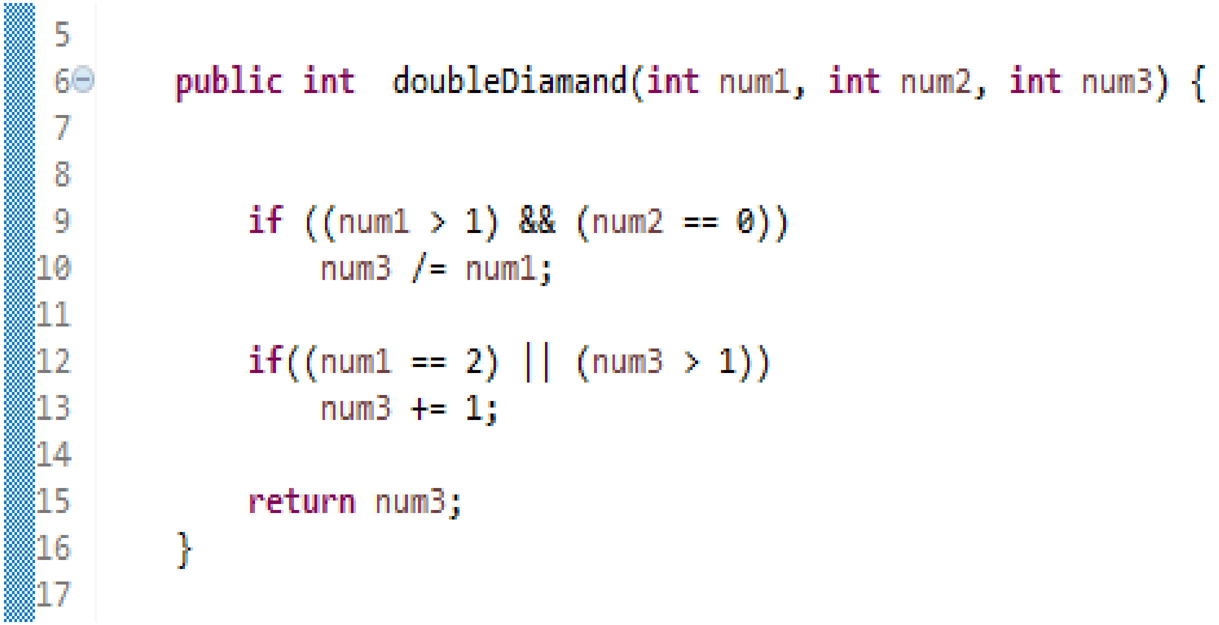
num2 == 0 独立影响(num1>1) &&(num2 == 0)

if ((num1 > 1) && (num2 == 0))	
(num1 > 1)	(num2 == 0)
T	T
T	F

num3 > 1 独立影响 (num1 == 2) | | (num3 > 1)

if((num1 == 2) || (num3 >1))	
(num1 == 2)	(num3 >1)
F	T
F	F

Modified Decision-Condition Coverage

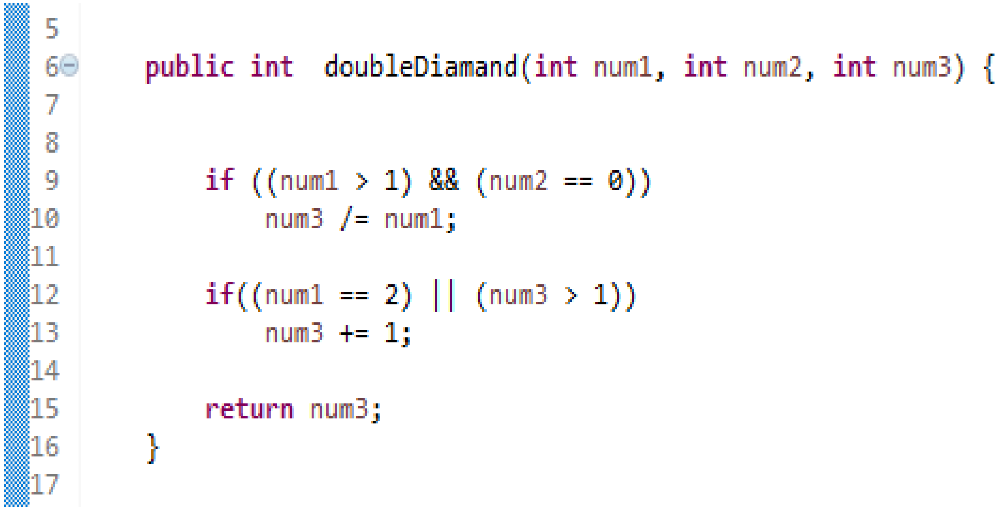


```
public int  doubleDiamand(int num1, int num2, int num3) {


    if ((num1 > 1) && (num2 == 0))
        num3 /= num1;

    if((num1 == 2) || (num3 > 1))
        num3 += 1;

    return num3;
}
```

✓满足修正的判定-条件覆盖（同时也满足判定-条件覆盖）

测试集合5	(num1=2, num2=0, num3=4) , 3	T	T	T	N/A	100%	100%	100%
	(num1=3, num2=1, num3=1) , 1	T	F	F	F			
	(num1=0, num2=0, num3=2) ,3	F	N/A	F	T			

Multiple Condition Coverage



```
public int  doubleDiamand(int num1, int num2, int num3) {


    if ((num1 > 1) && (num2 == 0))
        num3 /= num1;

    if((num1 == 2) || (num3 > 1))
        num3 += 1;

    return num3;
}
```

① num1>1 为真且 num2==0 为真

② num1>1 为真且 num2==0 为假

③ num1>1 为假且 num2==0 为真

④ num1>1 为假且 num2==0 为假

⑤ num1==2 为真且 num3>1 为真

⑥ num1==2 为真且 num3>1 为假

⑦ num1==2 为假且 num3>1 为真

⑧ num1==2 为假且 num3>1 为假

给出一个满足多条件覆盖的测试集合吧！

Logical Testing & Tools

- Logical Coverage Criteria
  - Statement Coverage
  - Decision Coverage
  - Condition Coverage
  - Decision-Condition Coverage
  - Modified Decision-Condition Coverage
  - Multiple Condition Coverage
- Logical Coverage Criteria Tools

Example

- specification

An absSum method takes two integer arguments and then return the absolute sum of the two arguments. An Integer type can hold a NULL value, so the method checks for NULL. If both arguments are NULL, then 0 is returned.

- Test cases:

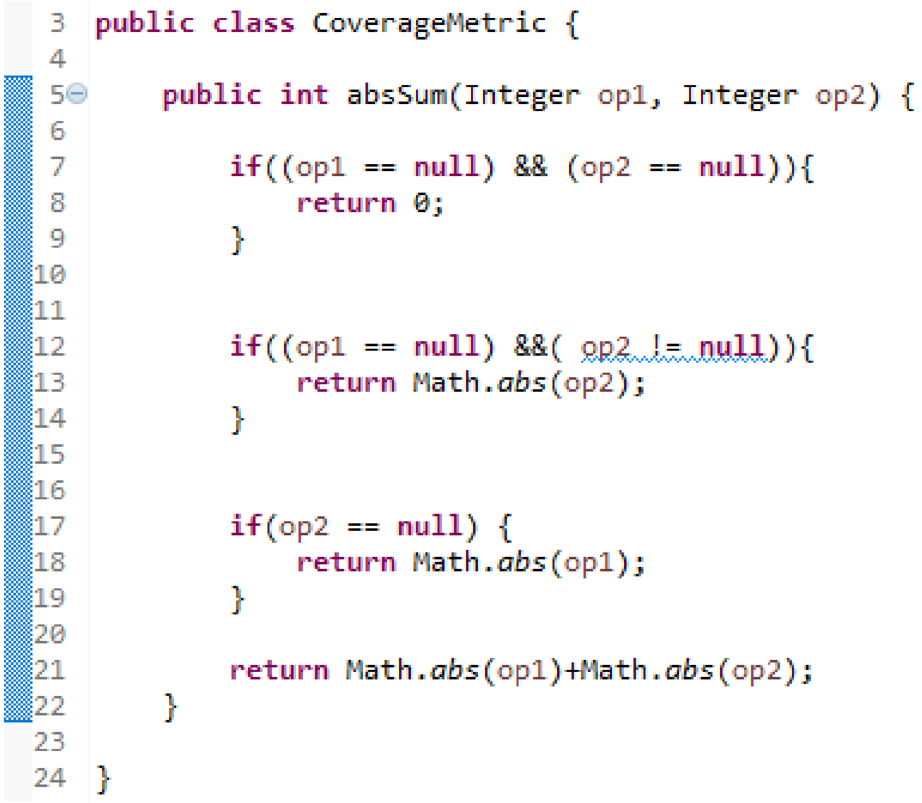
Test inputs, expected result

① (op1 =null, op2 = null), 0

② (op1 = null, op2 = 10), 10

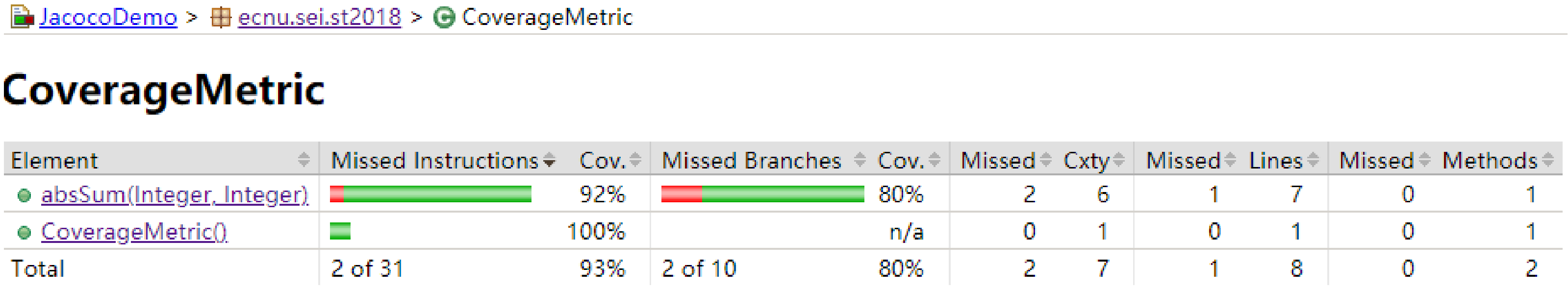
③ (op1=10, op2 = null), 10

④ (op1 = 10, op2 = 10), 20



```
public class CoverageMetric {

    public int absSum(Integer op1, Integer op2) {

        if((op1 == null) && (op2 == null)){
            return 0;
        }


        if((op1 == null) &&( op2 != null)){
            return Math.abs(op2);
        }


        if(op2 == null) {
            return Math.abs(op1);
        }

        return Math.abs(op1)+Math.abs(op2);
    }

}
```

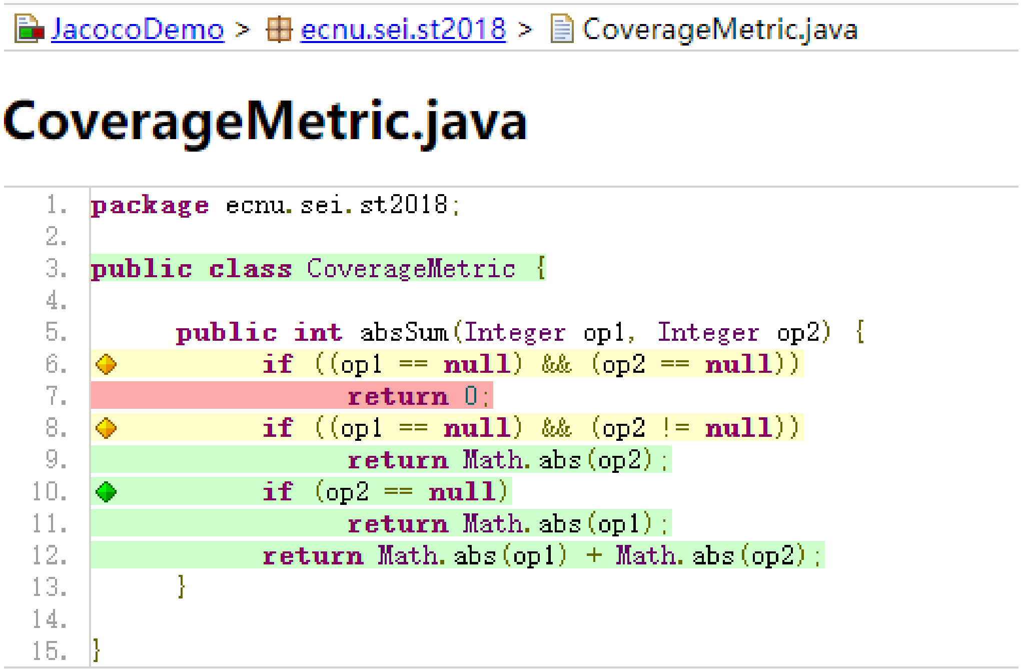
Coverage Report



JacocoDemo > ecnu.sei.st2018 > CoverageMetric

CoverageMetric

Element	Missed Instructions	Cov.	Missed Branches	Cov.	Missed	Cxty	Missed	Lines	Missed	Methods
absSum(Integer, Integer)	🟥🟩	92%	🟥🟩	80%	2	6	1	7	0	1
CoverageMetric()	🟩	100%		n/a	0	1	0	1	0	1
Total	2 of 31	93%	2 of 10	80%	2	7	1	8	0	2



JacocoDemo > ecnu.sei.st2018 > CoverageMetric.java

CoverageMetric.java

```
package ecnu.sei.st2018;

public class CoverageMetric {

	public int absSum(Integer op1, Integer op2) {
		if ((op1 == null) && (op2 == null))
			return 0;
		if ((op1 == null) && (op2 != null))
			return Math.abs(op2);
		if (op2 == null)
			return Math.abs(op1);
		return Math.abs(op1) + Math.abs(op2);
	}

}
```

红色背景：没有指令被执行的代码行

黄色背景：部分指令被执行的代码行

绿色背景：全部指令被执行的代码行

红色菱形：没有被执行的分支

黄色菱形：部分被执行的分支

绿色菱形：全部被执行的分支

Jacoco

- JaCoCo (http://jacoco.org/jacoco/) which is a coverage metric library and works on byte code level
- JaCoCo Coverage Counters
  - Instructions
  - Branches
  - Cyclomatic Complexity
  - Lines
  - Methods
  - Classes

JaCoCo

- JaCoCo Coverage Counters
  - Instructions:
    - The smallest unit JaCoCo counts are single Java byte code instructions.
    - Instruction coverage provides information about the amount of code that has been executed or missed.
    - This metric is completely independent from source formatting and always available
  - Branches
    - calculates branch coverage for all if and switch statements.
      - No coverage: No branches in the line has been executed (**red diamond**)
      - Partial coverage: Only a part of the branches in the line have been executed (**yellow diamond**)
      - Full coverage: All branches in the line have been executed (**green diamond**)

JaCoCo

- JaCoCo Coverage Counters
  - Cyclomatic Complexity
    - the minimum number of paths that can, in (linear) combination, generate all possible paths through a method
  - Lines
    - A source line is considered executed when at least one instruction that is assigned to this line has been executed
      - No coverage: No instruction in the line has been executed (red background)
      - Partial coverage: Only a part of the instruction in the line have been executed (yellow background)
      - Full coverage: All instructions in the line have been executed (green background)

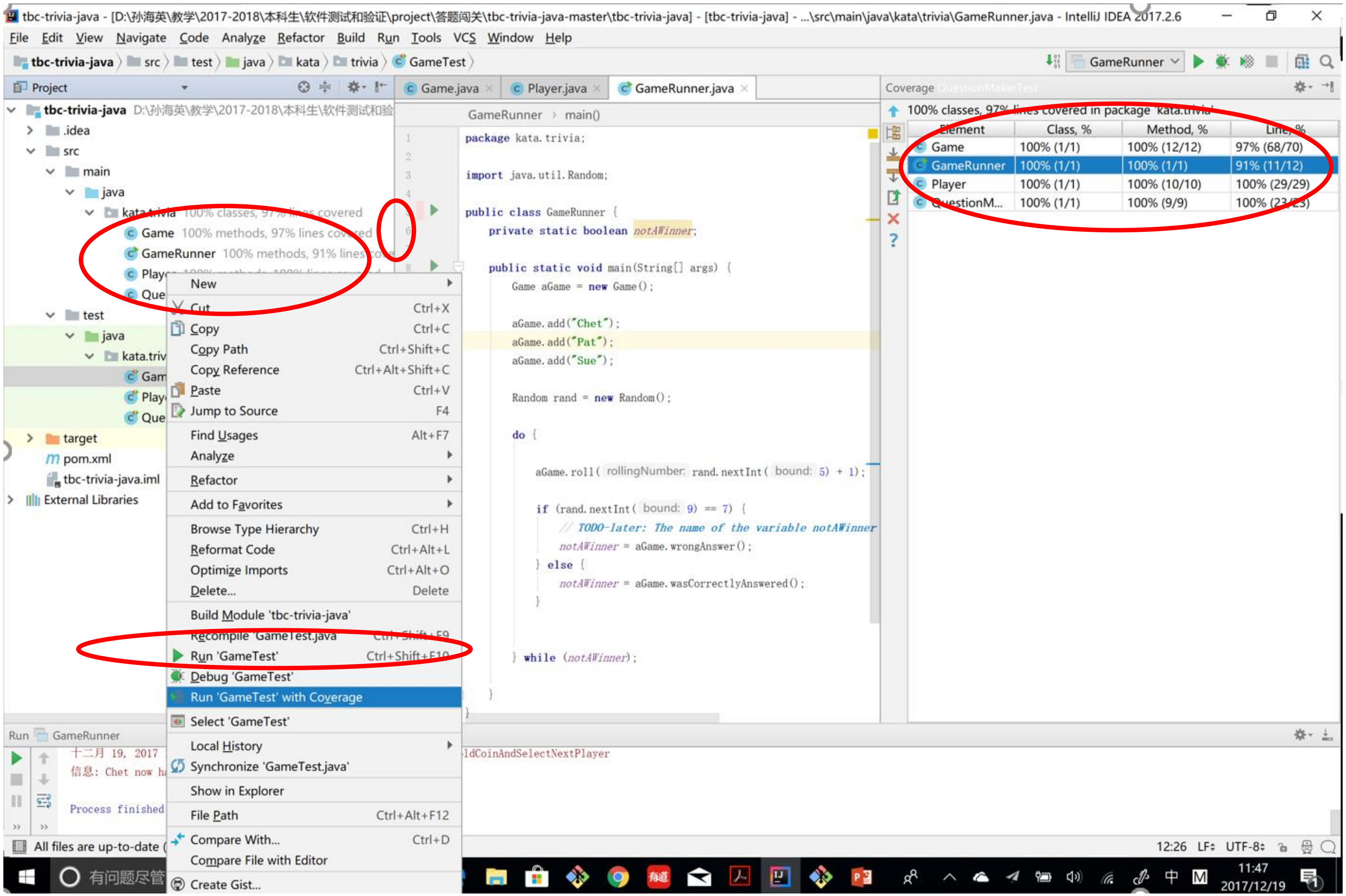
JaCoCo

- JaCoCo Coverage Counters
  - Methods
    - Each non-abstract method contains at least one instruction.
    - A method is considered as executed when at least one instruction has been executed.
  - Classes
    - A class is considered as executed when at least one of its methods has been executed

IDEA Code Coverage Tool

- IntelliJ IDEA Code Coverage Runner/JaCoCo
  - Specify how you want to process the coverage results.
    - Select Coverage Tool and Modes
  - Create tests for the target code
  - Configure code coverage measurement in the desired run/debug configuration.
  - Run with coverage
  - After running with coverage has been executed,
    ① View code coverage data.
    ② Generate code coverage report.

IntelliJ IDEA code coverage runner



tbc-trivia-java - [D:\孙海英\教学\2017-2018\本科生\软件测试和验证\project\答题闯关\tbc-trivia-java-master\tbc-trivia-java] - [tbc-trivia-java] - ...\src\main\java\kata\trivia\GameRunner.java - IntelliJ IDEA 2017.2.6

tbc-trivia-java D:\孙海英\教学\2017-2018\本科生\软件测试和验证

kata.trivia 100% classes, 97% lines covered

Game 100% methods, 97% lines covered

GameRunner 100% methods, 91% lines co

```
package kata.trivia;

import java.util.Random;

public class GameRunner {
    private static boolean notAWinner;

    public static void main(String[] args) {
        Game aGame = new Game();

        aGame.add("Chet");
        aGame.add("Pat");
        aGame.add("Sue");

        Random rand = new Random();

        do {

            aGame.roll( rollingNumber: rand.nextInt( bound: 5) + 1);

            if (rand.nextInt( bound: 9) == 7) {
                // TODO-later: The name of the variable notAWinner
                notAWinner = aGame.wrongAnswer();
            } else {
                notAWinner = aGame.wasCorrectlyAnswered();
            }


        } while (notAWinner);

    }
}
```

Coverage QuestionMakerTest

100% classes, 97% lines covered in package 'kata.trivia'

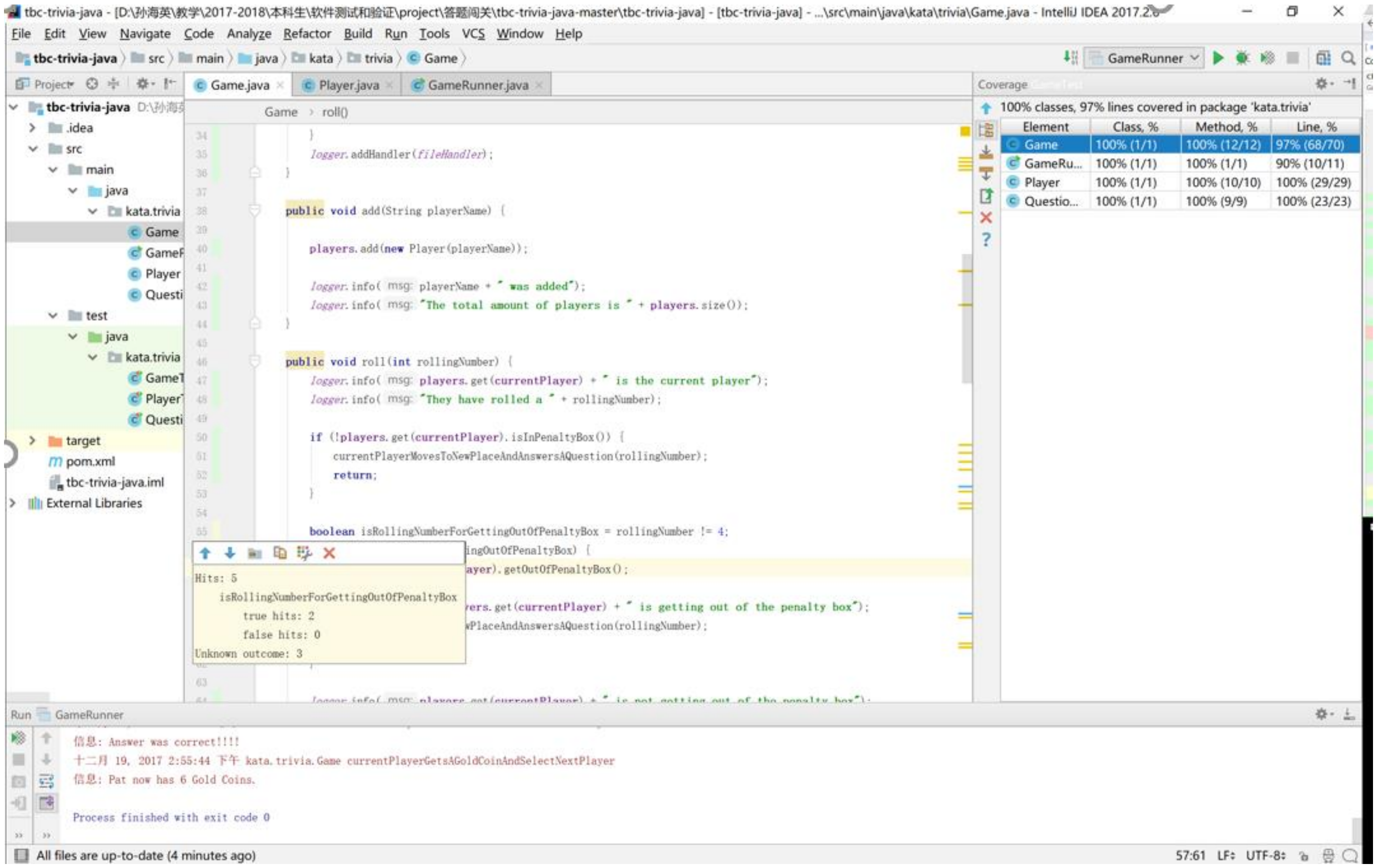
Element	Class, %	Method, %	Line, %
Game	100% (1/1)	100% (12/12)	97% (68/70)
GameRunner	100% (1/1)	100% (1/1)	91% (11/12)
Player	100% (1/1)	100% (10/10)	100% (29/29)
QuestionM...	100% (1/1)	100% (9/9)	100% (23/23)

```
十二月 19, 2017
信息: Chet now h
...ldCoinAndSelectNextPlayer

Process finished
```

All files are up-to-date (

IntelliJ IDEA code coverage runner

- Trace Mode (Run->Edit Configuration->Trace Modes)


tbc-trivia-java - [D:\孙海英\教学\2017-2018\本科生\软件测试和验证\project\答题闯关\tbc-trivia-java-master\tbc-trivia-java] - [tbc-trivia-java] - ...\src\main\java\kata\trivia\Game.java - IntelliJ IDEA 2017.2.6
File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help
tbc-trivia-java › src › main › java › kata › trivia › Game
GameRunner
Project
tbc-trivia-java
.idea
src
main
java
kata.trivia
Game
GameRunner
Player
Questions
test
java
kata.trivia
GameTest
PlayerTest
QuestionsTest
target
pom.xml
tbc-trivia-java.iml
External Libraries
Game.java
Player.java
GameRunner.java
Game › roll()
Coverage
Run
GameRunner

```
        }
        logger.addHandler(fileHandler);
    }

    public void add(String playerName) {

        players.add(new Player(playerName));

        logger.info( msg: playerName + " was added");
        logger.info( msg: "The total amount of players is " + players.size());
    }

    public void roll(int rollingNumber) {
        logger.info( msg: players.get(currentPlayer) + " is the current player");
        logger.info( msg: "They have rolled a " + rollingNumber);

        if (!players.get(currentPlayer).isInPenaltyBox()) {
            currentPlayerMovesToNewPlaceAndAnswersAQuestion(rollingNumber);
            return;
        }

        boolean isRollingNumberForGettingOutOfPenaltyBox = rollingNumber != 4;
                          ingOutOfPenaltyBox) {
                          ayer).getOutOfPenaltyBox();

                          ers.get(currentPlayer) + " is getting out of the penalty box");
                          PlaceAndAnswersAQuestion(rollingNumber);

        }

        logger.info( msg: players.get(currentPlayer) + " is not getting out of the penalty box");
```

```
Hits: 5
    isRollingNumberForGettingOutOfPenaltyBox
        true hits: 2
        false hits: 0
Unknown outcome: 3
```

100% classes, 97% lines covered in package 'kata.trivia'

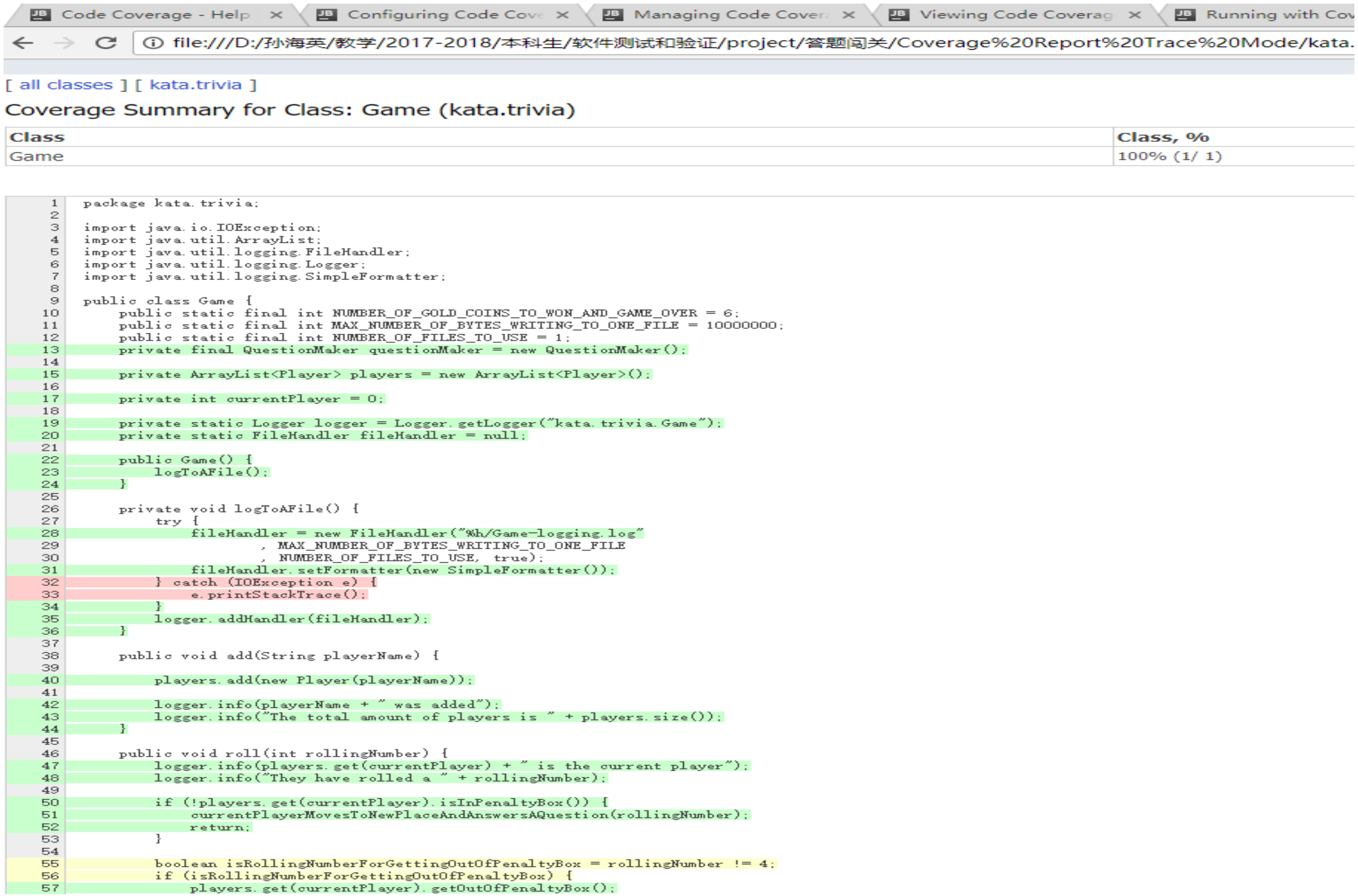
Element	Class, %	Method, %	Line, %
Game	100% (1/1)	100% (12/12)	97% (68/70)
GameRu...	100% (1/1)	100% (1/1)	90% (10/11)
Player	100% (1/1)	100% (10/10)	100% (29/29)
Questio...	100% (1/1)	100% (9/9)	100% (23/23)

```
信息: Answer was correct!!!!
十二月 19, 2017 2:55:44 下午 kata.trivia.Game currentPlayerGetsAGoldCoinAndSelectNextPlayer
信息: Pat now has 6 Gold Coins.

Process finished with exit code 0
```

All files are up-to-date (4 minutes ago)

57:61 LF: UTF-8:

IntelliJ IDEA Code Coverage Runner



file:///D:/孙海英/教学/2017-2018/本科生/软件测试和验证/project/答题闯关/Coverage%20Report%20Trace%20Mode/kata.

[all classes] [kata.trivia]

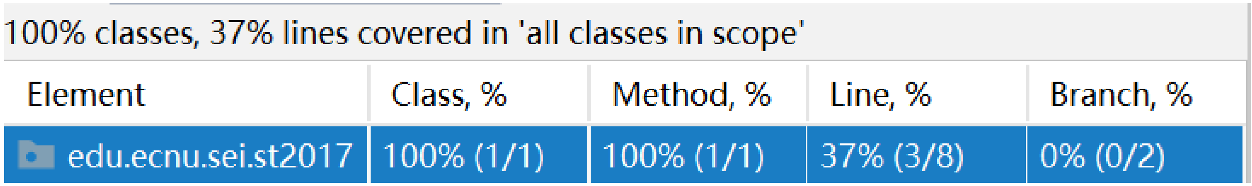
Coverage Summary for Class: Game (kata.trivia)

Class	Class, %
Game	100% (1/ 1)

```java
package kata.trivia;

import java.io.IOException;
import java.util.ArrayList;
import java.util.logging.FileHandler;
import java.util.logging.Logger;
import java.util.logging.SimpleFormatter;

public class Game {
    public static final int NUMBER_OF_GOLD_COINS_TO_WON_AND_GAME_OVER = 6;
    public static final int MAX_NUMBER_OF_BYTES_WRITING_TO_ONE_FILE = 10000000;
    public static final int NUMBER_OF_FILES_TO_USE = 1;
    private final QuestionMaker questionMaker = new QuestionMaker();

    private ArrayList<Player> players = new ArrayList<Player>();

    private int currentPlayer = 0;

    private static Logger logger = Logger.getLogger("kata.trivia.Game");
    private static FileHandler fileHandler = null;

    public Game() {
        logToAFile();
    }

    private void logToAFile() {
        try {
            fileHandler = new FileHandler("%h/Game-logging.log"
                    , MAX_NUMBER_OF_BYTES_WRITING_TO_ONE_FILE
                    , NUMBER_OF_FILES_TO_USE, true);
            fileHandler.setFormatter(new SimpleFormatter());
        } catch (IOException e) {
            e.printStackTrace();
        }
        logger.addHandler(fileHandler);
    }

    public void add(String playerName) {

        players.add(new Player(playerName));

        logger.info(playerName + " was added");
        logger.info("The total amount of players is " + players.size());
    }

    public void roll(int rollingNumber) {
        logger.info(players.get(currentPlayer) + " is the current player");
        logger.info("They have rolled a " + rollingNumber);

        if (!players.get(currentPlayer).isInPenaltyBox()) {
            currentPlayerMovesToNewPlaceAndAnswersAQuestion(rollingNumber);
            return;
        }

        boolean isRollingNumberForGettingOutOfPenaltyBox = rollingNumber != 4;
        if (isRollingNumberForGettingOutOfPenaltyBox) {
            players.get(currentPlayer).getOutOfPenaltyBox();
```

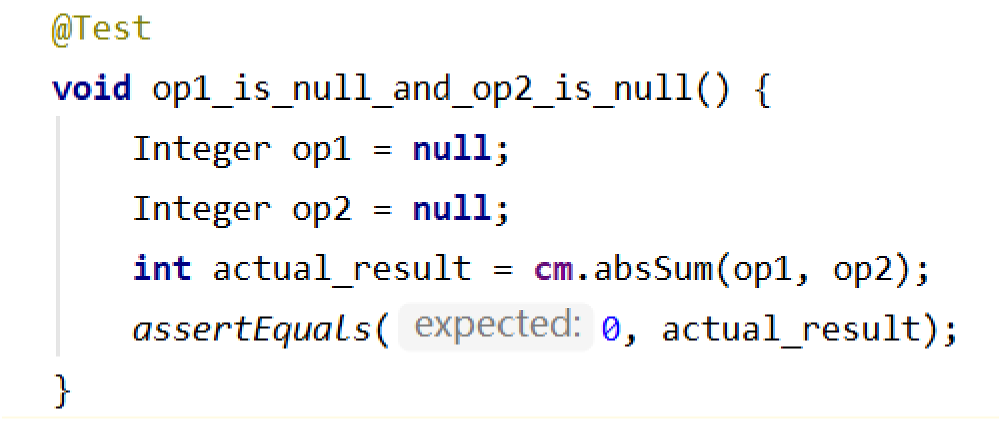
Branch Coverage in IDEA

- IDEA branch coverage的计算方法与Jacoco不一样，而且似乎有缺陷



100% classes, 37% lines covered in 'all classes in scope'

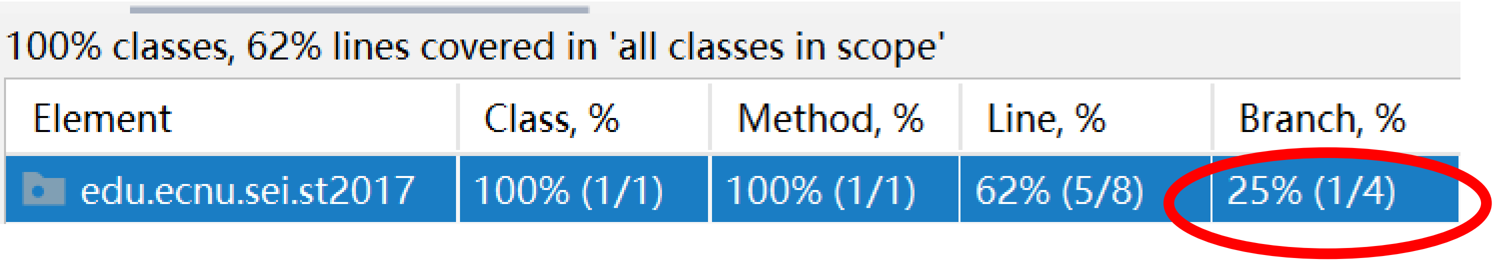
Element	Class, %	Method, %	Line, %	Branch, %
edu.ecnu.sei.st2017	100% (1/1)	100% (1/1)	37% (3/8)	0% (0/2)



```
@Test
void op1_is_null_and_op2_is_null() {
    Integer op1 = null;
    Integer op2 = null;
    int actual_result = cm.absSum(op1, op2);
    assertEquals( expected: 0, actual_result);
}
```

Branch的真假都测到才算被覆盖

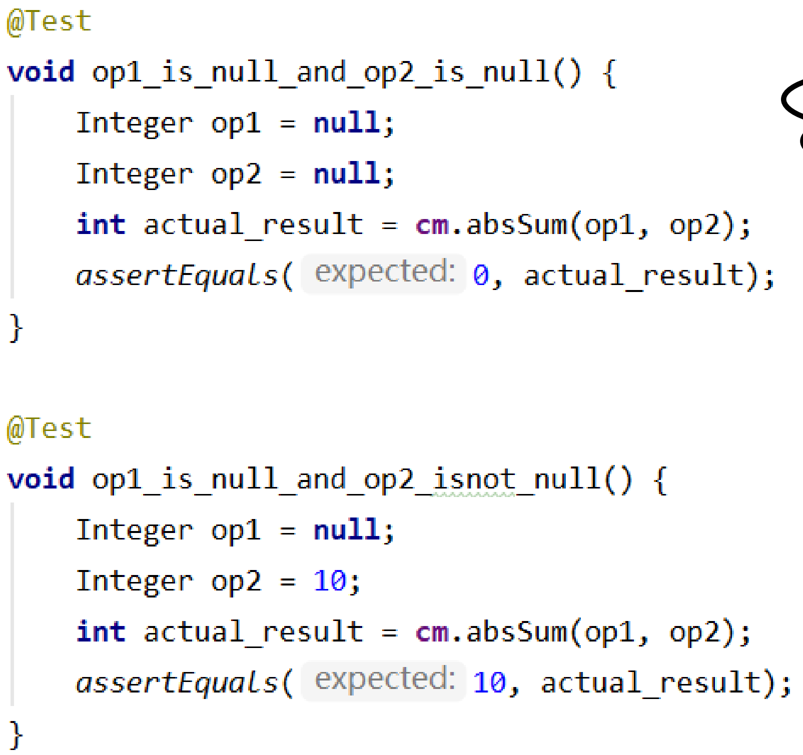
Branch Coverage in IDEA



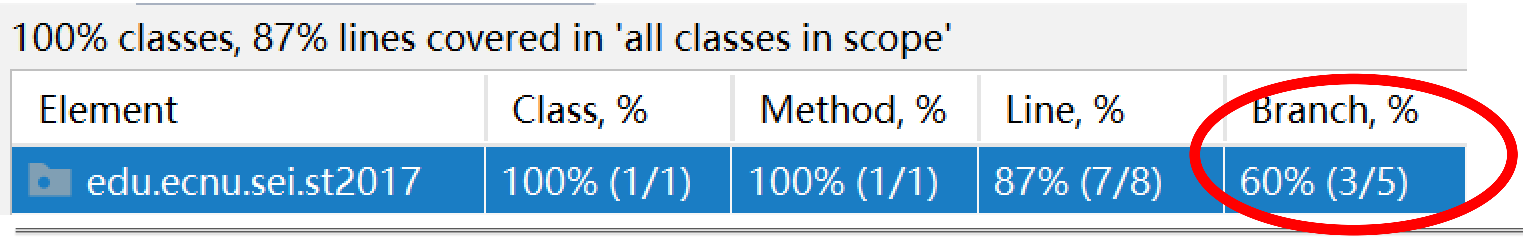
100% classes, 62% lines covered in 'all classes in scope'

Element	Class, %	Method, %	Line, %	Branch, %
edu.ecnu.sei.st2017	100% (1/1)	100% (1/1)	62% (5/8)	25% (1/4)

Branch的总数发生变化

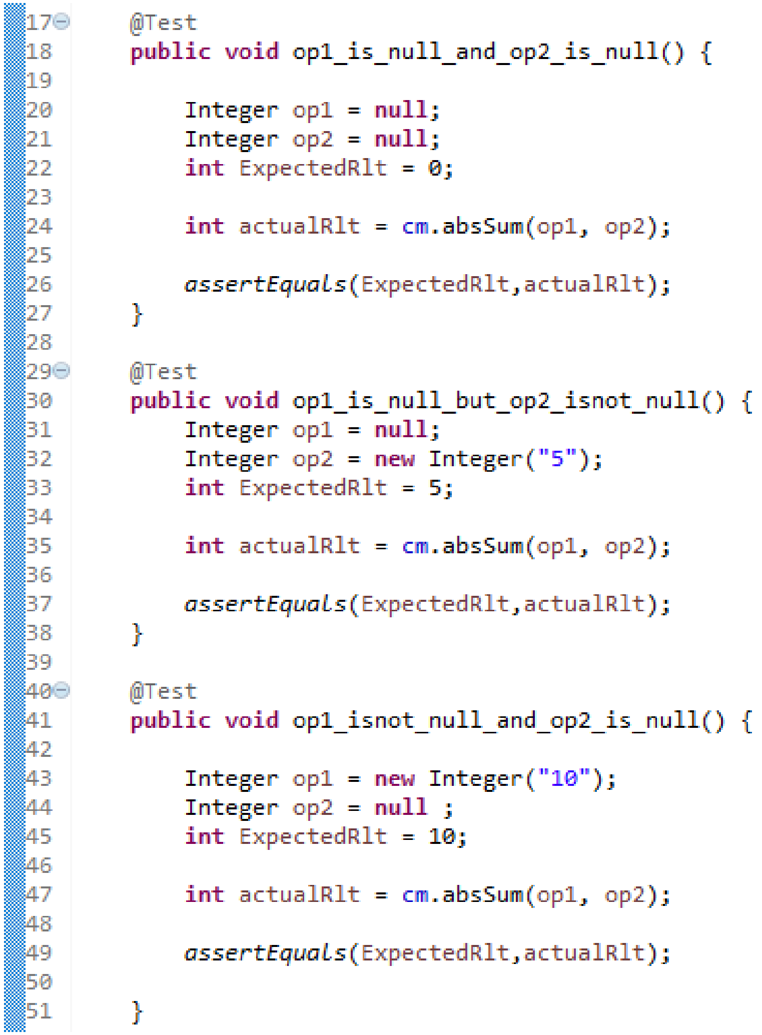


```
@Test
void op1_is_null_and_op2_is_null() {
    Integer op1 = null;
    Integer op2 = null;
    int actual_result = cm.absSum(op1, op2);
    assertEquals( expected: 0, actual_result);
}

@Test
void op1_is_null_and_op2_isnot_null() {
    Integer op1 = null;
    Integer op2 = 10;
    int actual_result = cm.absSum(op1, op2);
    assertEquals( expected: 10, actual_result);
}
```



100% classes, 87% lines covered in 'all classes in scope'

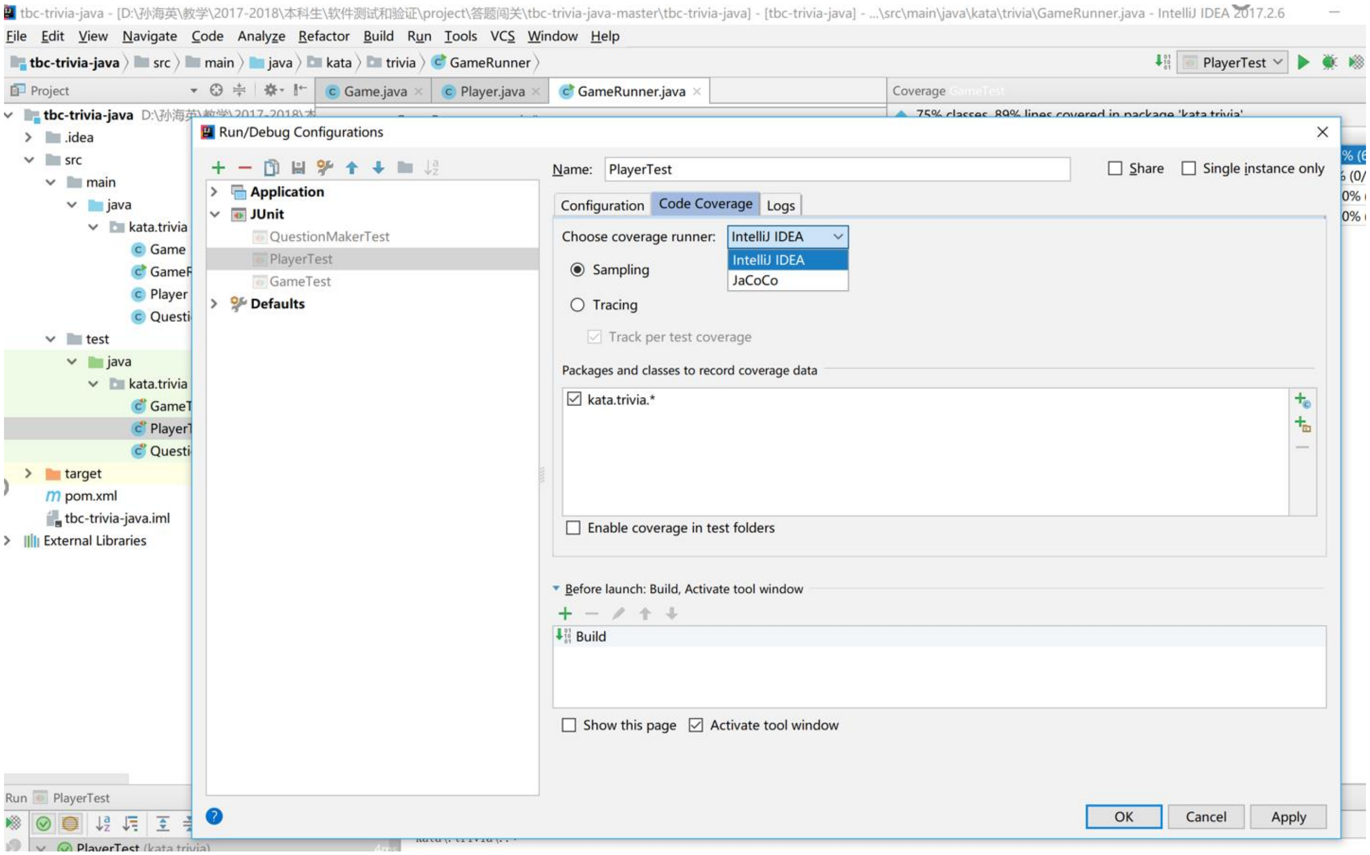
Element	Class, %	Method, %	Line, %	Branch, %
edu.ecnu.sei.st2017	100% (1/1)	100% (1/1)	87% (7/8)	60% (3/5)



```
    @Test
    public void op1_is_null_and_op2_is_null() {

        Integer op1 = null;
        Integer op2 = null;
        int ExpectedRlt = 0;

        int actualRlt = cm.absSum(op1, op2);

        assertEquals(ExpectedRlt,actualRlt);
    }

    @Test
    public void op1_is_null_but_op2_isnot_null() {
        Integer op1 = null;
        Integer op2 = new Integer("5");
        int ExpectedRlt = 5;

        int actualRlt = cm.absSum(op1, op2);

        assertEquals(ExpectedRlt,actualRlt);
    }

    @Test
    public void op1_isnot_null_and_op2_is_null() {

        Integer op1 = new Integer("10");
        Integer op2 = null ;
        int ExpectedRlt = 10;

        int actualRlt = cm.absSum(op1, op2);

        assertEquals(ExpectedRlt,actualRlt);

    }
```

Jacoco in IDEA

- Run->edit Configurations...->code coverage Tab



tbc-trivia-java - [D:\孙海英\教学\2017-2018\本科生\软件测试和验证\project\答题闯关\tbc-trivia-java-master\tbc-trivia-java] - [tbc-trivia-java] - ...\src\main\java\kata\trivia\GameRunner.java - IntelliJ IDEA 2017.2.6

Run/Debug Configurations

- Application
- JUnit
  - QuestionMakerTest
  - PlayerTest
  - GameTest
- Defaults

Name: PlayerTest

□ Share □ Single instance only

Configuration | Code Coverage | Logs

Choose coverage runner: IntelliJ IDEA (IntelliJ IDEA, JaCoCo)

● Sampling

○ Tracing

☑ Track per test coverage

Packages and classes to record coverage data

☑ kata.trivia.*

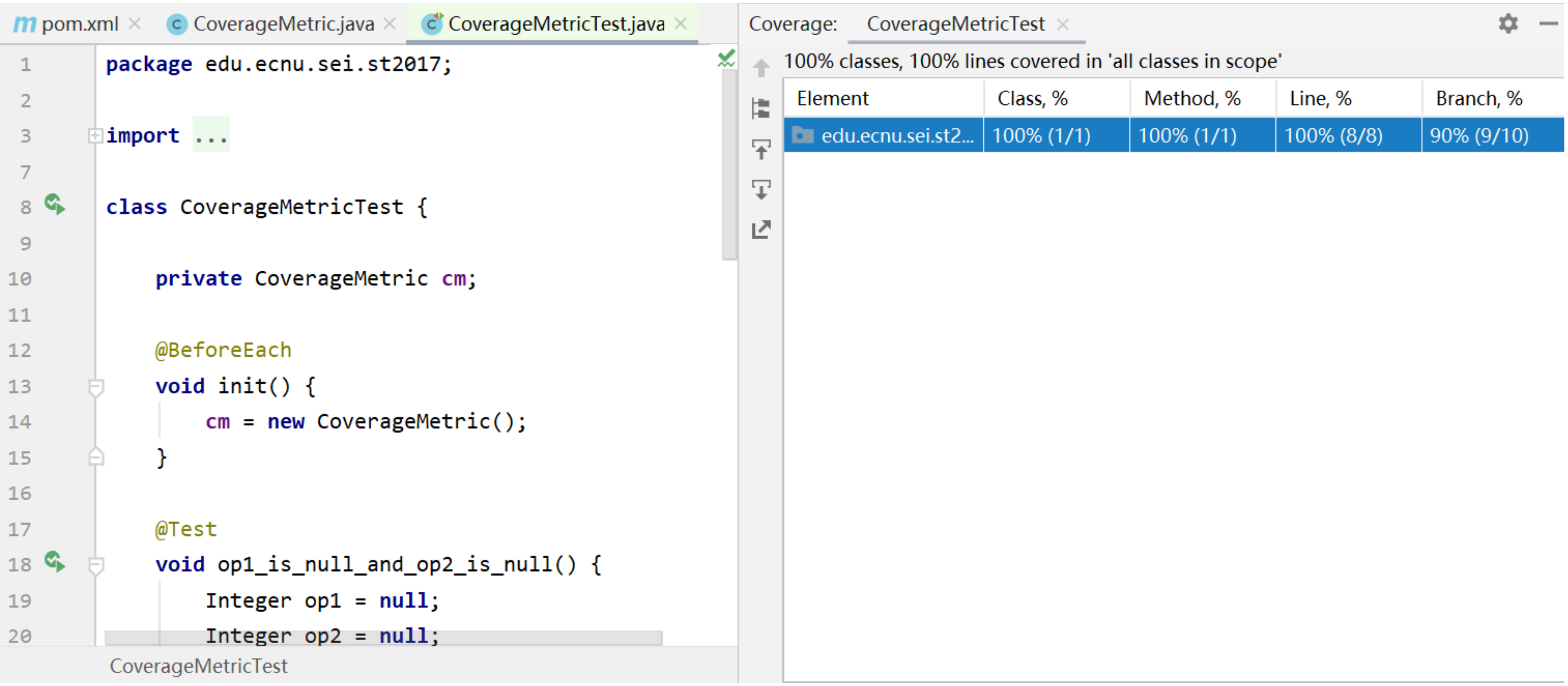
□ Enable coverage in test folders

Before launch: Build, Activate tool window

Build

□ Show this page ☑ Activate tool window

JaCoCo in IDEA



```
package edu.ecnu.sei.st2017;

import ...

class CoverageMetricTest {

    private CoverageMetric cm;

    @BeforeEach
    void init() {
        cm = new CoverageMetric();
    }

    @Test
    void op1_is_null_and_op2_is_null() {
        Integer op1 = null;
        Integer op2 = null;
```

Coverage: CoverageMetricTest

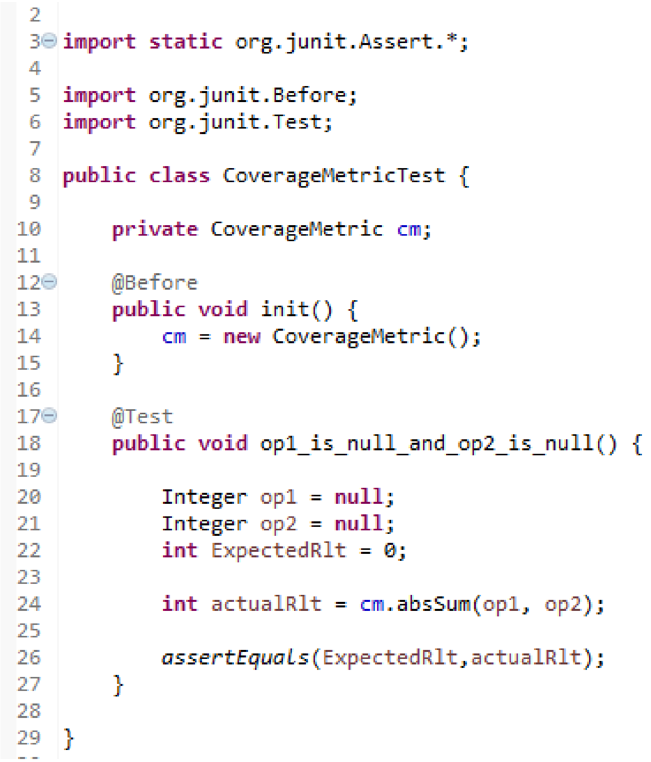
100% classes, 100% lines covered in 'all classes in scope'

Element	Class, %	Method, %	Line, %	Branch, %
edu.ecnu.sei.st2...	100% (1/1)	100% (1/1)	100% (8/8)	90% (9/10)

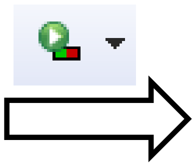
Eclipse的Eclemma

- 使用步骤
  - Eclipse Marketplace中安装eclEmma
  - 设计测试用例，编写测试类
  - 以覆盖形式运行测试类
  - 检查覆盖度是否达到要求，如果没有达到要求，则补充测试用例，再次运行测试，直到满足期望的覆盖要求为止

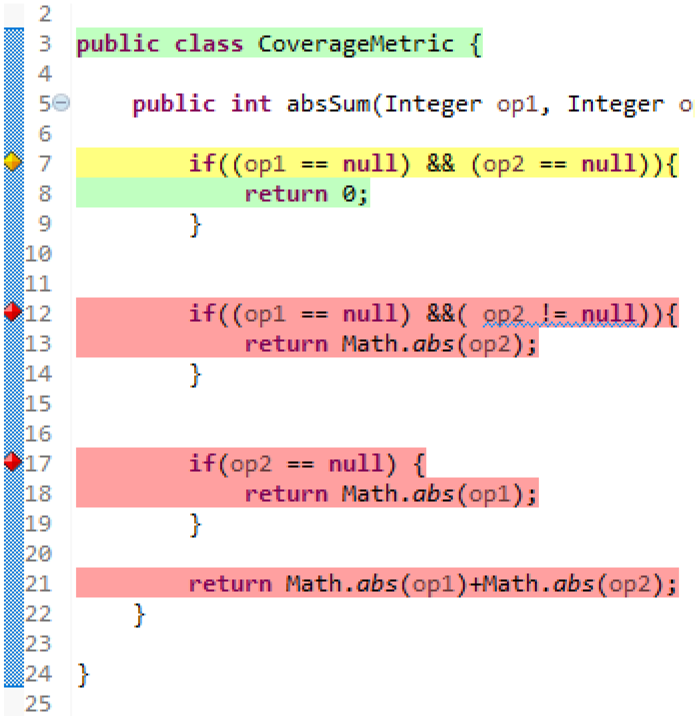
Use Eclemma



```
import static org.junit.Assert.*;

import org.junit.Before;
import org.junit.Test;

public class CoverageMetricTest {

    private CoverageMetric cm;

    @Before
    public void init() {
        cm = new CoverageMetric();
    }

    @Test
    public void op1_is_null_and_op2_is_null() {

        Integer op1 = null;
        Integer op2 = null;
        int ExpectedRlt = 0;

        int actualRlt = cm.absSum(op1, op2);

        assertEquals(ExpectedRlt,actualRlt);
    }

}
```



以覆盖运行测试

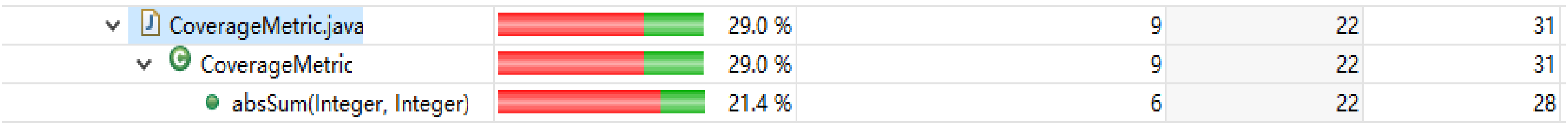


```
public class CoverageMetric {

    public int absSum(Integer op1, Integer op2) {

        if((op1 == null) && (op2 == null)){
            return 0;
        }


        if((op1 == null) &&( op2 != null)){
            return Math.abs(op2);
        }


        if(op2 == null) {
            return Math.abs(op1);
        }

        return Math.abs(op1)+Math.abs(op2);
    }

}
```

红色背景：没有指令被执行的代码 **红色菱形：**没有被执行的分支

黄色背景：部分指令被执行的代码 **黄色菱形：**部分被执行的分支

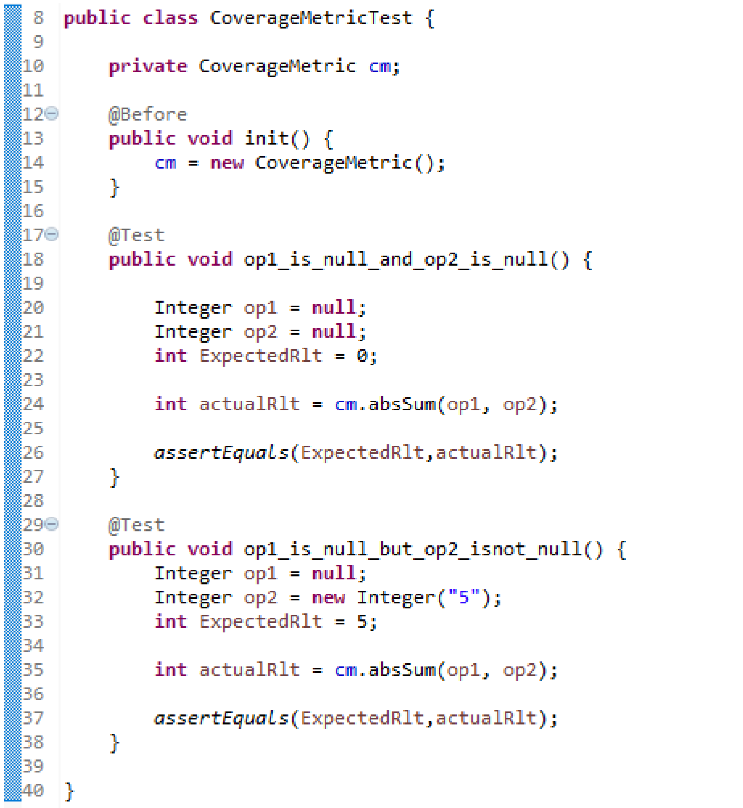
绿色背景：全部指令被执行的代码 **绿色菱形：**全部被执行的分支



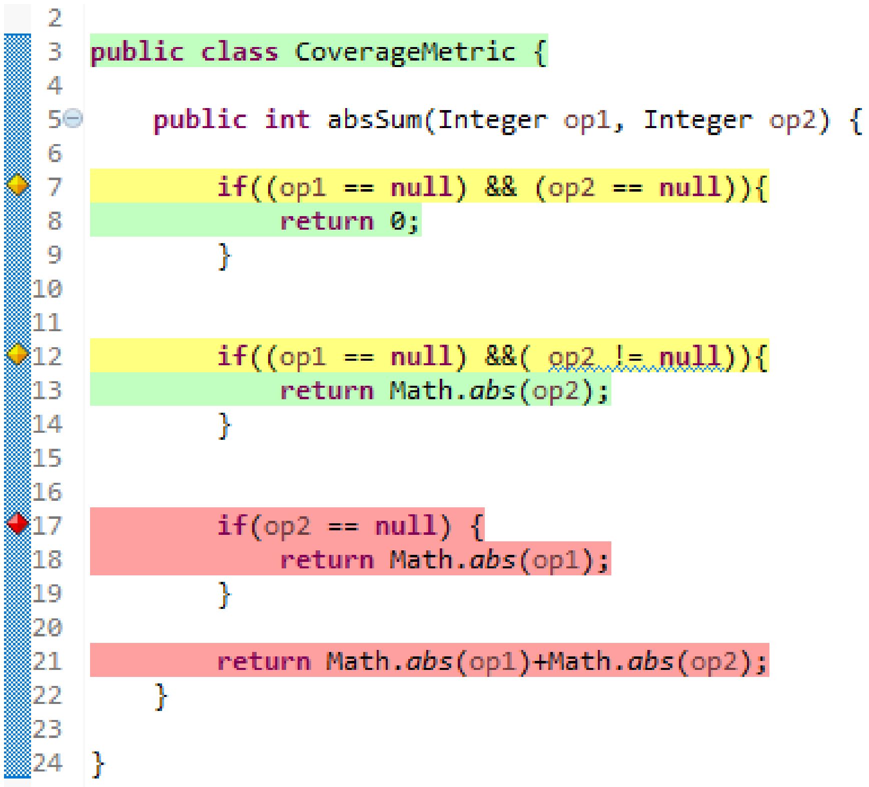
CoverageMetric.java	🟥🟩	29.0 %	9	22	31
CoverageMetric	🟥🟩	29.0 %	9	22	31
absSum(Integer, Integer)	🟥🟩	21.4 %	6	22	28

absSum输入**op1**为**null**，**op2**为**null**的指令覆盖度

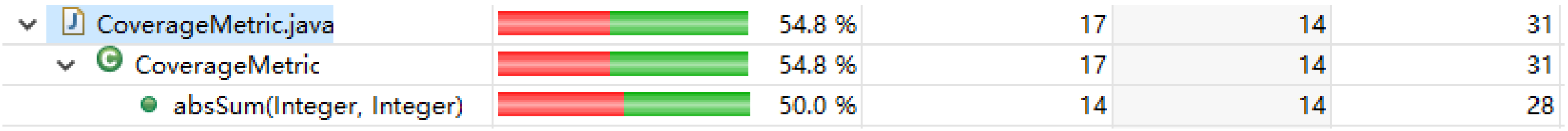
Use Eclemma



```java
public class CoverageMetricTest {

    private CoverageMetric cm;

    @Before
    public void init() {
        cm = new CoverageMetric();
    }

    @Test
    public void op1_is_null_and_op2_is_null() {

        Integer op1 = null;
        Integer op2 = null;
        int ExpectedRlt = 0;

        int actualRlt = cm.absSum(op1, op2);

        assertEquals(ExpectedRlt,actualRlt);
    }

    @Test
    public void op1_is_null_but_op2_isnot_null() {
        Integer op1 = null;
        Integer op2 = new Integer("5");
        int ExpectedRlt = 5;

        int actualRlt = cm.absSum(op1, op2);

        assertEquals(ExpectedRlt,actualRlt);
    }

}
```

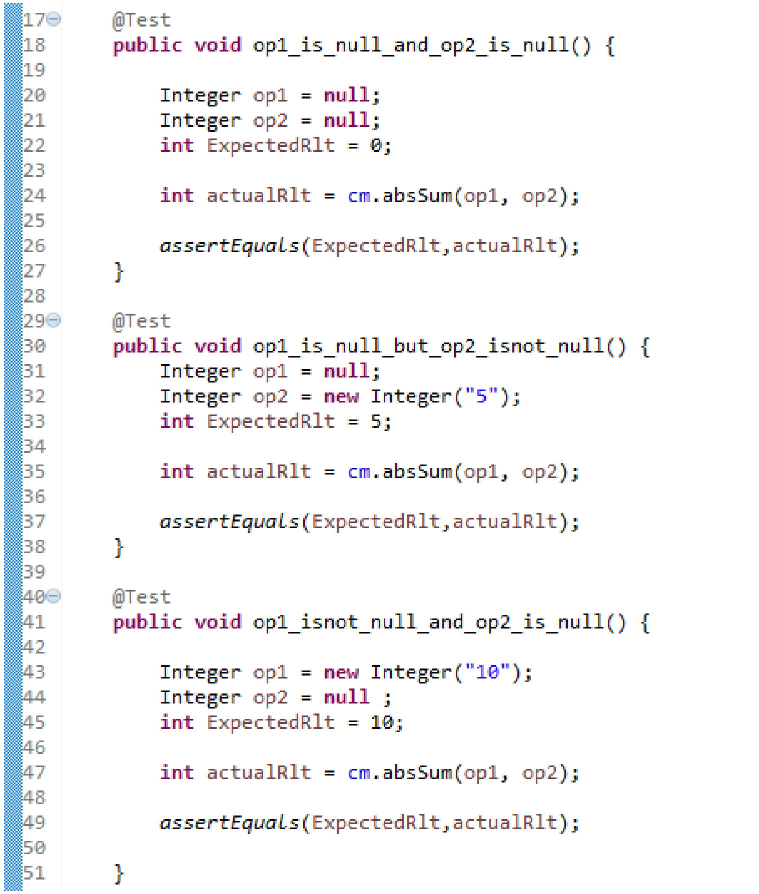


```java

public class CoverageMetric {

    public int absSum(Integer op1, Integer op2) {

        if((op1 == null) && (op2 == null)){
            return 0;
        }


        if((op1 == null) &&( op2 != null)){
            return Math.abs(op2);
        }


        if(op2 == null) {
            return Math.abs(op1);
        }

        return Math.abs(op1)+Math.abs(op2);
    }

}
```

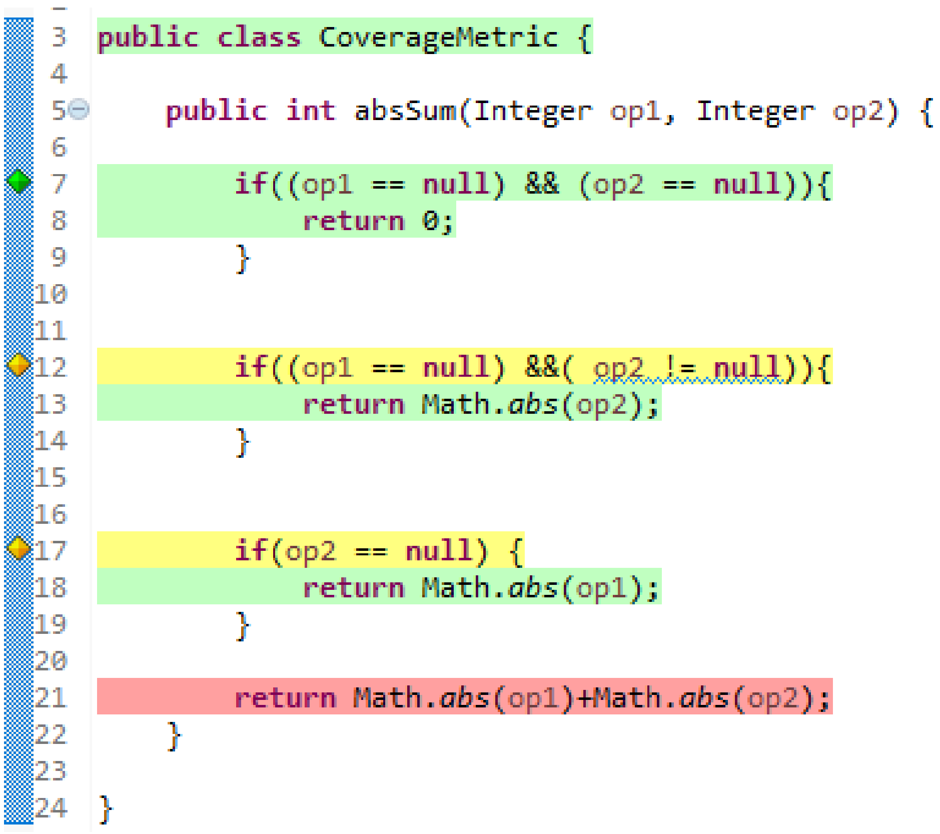


CoverageMetric.java	🟥🟩	54.8 %	17	14	31
CoverageMetric	🟥🟩	54.8 %	17	14	31
absSum(Integer, Integer)	🟥🟩	50.0 %	14	14	28

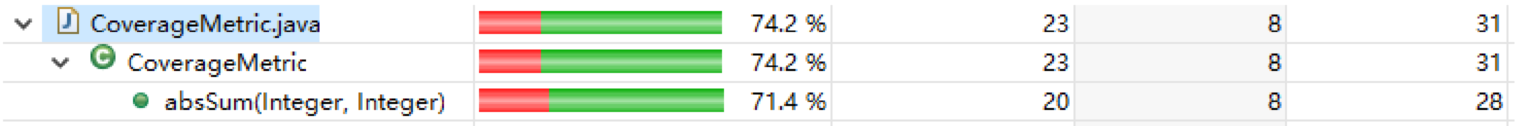
Use Eclemma



```java
    @Test
    public void op1_is_null_and_op2_is_null() {

        Integer op1 = null;
        Integer op2 = null;
        int ExpectedRlt = 0;

        int actualRlt = cm.absSum(op1, op2);

        assertEquals(ExpectedRlt,actualRlt);
    }

    @Test
    public void op1_is_null_but_op2_isnot_null() {
        Integer op1 = null;
        Integer op2 = new Integer("5");
        int ExpectedRlt = 5;

        int actualRlt = cm.absSum(op1, op2);

        assertEquals(ExpectedRlt,actualRlt);
    }

    @Test
    public void op1_isnot_null_and_op2_is_null() {

        Integer op1 = new Integer("10");
        Integer op2 = null ;
        int ExpectedRlt = 10;

        int actualRlt = cm.absSum(op1, op2);

        assertEquals(ExpectedRlt,actualRlt);

    }
```

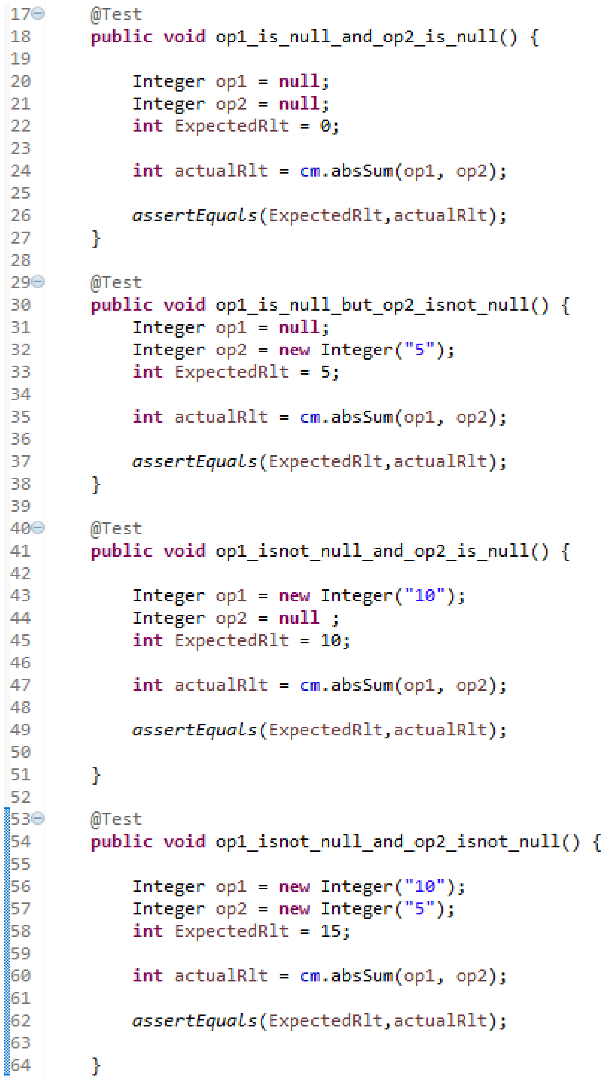


```java
public class CoverageMetric {

    public int absSum(Integer op1, Integer op2) {

        if((op1 == null) && (op2 == null)){
            return 0;
        }


        if((op1 == null) &&( op2 != null)){
            return Math.abs(op2);
        }


        if(op2 == null) {
            return Math.abs(op1);
        }

        return Math.abs(op1)+Math.abs(op2);
    }

}
```

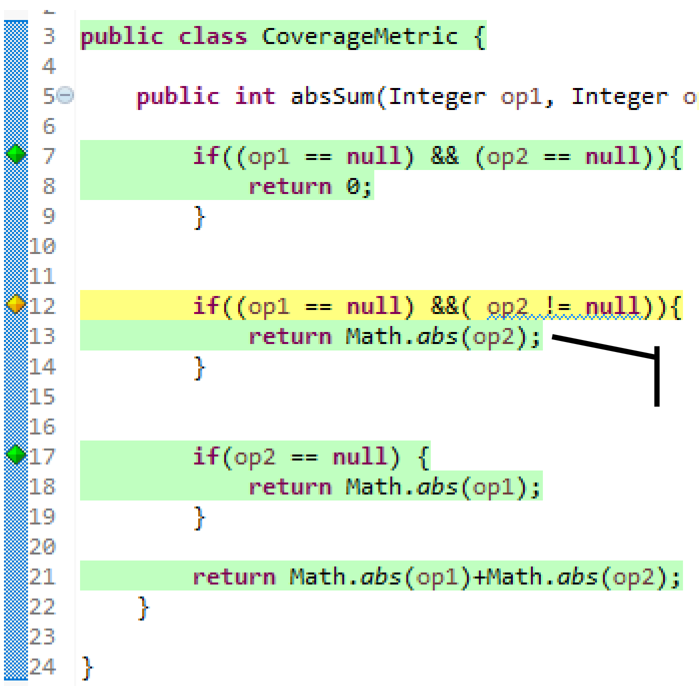


CoverageMetric.java	🟥🟩	74.2 %	23	8	31
CoverageMetric	🟥🟩	74.2 %	23	8	31
absSum(Integer, Integer)	🟥🟩	71.4 %	20	8	28

Use Eclemma



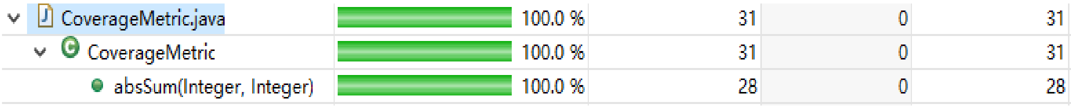
```java
    @Test
    public void op1_is_null_and_op2_is_null() {

        Integer op1 = null;
        Integer op2 = null;
        int ExpectedRlt = 0;

        int actualRlt = cm.absSum(op1, op2);

        assertEquals(ExpectedRlt,actualRlt);
    }

    @Test
    public void op1_is_null_but_op2_isnot_null() {
        Integer op1 = null;
        Integer op2 = new Integer("5");
        int ExpectedRlt = 5;

        int actualRlt = cm.absSum(op1, op2);

        assertEquals(ExpectedRlt,actualRlt);
    }

    @Test
    public void op1_isnot_null_and_op2_is_null() {

        Integer op1 = new Integer("10");
        Integer op2 = null ;
        int ExpectedRlt = 10;

        int actualRlt = cm.absSum(op1, op2);

        assertEquals(ExpectedRlt,actualRlt);

    }

    @Test
    public void op1_isnot_null_and_op2_isnot_null() {

        Integer op1 = new Integer("10");
        Integer op2 = new Integer("5");
        int ExpectedRlt = 15;

        int actualRlt = cm.absSum(op1, op2);

        assertEquals(ExpectedRlt,actualRlt);

    }
```



```java
public class CoverageMetric {

    public int absSum(Integer op1, Integer op2) {

        if((op1 == null) && (op2 == null)){
            return 0;
        }


        if((op1 == null) &&( op2 != null)){
            return Math.abs(op2);
        }


        if(op2 == null) {
            return Math.abs(op1);
        }

        return Math.abs(op1)+Math.abs(op2);
    }

}
```

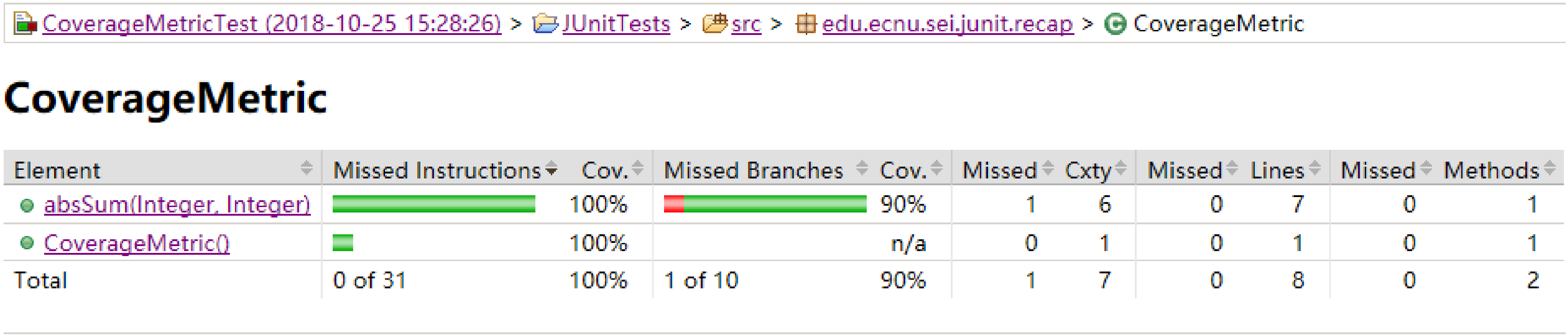
1 of 4 branches missed

Which branch is missed ?



CoverageMetric.java	🟩	100.0 %	31	0	31
CoverageMetric	🟩	100.0 %	31	0	31
absSum(Integer, Integer)	🟩	100.0 %	28	0	28

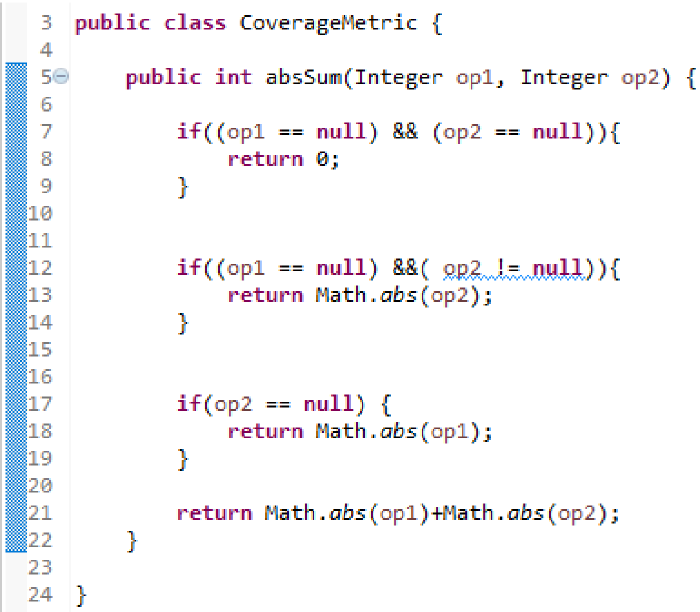
Use Eclemma



CoverageMetricTest (2018-10-25 15:28:26) > JUnitTests > src > edu.ecnu.sei.junit.recap > CoverageMetric

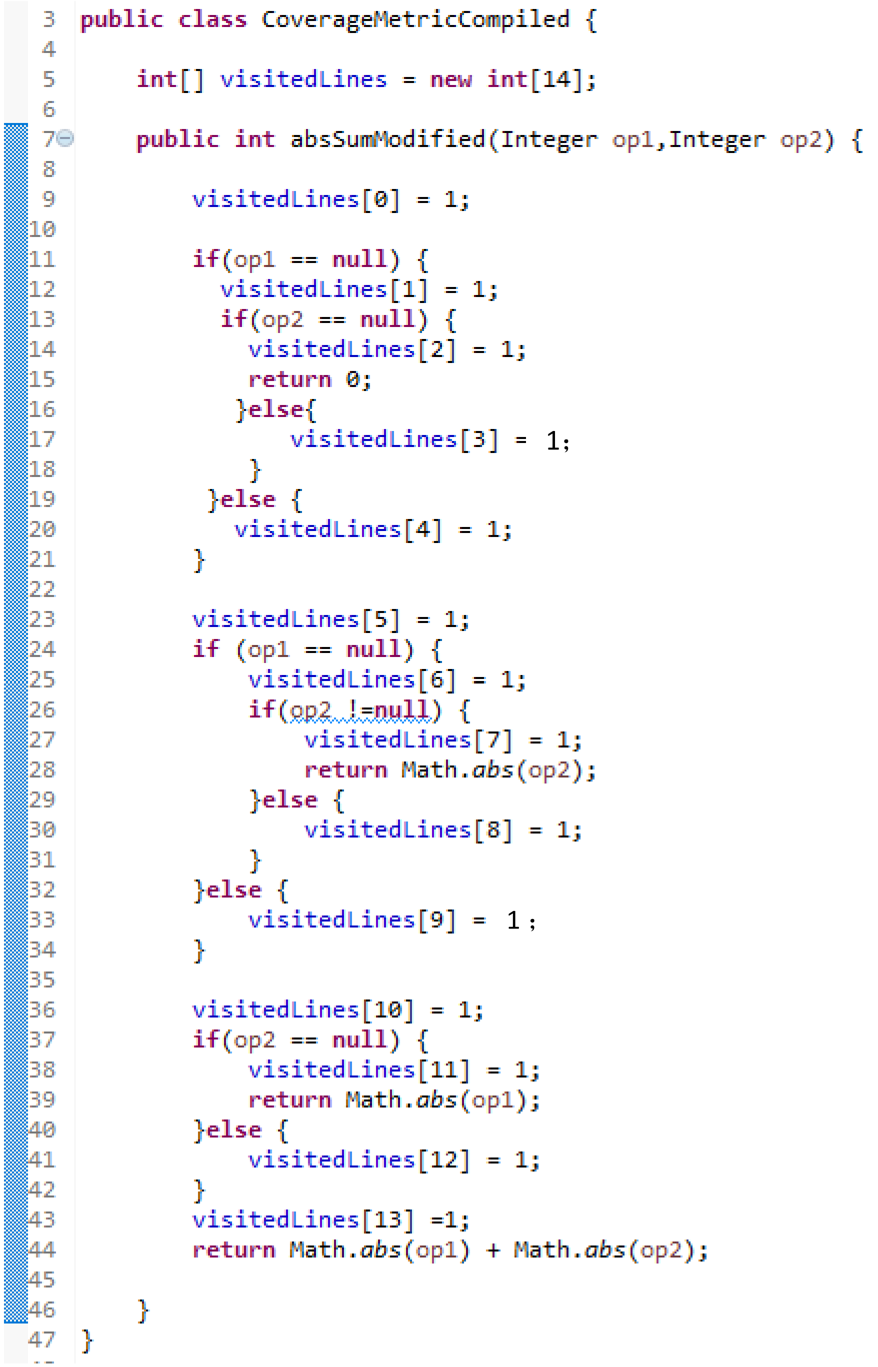
CoverageMetric

Element	Missed Instructions	Cov.	Missed Branches	Cov.	Missed	Cxty	Missed	Lines	Missed	Methods
● absSum(Integer, Integer)	🟩	100%	🟥🟩	90%	1	6	0	7	0	1
● CoverageMetric()	🟩	100%		n/a	0	1	0	1	0	1
Total	0 of 31	100%	1 of 10	90%	1	7	0	8	0	2



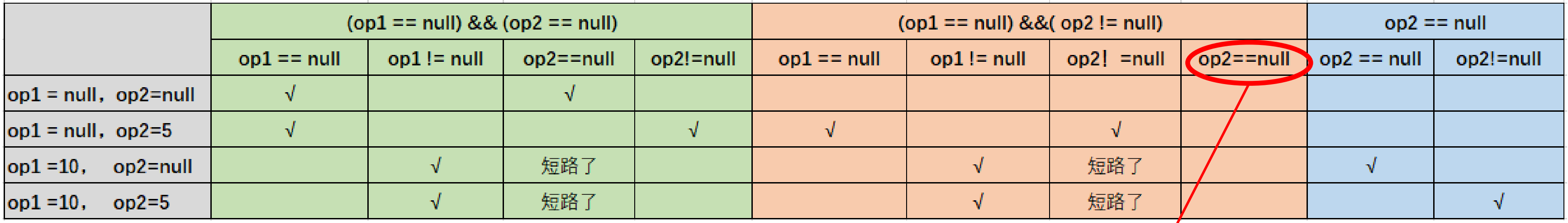
```
public class CoverageMetric {

    public int absSum(Integer op1, Integer op2) {

        if((op1 == null) && (op2 == null)){
            return 0;
        }


        if((op1 == null) &&( op2 != null)){
            return Math.abs(op2);
        }


        if(op2 == null) {
            return Math.abs(op1);
        }

        return Math.abs(op1)+Math.abs(op2);
    }

}
```

为收集覆盖度，插桩之后的代码

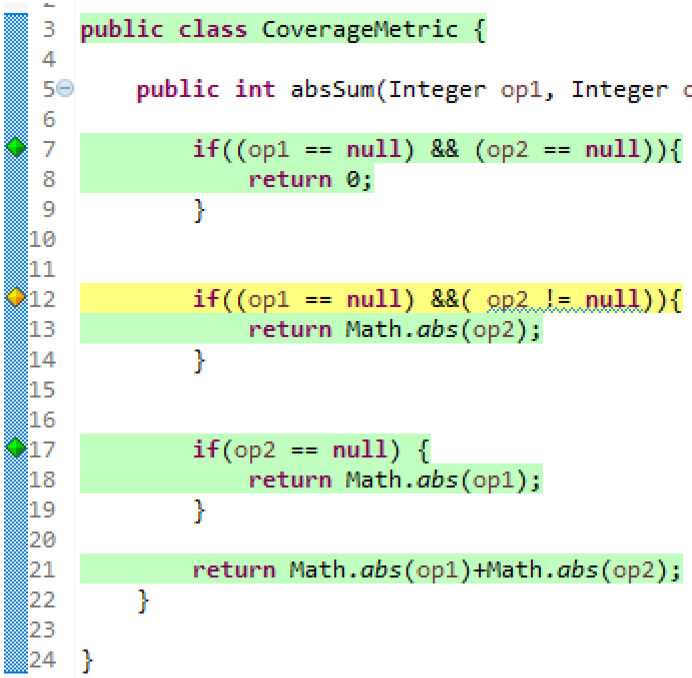


```
public class CoverageMetricCompiled {

    int[] visitedLines = new int[14];

    public int absSumModified(Integer op1,Integer op2) {

        visitedLines[0] = 1;

        if(op1 == null) {
          visitedLines[1] = 1;
          if(op2 == null) {
            visitedLines[2] = 1;
            return 0;
           }else{
               visitedLines[3] = 1;
            }
         }else {
           visitedLines[4] = 1;
        }

        visitedLines[5] = 1;
        if (op1 == null) {
            visitedLines[6] = 1;
            if(op2 !=null) {
                visitedLines[7] = 1;
                return Math.abs(op2);
            }else {
                visitedLines[8] = 1;
            }
        }else {
            visitedLines[9] = 1;
        }

        visitedLines[10] = 1;
        if(op2 == null) {
            visitedLines[11] = 1;
            return Math.abs(op1);
        }else {
            visitedLines[12] = 1;
        }
        visitedLines[13] =1;
        return Math.abs(op1) + Math.abs(op2);

    }
}
```

Understanding the details



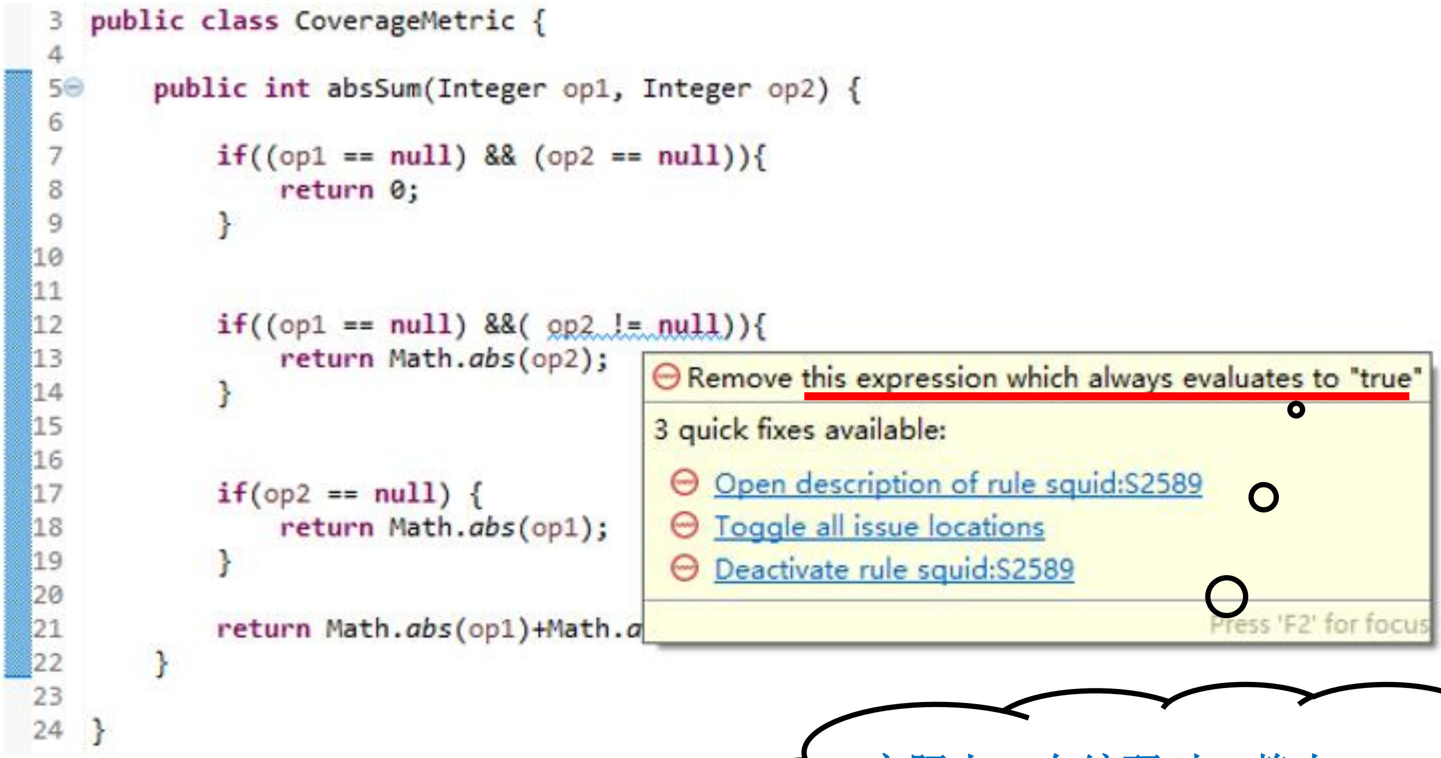
	(op1 == null) && (op2 == null)				(op1 == null) &&(op2 != null)				op2 == null	
	op1 == null	op1 != null	op2==null	op2!=null	op1 == null	op1 != null	op2! =null	op2==null	op2 == null	op2!=null
op1 = null，op2=null	√		√							
op1 = null，op2=5	√			√	√		√			
op1 =10， op2=null		√	短路了			√	短路了		√	
op1 =10， op2=5		√	短路了			√	短路了			√



```
public class CoverageMetric {

    public int absSum(Integer op1, Integer op2) {

        if((op1 == null) && (op2 == null)){
            return 0;
        }


        if((op1 == null) &&( op2 != null)){
            return Math.abs(op2);
        }


        if(op2 == null) {
            return Math.abs(op1);
        }

        return Math.abs(op1)+Math.abs(op2);
    }

}
```

1 of 4 branches missed

Understanding the details



```
public class CoverageMetric {

    public int absSum(Integer op1, Integer op2) {

        if((op1 == null) && (op2 == null)){
            return 0;
        }


        if((op1 == null) &&( op2 != null)){
            return Math.abs(op2);
        }


        if(op2 == null) {
            return Math.abs(op1);
        }

        return Math.abs(op1)+Math.a
    }

}
```

Remove this expression which always evaluates to "true"

3 quick fixes available:

- Open description of rule squid:S2589
- Toggle all issue locations
- Deactivate rule squid:S2589

Press 'F2' for focus

实际上，在编码时，静态分析器已经给出提示了

Summary

- Unit is defined by function and size
- Unit testing is to write code to test code which executed in very short time
- Logical code coverage criteria are intended to detect logical bugs
- Different coverage criterion has different defect-detective ability
- Statement coverage is the weakest while Multiple coverage is the strongest but need more test cases
- Coverage tools are the practical implementation of logical coverage criteria theory. Because of different coverage data collection strategies, one should check the tools coverage definitions before using them.

The End