화이트박스 테스팅

Code Coverage 분석의 필요성

❖Black box 테스트의 취약점

함수 이름	getMax	
매개변수	int values[]	int 값에 대한 배열
	int size	values 배열의 크기
반환 값	int	values 중에서 최대값
기능 명세	매개변수로 주어진 values 중에서 최대값을 반환한다.	

Code Coverage 분석의 필요성

❖ Black box 테스트 케이스

식별자	설명	입력 값		예상 결과
		values[]	size	반환 값
TC-1	1번째 원소가 최대값인 경우	20, 3, 5, 7	4	20
TC-2	2번째 원소가 최대값인 경우	2, 25, 9, 8	4	25
TC-3	3번째 원소가 최대값인 경우	7, 6, 10, 8	4	10
TC-4	4번째 원소가 최대값인 경우	6, 5, 1, 9	4	9

Code Coverage 분석의 필요성

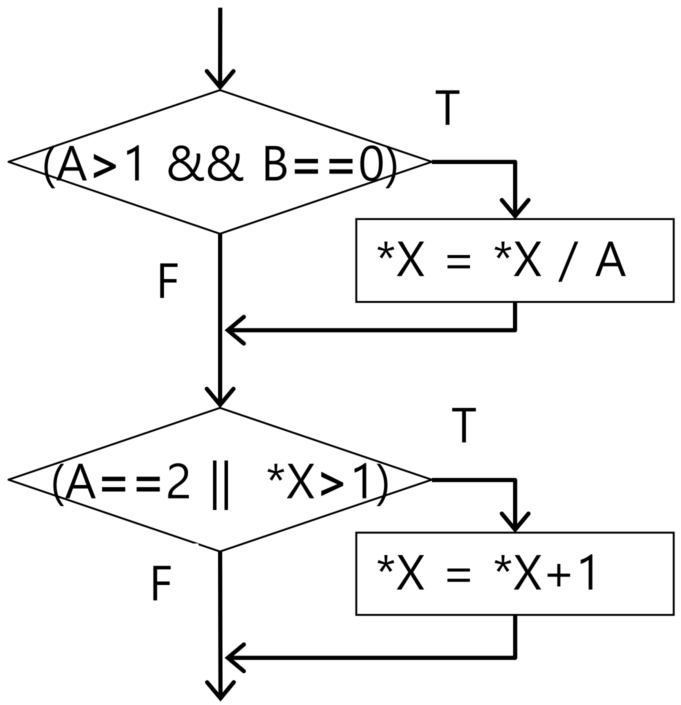
	코드	TC-1	TC-2	TC-3	TC-4
	int getMax(int values[], int size) {				
1	int max = values[0] ;	√	√	√	√
2	if (size == 1)	√	√	√	√
3	return max ;				
4	if (size == 2) {	√	√	√	√
5	if (max >= values[1])				
6	return max ;				
7	else				
8	return values[1] ;				
	}				
9	for (int i = 1 ; i < size ; i ++)	√	√	√	√
10	if (max < values[i])	√	√	√	√
11	max = values[i] ;	√	√	√	√
12	return max ;	√	√	√	√
	}				

Code Coverage 분석의 필요성

	코드	TC-5 {10}, 1	TC-6 {2, 2}, 2	TC-7 {1, 2}, 2
	`int getMax(int values[], int size) {`			
1	`  int max = values[0] ;`	√	√	√
2	`  if ( size == 1 )`	√	√	√
3	`    return max ;`	√		
4	`  if ( size == 2 ) {`		√	√
5	`    if ( max >= values[1] )`		√	√
6	`      return max ;`		√	
7	`    else`			√
8	`      return values[1] ;`			√
	`  }`			
9	`  for ( int i = 1 ; i < size ; i ++ )`			
10	`    if ( max > values[i] )`			
11	`      max = values[i] ;`			
12	`  return max ;`			
	`}`			

Example Code

```
void F(int A, int B, int* X) {
    if ( A > 1 && B == 0 )
        *X = *X / A;
    if ( A == 2 ||  *X > 1 )
        *X = *X + 1;
}
```

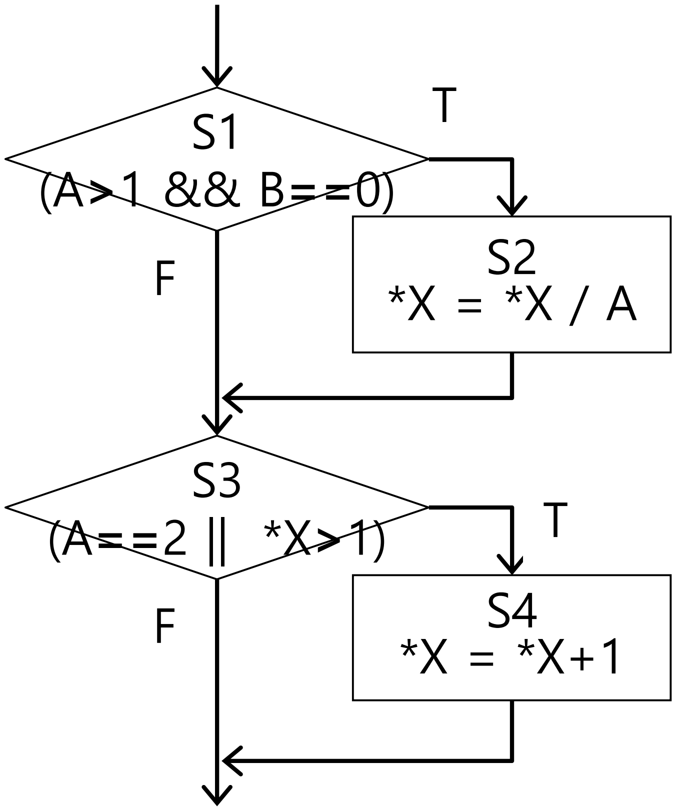


Statement Coverage

❖ The percentage of the statements exercised by the test suite

❖ e.g.)

	TC1			TC2		
	A	B	*X	A	B	*X
	3	0	1	2	0	3
S1	√			√		
S2	√			√		
S3	√			√		
S4				√		
	3 / 4			4 / 4		
	4 / 4					



Exercise

	코드	TC-1		TC-2	
		{1, 2}	2	{20, 3, 5, 7}	4
	`int getMax(int values[], int size) {`				
1	`int max = values[0] ;`	√		√	
2	`if ( size == 1 )`	√		√	
3	`return max ;`				
4	`if ( size == 2 ) {`	√		√	
5	`if ( max >= values[1] )`	√			
6	`return max ;`				
	`else`				
7	`return values[1] ;`	√			
	`}`				
8	`for ( int i = 1 ; i < size ; i ++ )`			√	
9	`if ( max < values[i] )`			√	
10	`max = values[i] ;`				
11	`return max ;`			√	
	`}`				
Statement coverage		5 / 11		6 / 11	
		8 / 11			

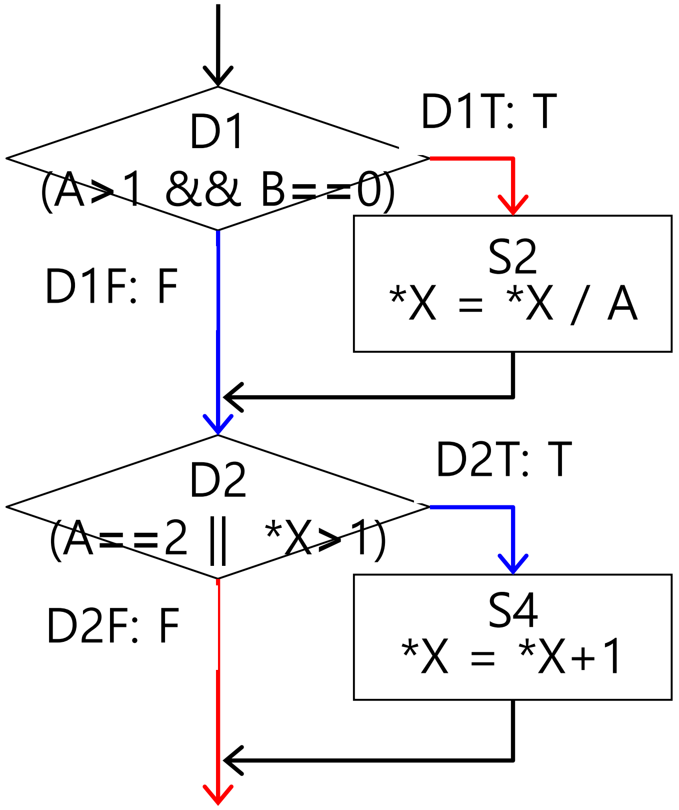
Practice Code 1 – Premium Change #1

	코드	TC1		TC2	
		ag	ac	ag	ac
		21	2	18	3
	`int calculate1(int age, int accident) {`				
1	`cout << age << " " << accident << endl;`	√		√	
2	`int rateChange = 0 ;`	√		√	
3	`if ( accident <= 2 )`	√		√	
4	`rateChange = 5 ;`	√			
	`else`				
5	`rateChange = 10 ;`			√	
6	`if ( accident >= 3 || age <= 20 )`	√		√	
7	`rateChange += 5 ;`			√	
8	`cout << rateChange << endl ;`	√		√	
9	`return rateChange ;`	√		√	
	`}`				
Statement coverage		7 / 9		8 / 9	
		9 / 9			

Branch Coverage (Decision Coverage)

❖ The percentage of branches exercised by the test suite

	TC1			TC2		
	A	B	*X	A	B	*X
	3	0	3	2	1	1
D1T	√					
D1F				√		
D2T				√		
D2F	√					
	2 / 4			2 / 4		
	4 / 4					



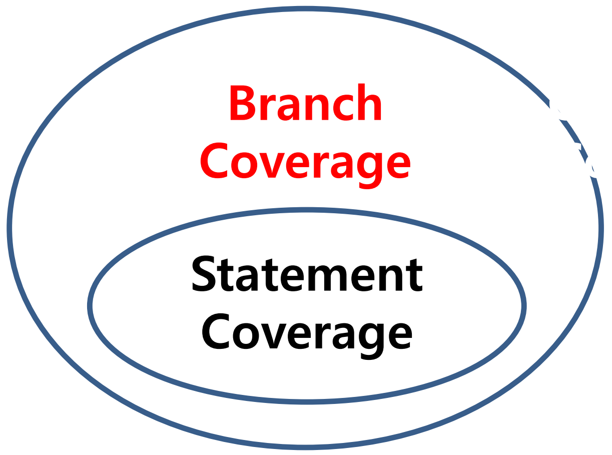
Exercise

	코드	TC-1		TC-2	
		{1, 2}	2	{20, 3, 5, 7}	4
	`int getMax(int values[], int size) {`				
	`  int max = values[0] ;`				
D1	`  if ( size == 1 )`	D1F		D1F	
	`    return max ;`				
D2	`  if ( size == 2 ) {`	D2T		D2F	
D3	`    if ( max >= values[1] )`	D3F			
	`      return max ;`				
	`    else`				
	`      return values[1] ;`				
	`  }`				
D4	`  for ( int i = 1 ; i < size ; i ++ )`			D4T, D4F	
D5	`    if ( max < values[i] )`			D5F	
	`      max = values[i] ;`				
	`  return max ;`				
	`}`				
Branch coverage		3 / 10		5 / 10	
		7 / 10			

Practice Code 1 – Premium Change #1

	코드	TC1		TC2	
		ag	ac	ag	ac
		21	2	18	3
	`int calculate1(int age, int accident) {`				
	`  cout << age << " " << accident << endl;`				
	`  int rateChange = 0 ;`				
D1	`  if ( accident <= 2 )`	D1T		D1F	
	`     rateChange = 5 ;`				
	`  else`				
	`     rateChange = 10 ;`				
D2	`  if ( accident >= 3 || age <= 20 )`	D2F		D2T	
	`     rateChange += 5 ;`				
	`  cout << rateChange << endl ;`				
	`  return rateChange ;`				
	`}`				
Branch coverage		2 / 4		2 / 4	
		4 / 4			

Comparison



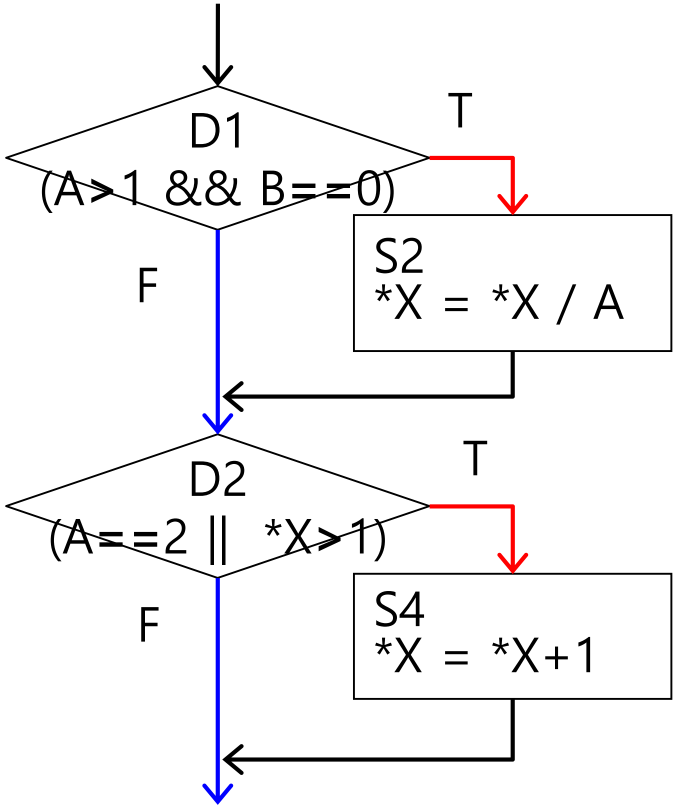
Condition Coverage

❖ The percentage of the conditions exercised by a test suite

	TC1	TC2

	A=2
	B=0
	*X=4
		A=1
		B=1
		*X=1

A > 1	T	F
B == 0	T	F
A == 2	T	F
*X > 1	T	F
	4 / 8	4 / 8
	8 / 8	



Practice Code 1 – Premium Change #1

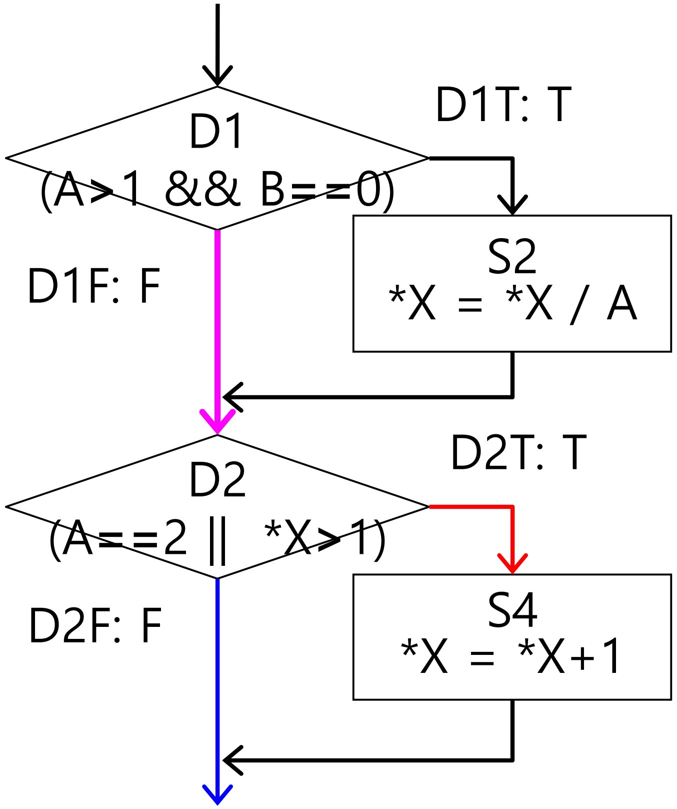
	코드	TC1		TC2	
		ag	ac	ag	ac
		20	1	21	3
	int calculate1(int age, int accident) {				
	cout << age << " " << accident << endl;				
	int rateChange = 0 ;				
C1	if (accident <= 2)	C1T		C1F	
	rateChange = 5 ;				
	else				
	rateChange = 10 ;				
C2, C3	if (accident >= 3 || age <= 20)	C2F, C3T		C2T, C3F	
	rateChange += 5 ;				
	cout << rateChange << endl ;				
	return rateChange ;				
	}				
Condition coverage		3 / 6		3 / 6	
		6 / 6			

Condition Coverage vs. Branch Coverage

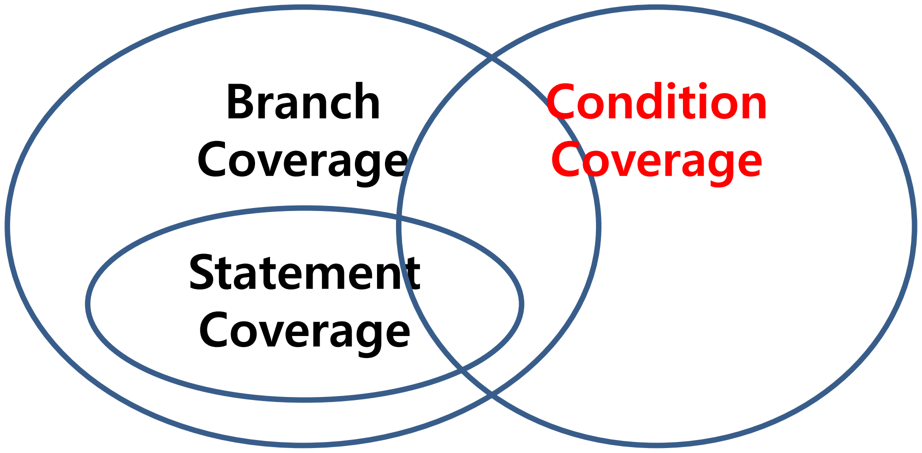
- Question: Does condition coverage imply branch coverage?
- Answer: Not always.

	TC1	TC2
	A=1 B=0 *X=3	A=2 B=1 *X=1
A > 1	F	T
B == 0	T	F
A == 2	F	T
*X > 1	T	F
	4 / 8	4 / 8
	8 / 8	

	TC1	TC2
	A=1 B=0 *X=3	A=2 B=1 *X=1
D1T		
D1F	√	√
D2T	√	
D2F		√
	2 / 4	2 / 4
	3 / 4	



Comparison



Condition/Decision Coverage

- The percentage of the conditions and decisions exercised by a test suite
- e.g.) 11 / 12

	TC1	TC2

	A=1
	B=0
	*X=3
		A=2
		B=1
		*X=1

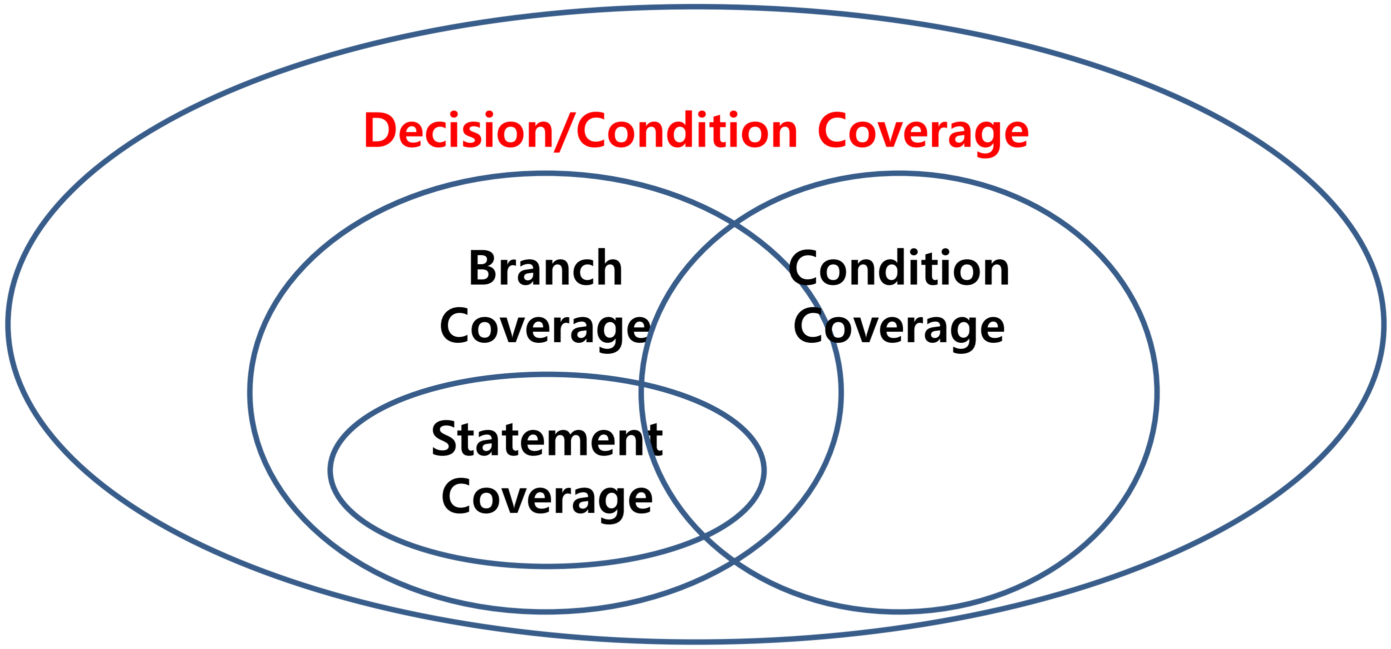
A > 1	F	T
B == 0	T	F
A == 2	F	T
*X > 1	T	F
	4 / 8	4 / 8
	8 / 8	

	TC1	TC2

	A=1
	B=0
	*X=3
		A=2
		B=1
		*X=1

D1T		
D1F	√	√
D2T	√	
D2F		√
	2 / 4	2 / 4
	3 / 4	

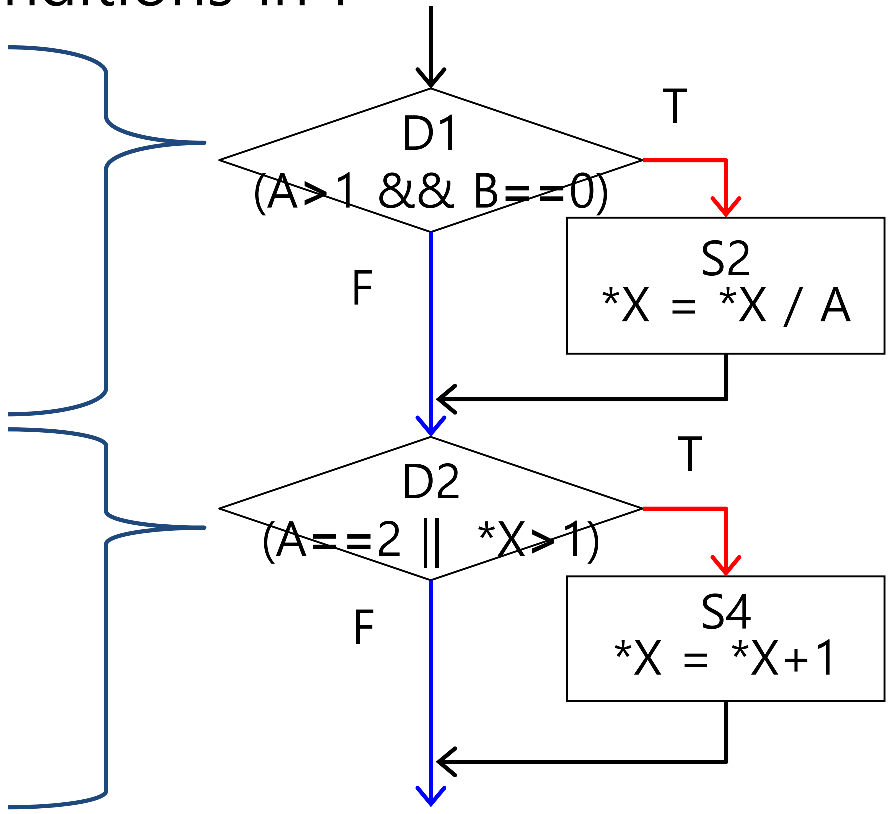
Comparison


Decision/Condition Coverage
Branch Coverage
Condition Coverage
Statement Coverage

Multiple-Condition Coverage

- The percentage of all combinations of each condition exercised by a test suite
- e.g.) combinations of conditions in F

1	A > 1, B = 0
2	A > 1, B ≠ 0
3	A ≤ 1, B = 0
4	A ≤ 1, B ≠ 0
5	A = 2, *X > 1
6	A = 2, *X ≤ 1
7	A ≠ 2, *X > 1
8	A ≠ 2, *X ≤ 1



Multiple-Condition Coverage (Compound-Condition Coverage)

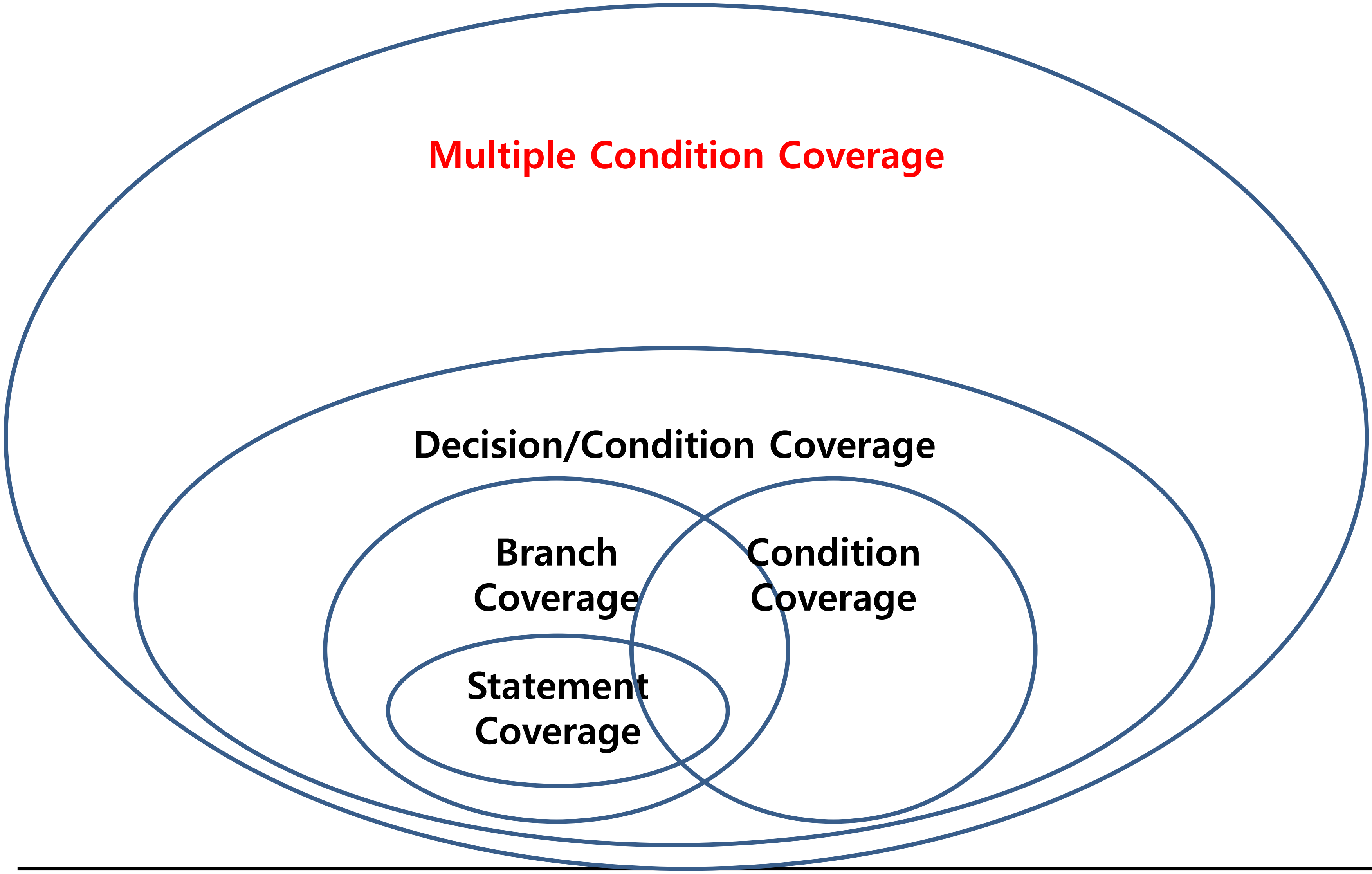
		A=2 B=0 *X=0	A=2 B=1 *X=1	A=1 B=0 *X=2	A=1 B=1 *X=1
1	A > 1, B = 0	√			
2	A > 1, B ≠ 0		√		
3	A ≤ 1, B = 0			√	
4	A ≤ 1, B ≠ 0				√
5	A = 2, *X > 1	√			
6	A = 2, *X ≤ 1		√		
7	A ≠ 2, *X > 1			√	
8	A ≠ 2, *X ≤ 1				√
Coverage		2 / 8	2 / 8	2 / 8	2 / 8
		8 / 8			

Multiple-Condition Coverage (Compound-Condition Coverage)

❖ Compound-Condition Coverage satisfies Decision/Condition Coverage

		A=2 B=0 *X=0	A=2 B=1 *X=1	A=1 B=0 *X=2	A=1 B=1 *X=1
Decision Coverage	A>1 && B==0	T	F	F	F
	A==2 || *X>1	T	T	F	F
Condition Coverage	A>1	T	T	F	F
	B==0	T	F	T	F
	A==2	T	T	F	F
	*X>1	F	F	T	F

Comparison


Multiple Condition Coverage
Decision/Condition Coverage
Branch Coverage
Condition Coverage
Statement Coverage

Practice Code 1 – Premium Change #1

```
int calculate1(int age, int accident) {
  cout << age << " " << accident << endl;
  int rateChange = 0 ;
  if ( accident <= 2 )
     rateChange = 5 ;
  else
     rateChange = 10 ;
  if ( accident >= 3 || age <= 20 )
     rateChange += 5 ;
  cout << rateChange << endl ;
  return rateChange ;
}
```

		ac= ag=	ac= ag=	ac= ag=	ac= ag=
1	ac<=2				
2	ac>2				
3	ac>=3, ag<=20				
4	ac>=3, ag>20				
5	ac<3, ag<=20				
6	ac<3, ag>20				
Coverage		/ 6	/ 6	/ 6	/ 6
		/ 6			

Practice Code 3 – Premium Change #2

```
int calculate2(int age, int accident, bool male) {
  int rateChange = 0 ;
  if ( accident <= 2 )
     rateChange = 5 ;
  else
     rateChange = 10 ;
  if ( accident >= 5 || (age <= 20 && male) )
     rateChange += 5 ;
  cout << rateChange << endl ;
  return rateChange ;
}
```

		ac= ag= m=	ac= ag= m=	ac= ag= m=	ac= ag= m=
1	ac<=2				
2	ac>2				
3	ac>=5, ag<=20, male				
4	ac>=5, ag<=20, !male				
5	ac>=5, ag>20, male				
6	ac>=5, ag>20, !male				
7	ac<5, ag<=20, male				
8	ac<5, ag<=20, !male				
9	ac<5, ag>20, male				
10	ac<5, ag>20, !male				

MCDC

- Multiple Condition Coverage requires 2^n test cases for *n* conditions.
- **Modified Condition Decision Coverag**e requires test cases to show that <u>each condition can independently affect the outcome of the decision</u>
- MCDC requires between n+1 and 2*n test cases

A	B	A and B
T	F	F
T	T	T
F	T	F
F	F	F

A	B	A or B
T	T	T
T	F	T
F	F	F
F	T	T

MCDC

- A and B and C
  - 1, 2, 3, 5
- A or B or C
  - 4, 6, 7, 8

TC	A	B	C	**A and B and C**				**A or B or C**			
				Outcome	A	B	C	Outcome	A	B	C
1	T	T	T	T	√	√	√	T			
2	T	T	F	F			√	T			
3	T	F	T	F		√		T			
4	T	F	F	F				T	√		
5	F	T	T	F	√			T			
6	F	T	F	F				T		√	
7	F	F	T	F				T			√
8	F	F	F	F				F	√	√	√

MCDC

- A or (B and C)
  - (2 or 3), 5, 6, 7

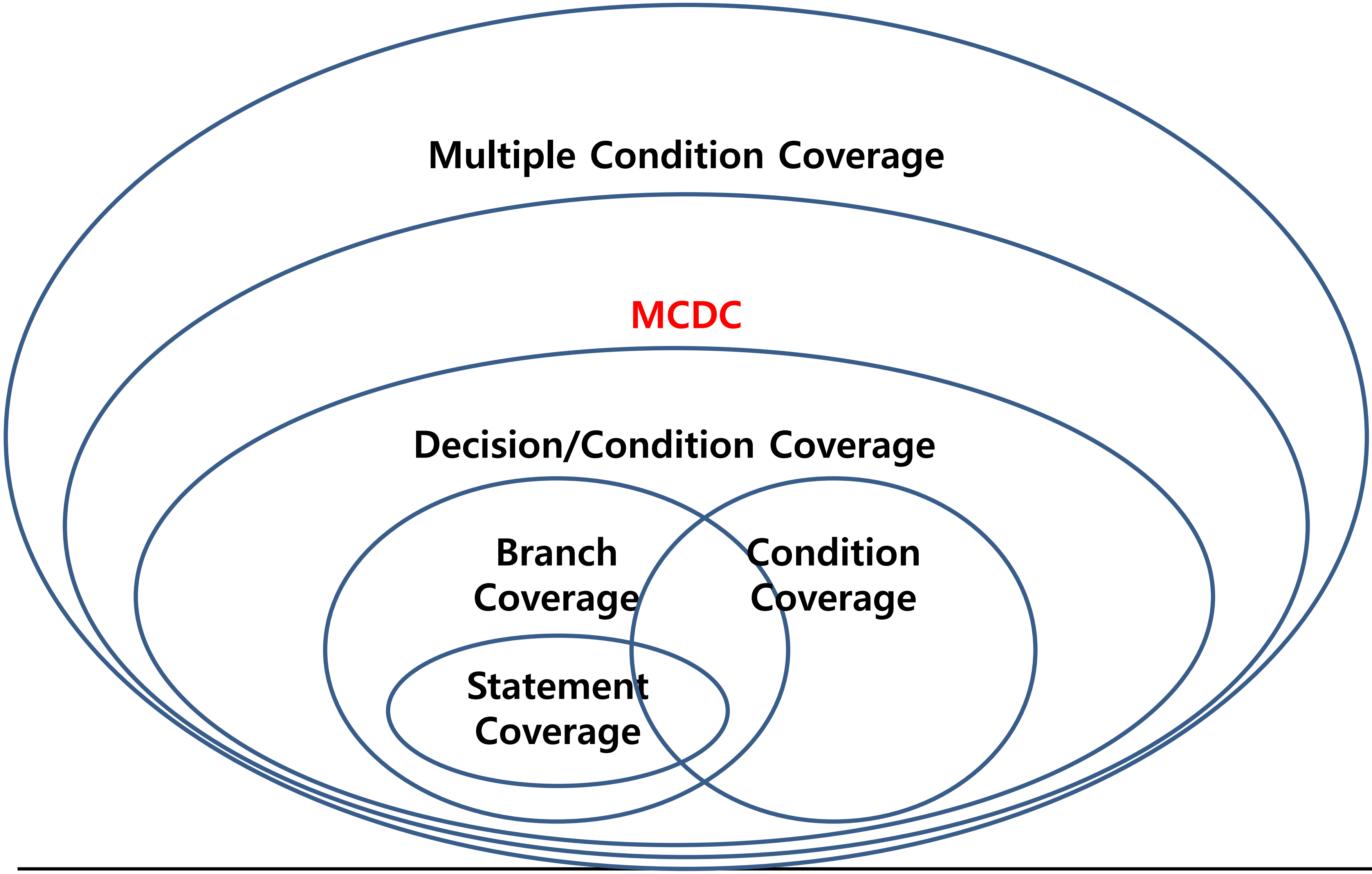
				A or (B and C)					
	A	B	C	Outcome	A			B	C
1	T	T	T	T					
2	T	T	F	T	√				
3	T	F	T	T		√			
4	T	F	F	T			√		
5	F	T	T	T				√	√
6	F	T	F	F	√				√
7	F	F	T	F		√		√	
8	F	F	F	F			√		

MCDC

- A and (B or C)
  - 2, 3, 4, (6 or 7)

	A	B	C	**A and (B or C)**				
				Outcome	A			B	C
1	T	T	T	T	√				
2	T	T	F	T		√		√	
3	T	F	T	T			√		√
4	T	F	F	F				√	√
5	F	T	T	F	√				
6	F	T	F	F		√			
7	F	F	T	F			√		
8	F	F	F	F					

Comparison

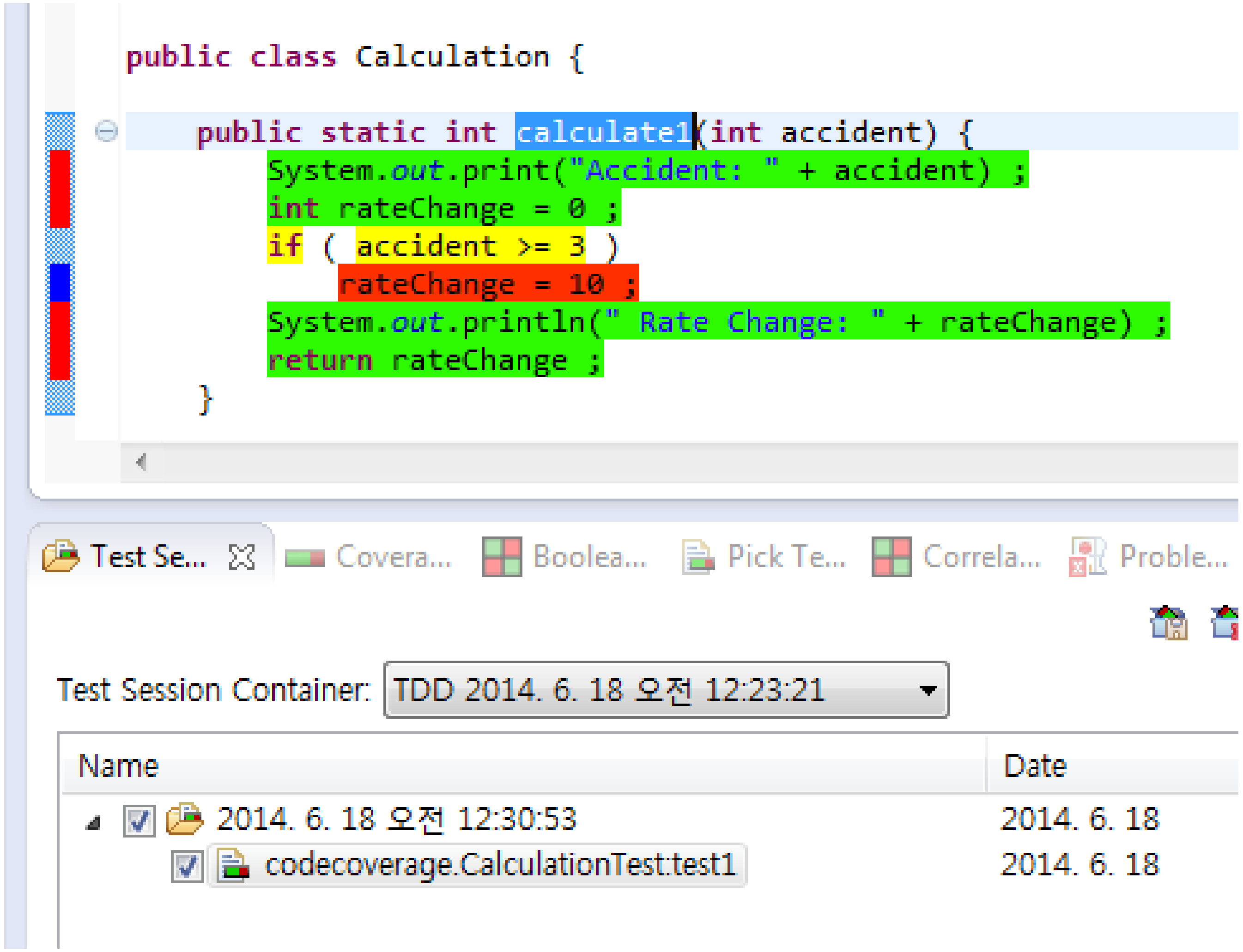

Multiple Condition Coverage
MCDC
Decision/Condition Coverage
Branch Coverage
Condition Coverage
Statement Coverage

Code Coverage - Summary

Coverage	Covered Element
Statement Coverage	Every statement
Decision Coverage	Every decision
Condition Coverage	Every condition
Condition/Decision Coverage	Every condition and decision
MC/DC	Every condition with independent effect on decision
Multiple Condition Coverage	Every combination of condition

CODE COVERAGE 측정

Coverage Analysis



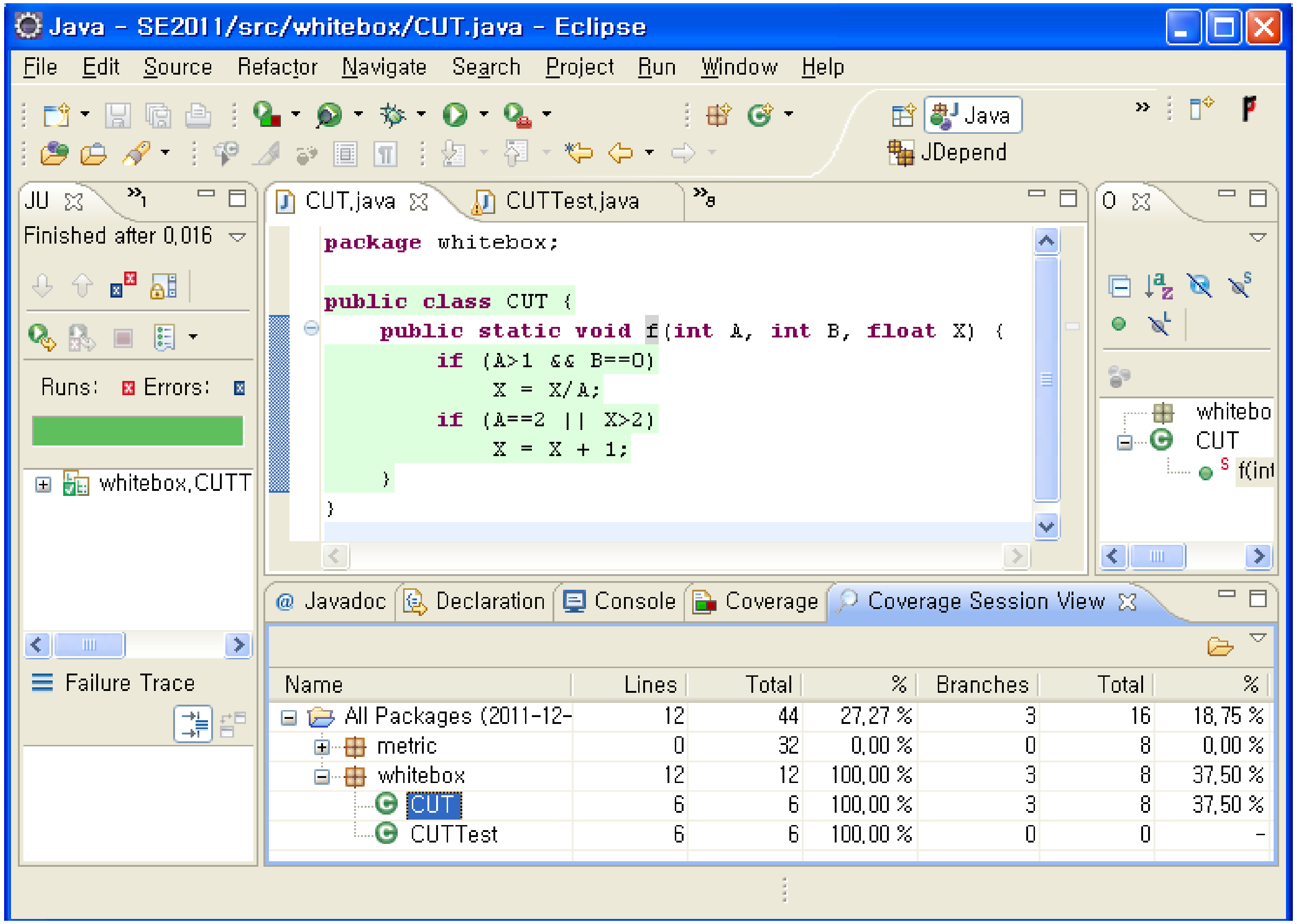
```
public class Calculation {

    public static int calculate1(int accident) {
        System.out.print("Accident: " + accident) ;
        int rateChange = 0 ;
        if ( accident >= 3 )
            rateChange = 10 ;
        System.out.println(" Rate Change: " + rateChange) ;
        return rateChange ;
    }
```

Test Session Container: TDD 2014. 6. 18 오전 12:23:21

Name	Date
2014. 6. 18 오전 12:30:53	2014. 6. 18
codecoverage.CalculationTest:test1	2014. 6. 18

Coverage Analysis Tool

❖eCobertura for Java in Eclipse

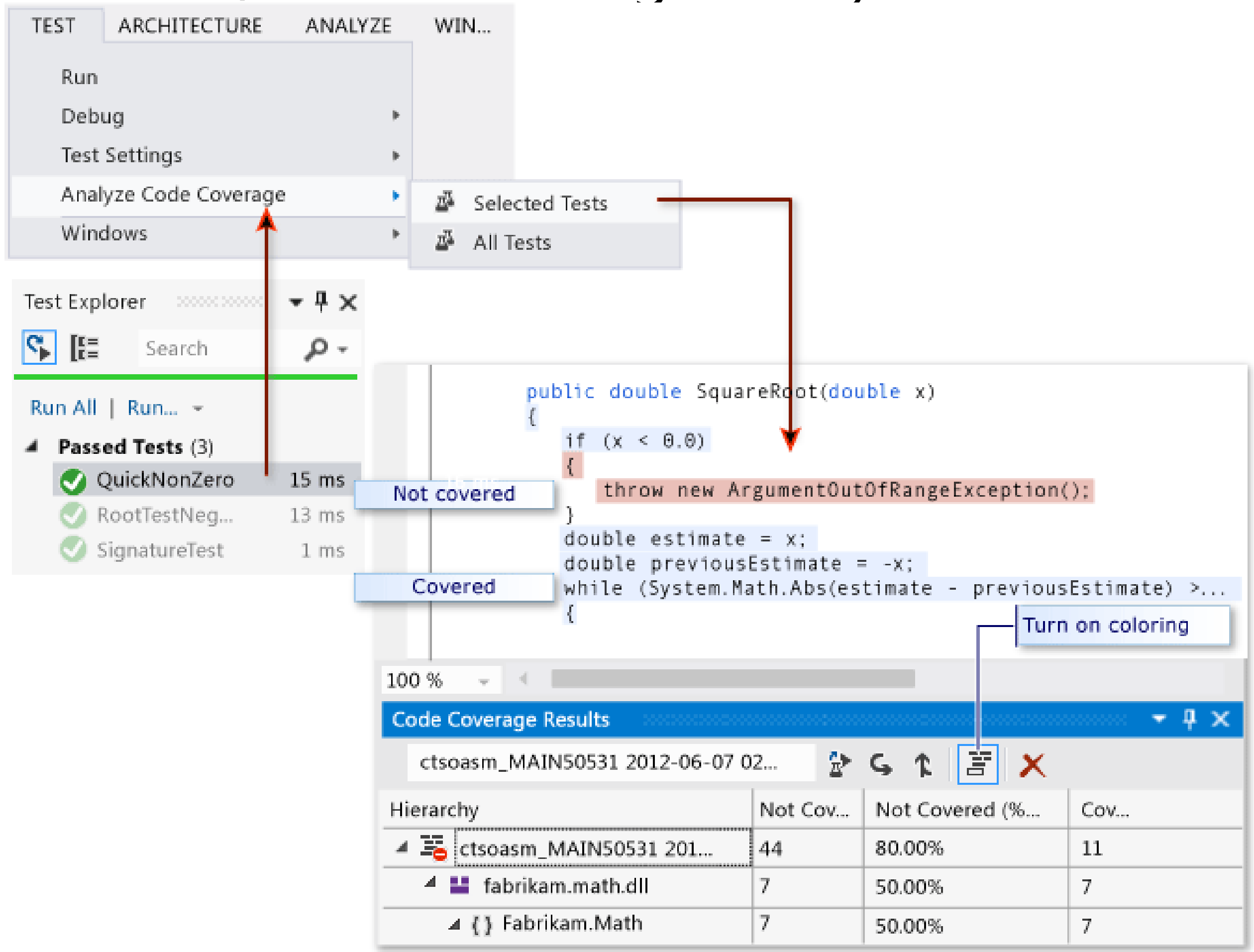


```
package whitebox;

public class CUT {
    public static void f(int A, int B, float X) {
        if (A>1 && B==0)
            X = X/A;
        if (A==2 || X>2)
            X = X + 1;
    }
}
```

Name	Lines	Total	%	Branches	Total	%
All Packages (2011-12-	12	44	27.27 %	3	16	18.75 %
metric	0	32	0.00 %	0	8	0.00 %
whitebox	12	12	100.00 %	3	8	37.50 %
CUT	6	6	100.00 %	3	8	37.50 %
CUTTest	6	6	100.00 %	0	0	-

Coverage Analysis Tool

❖ Visual Studio(Premium/Ultimate) supports statement/block coverage analysis

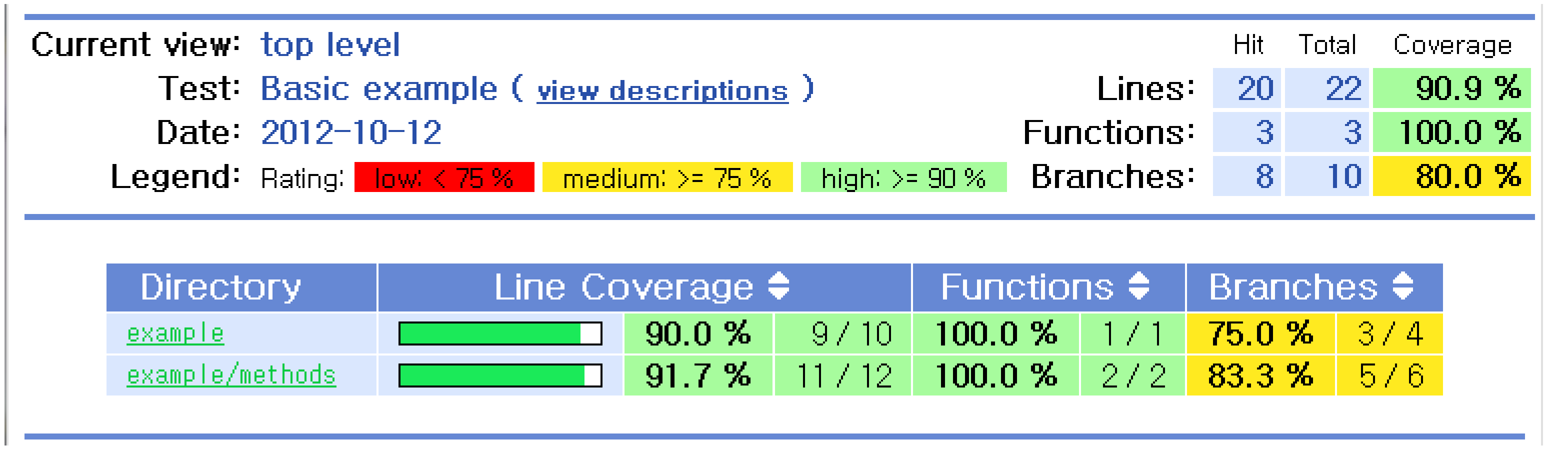


```
public double SquareRoot(double x)
{
    if (x < 0.0)
    {
        throw new ArgumentOutOfRangeException();
    }
    double estimate = x;
    double previousEstimate = -x;
    while (System.Math.Abs(estimate - previousEstimate) >...
    {
```

Hierarchy	Not Cov...	Not Covered (%...	Cov...
ctsoasm_MAIN50531 201...	44	80.00%	11
fabrikam.math.dll	7	50.00%	7
{ } Fabrikam.Math	7	50.00%	7

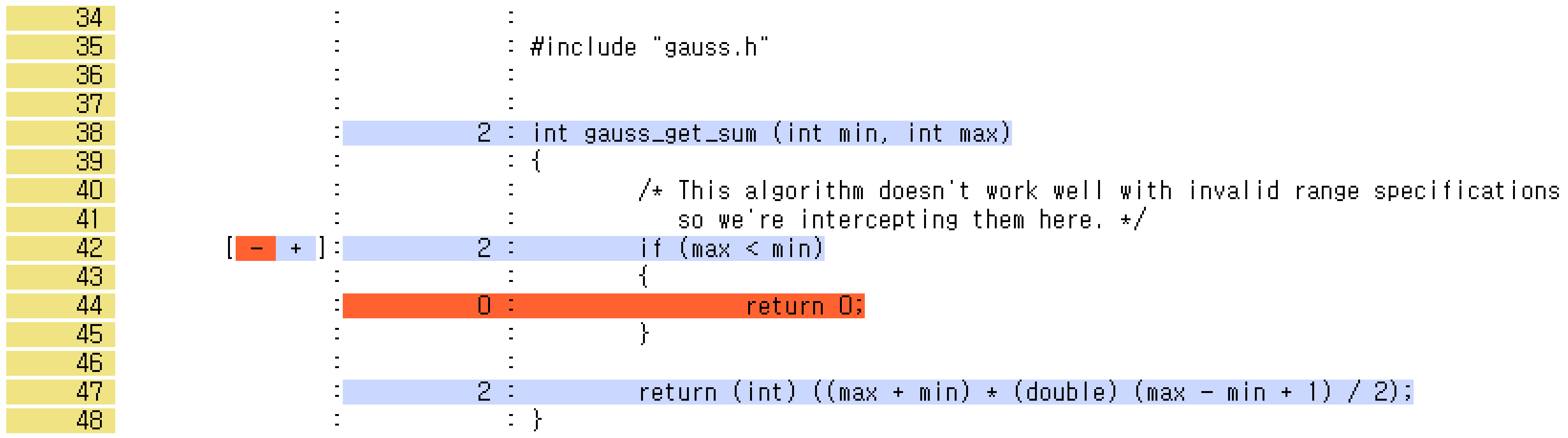
Coverage Analysis Tool

❖ LCOV is a graphical front-end for GCC's coverage testing tool gcov



			Hit	Total	Coverage
Current view:	top level				
Test:	Basic example (view descriptions)	Lines:	20	22	90.9 %
Date:	2012-10-12	Functions:	3	3	100.0 %
Legend:	Rating: low: < 75 % medium: >= 75 % high: >= 90 %	Branches:	8	10	80.0 %

Directory	Line Coverage ⇕		Functions ⇕		Branches ⇕	
example	90.0 %	9 / 10	100.0 %	1 / 1	75.0 %	3 / 4
example/methods	91.7 %	11 / 12	100.0 %	2 / 2	83.3 %	5 / 6



```
            :       :
            :       : #include "gauss.h"
            :       :
            :       :
            :     2 : int gauss_get_sum (int min, int max)
            :       : {
            :       :         /* This algorithm doesn't work well with invalid range specifications
            :       :            so we're intercepting them here. */
   [ - + ]  :     2 :         if (max < min)
            :       :         {
            :     0 :                 return 0;
            :       :         }
            :       :
            :     2 :         return (int) ((max + min) * (double) (max - min + 1) / 2);
            :       : }
```

Q&A