

Cover 커버리지 소개



index

1. 소스 뷰

- 1) 개요

2. Java 소스코드 커버리지

- 1) 함수 커버리지
- 2) 구문 커버리지
- 3) 라인 커버리지
- 4) 분기 커버리지
- 5) 함수 호출 커버리지
- 6) 엔트리 커버리지
- 7) 엑시트 커버리지
- 8) 변경 함수 커버리지
- 9) 변경 라인 커버리지

Software for safe world

소프트웨어로 안전한 세상

If not sure,
Make it sure

국내 최초이자 최고의 Mission-Critical
소프트웨어 전문 기업. 슈어소프트테크

1. 소스 뷰

개요

- 소스 뷰 페이지에서는 모듈의 파일, 함수, 소스코드 라인 별 커버리지를 확인 할 수 있음
- 모든 커버리지 값은 (실제 수행된 개수)/(수행될 수 있는 전체 개수) 형태로 계산되어 백분율로 표시됨
- 예를 들어 파일의 라인 커버리지가 5/9 이면, 파일에 수행될 수 있는 코드가 9라인 존재하고, 이중 5라인이 수행된 것을 의미

모듈
C:\Users\Wkhj\Documents\workspace-sts-3.9.2.RELEASE\Cover-Sample\src\CoverSample.java

실행정보

1. 실행 정보

측정 서버 수: 0

측정 수: 0

19-01-11 09:16:38

커버리지 현황

상태정보

함수: 66.67%

구문: 54.29% 분기: 50% 함수 호출: 60% 라인: 55.56% 엔트리: 50%

엑시트: 40% 변경 함수: N/A 변경 라인: N/A

전체파일

변경파일

파일명

2. 파일별 커버리지

전체함수

변경함수

함수명

3. 함수별 커버리지

파일명	함수(%)	라인(%)	구문(%)	분기(%)	함수 호출(%)	엔트리(%)	엑시트(%)
CoverSample.java	66.67 (2/3)	55.56 (5/9)	54.29 (19/35)	50 (3/6)	60 (3/5)	50 (1/2)	40 (2/5)

함수명	라인(%)	구문(%)	분기(%)	함수 호출(%)	엔트리(%)	엑시트(%)
sum (int, int)	0 (0/1)	0 (0/4)	N/A	N/A	0 (0/1)	0 (0/1)
divide (int, int)	66.67 (2/3)	54.55 (6/11)	50 (1/2)	0 (0/1)	100 (1/1)	100 (1/1)
calculate (String, int, int)	60 (3/5)	65 (13/20)	50 (2/4)	75 (3/4)	N/A	33.33 (1/3)

C:\Users\Wkhj\Documents\workspace-sts-3.9.2.RELEASE\Cover-Sample\src\CoverSample.java

```

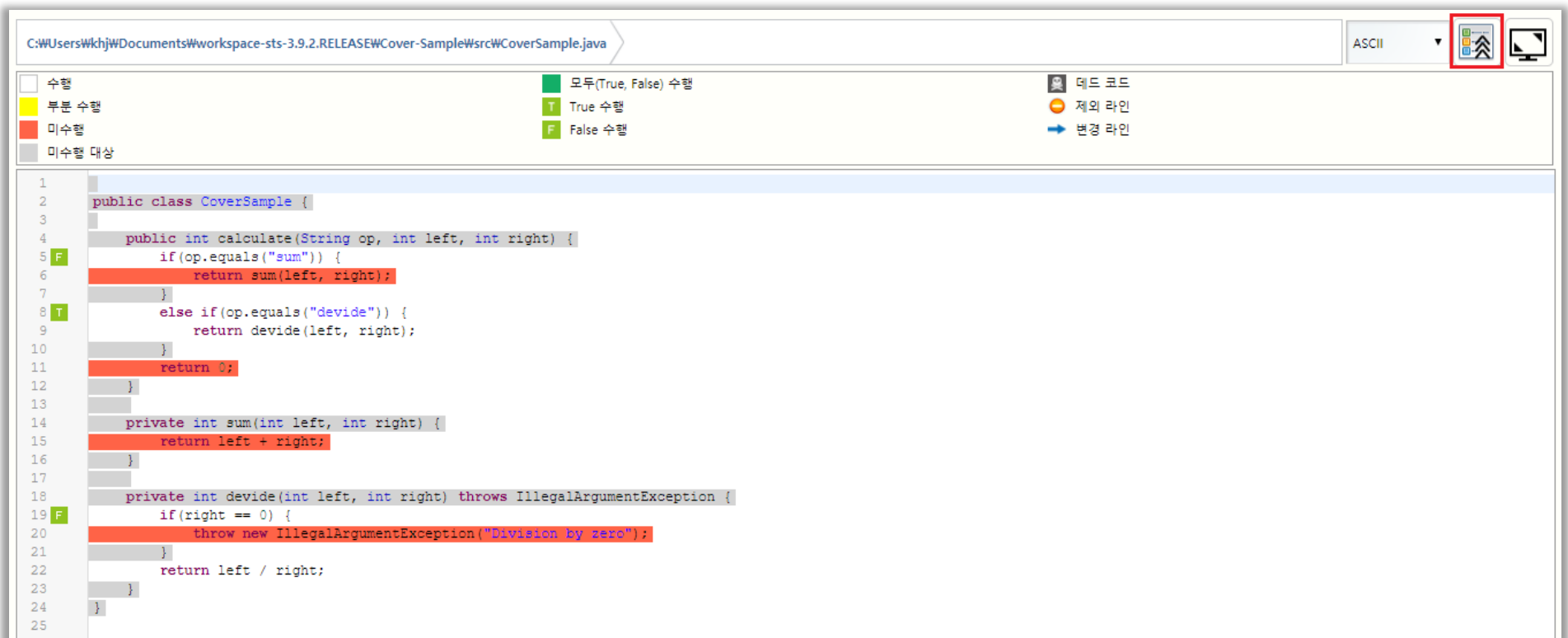
1  public class CoverSample {
2
3
4      public int calculate(String op, int left, int right) {
5          if (op.equals("sum")) {
6              return sum(left, right);
7          }
8          else if (op.equals("divide")) {
9              return divide(left, right);
10         }
11         return 0;
12     }
13
14     private int sum(int left, int right) {

```

4. 소스코드 라인별 커버리지

개요

- 우측 하단의 범례 버튼을 클릭하여 소스코드에 표시된 색상 및 아이콘에 대한 의미를 확인 할 수 있음
- 좌측의 수행, 부분 수행, 미수행, 미수행 대상은 소스코드의 라인 별 커버리지 상태를 의미하고 라인의 배경색으로 표시됨
- 가운데 모두 수행, True 수행, False 수행은 분기의 수행 상태를 의미하고 소스 라인 우측에 표시됨
- 예를 들어 아래 예제의 5번째 라인은 배경이 하얀색이므로 라인의 모든 코드가 수행된 것을 의미함
- 같은 라인의 False 수행 라벨은 해당 라인의 분기가 False 조건만 수행된 것을 의미함



If not sure,
Make it sure

국내 최초이자 최고의 Mission-Critical
소프트웨어 전문 기업. 슈어소프트테크

2. Java 소스코드 커버리지

함수 커버리지

- 소스코드에 포함된 모든 함수 중 수행된 개수를 표시함
- 아래 예제에는 총 3개의 함수가 존재하고, 2개가 수행되어 66.67% (2/3) 으로 계산됨

전체파일

변경파일

파일명

파일명	함수(%)	라인(%)	구문(%)	분기(%)	함수 호출(%)	엔트리(%)	엑시트(%)
CoverSample.java	66.67 (2/3)	55.56 (5/9)	54.29 (19/35)	50 (3/6)	60 (3/5)	50 (1/2)	40 (2/5)

전체함수

변경함수

함수명

함수명	라인(%)	구문(%)	분기(%)	함수 호출(%)	엔트리(%)	엑시트(%)
sum (int, int)	0 (0/1)	0 (0/4)	N/A	N/A	0 (0/1)	0 (0/1)
devide (int, int)	66.67 (2/3)	54.55 (6/11)	50 (1/2)	0 (0/1)	100 (1/1)	100 (1/1)
calculate (String, int, int)	60 (3/5)	65 (13/20)	50 (2/4)	75 (3/4)	N/A	33.33 (1/3)

C:\Users\Wkhjw\Documents\workspace-sts-3.9.2.RELEASE\Cover-Sample\src\CoverSample.java

ASCII

```

1  public class CoverSample {
2
3      public int calculate(String op, int left, int right) {
4          if (op.equals("sum")) {
5              return sum(left, right);
6          }
7          else if (op.equals("devide")) {
8              return devide(left, right);
9          }
10         }
11         return 0;
12     }
13
14     private int sum(int left, int right) {
15         return left + right;
16     }
17
18     private int devide(int left, int right) throws IllegalArgumentException {
19         if (right == 0) {
20             throw new IllegalArgumentException("Division by zero");
21         }
22         return left / right;
23     }
24 }

```

2) 구문 커버리지

- 소스코드를 컴파일하여 생성된 class 파일 내부의 명령어(instruction)의 수행 정도를 표시하는 값

전체파일

변경파일

파일명

파일명	합수(%)	라인(%)	구문(%)	분기(%)	함수 호출(%)	엔트리(%)	엑시트(%)
CoverSample.java	66.67 (2/3)	55.56 (5/9)	54.29 (19/35)	50 (3/6)	60 (3/5)	50 (1/2)	40 (2/5)

전체함수

변경함수

함수명

함수명	라인(%)	구문(%)	분기(%)	함수 호출(%)	엔트리(%)	엑시트(%)
sum (int, int)	0 (0/1)	0 (0/4)	N/A	N/A	0 (0/1)	0 (0/1)
devide (int, int)	66.67 (2/3)	54.55 (6/11)	50 (1/2)	0 (0/1)	100 (1/1)	100 (1/1)
calculate (String, int, int)	60 (3/5)	65 (13/20)	50 (2/4)	75 (3/4)	N/A	33.33 (1/3)

C:\Users\Wkhjw\Documents\workspace-sts-3.9.2.RELEASE\WCover-Sample\src\WCoverSample.java

ASCII

```

1  public class CoverSample {
2
3      public int calculate(String op, int left, int right) {
4          if (op.equals("sum")) {
5              return sum(left, right);
6          }
7          else if (op.equals("devide")) {
8              return devide(left, right);
9          }
10         }
11         return 0;
12     }
13
14     private int sum(int left, int right) {
15         return left + right;
16     }
17
18     private int devide(int left, int right) throws IllegalArgumentException {
19         if (right == 0) {
20             throw new IllegalArgumentException("Division by zero");
21         }
22         return left / right;
23     }
24 }

```

3) 라인 커버리지

- 소스코드상의 실행 가능한 라인을 기준으로 계산된 커버리지 값

전체파일

변경파일

파일명

파일명	합수(%)	라인(%)	구문(%)	분기(%)	함수 호출(%)	엔트리(%)	엑시트(%)
CoverSample.java	66.67 (2/3)	55.56 (5/9)	54.29 (19/35)	50 (3/6)	60 (3/5)	50 (1/2)	40 (2/5)

전체함수

변경함수

함수명

함수명	라인(%)	구문(%)	분기(%)	함수 호출(%)	엔트리(%)	엑시트(%)
sum (int, int)	0 (0/1)	0 (0/4)	N/A	N/A	0 (0/1)	0 (0/1)
devide (int, int)	66.67 (2/3)	54.55 (6/11)	50 (1/2)	0 (0/1)	100 (1/1)	100 (1/1)
calculate (String, int, int)	60 (3/5)	65 (13/20)	50 (2/4)	75 (3/4)	N/A	33.33 (1/3)

C:\Users\Wkhjw\Documents\workspace-sts-3.9.2.RELEASE\Cover-Sample\src\CoverSample.java

ASCII

```

1  public class CoverSample {
2
3      public int calculate(String op, int left, int right) {
4          if (op.equals("sum")) {
5              return sum(left, right);
6          }
7          else if (op.equals("devide")) {
8              return devide(left, right);
9          }
10         }
11         return 0;
12     }
13
14     private int sum(int left, int right) {
15         return left + right;
16     }
17
18     private int devide(int left, int right) throws IllegalArgumentException {
19         if (right == 0) {
20             throw new IllegalArgumentException("Division by zero");
21         }
22         return left / right;
23     }
24 }

```


4) 분기 커버리지

- if, for, while, switch 등과 같은 분기문 으로 생성된 실행 분기의 커버리지를 나타냄
- 전체 실행 가능한 분기 개수를 분모, 실제로 실행된 분기 개수를 분자로 계산함

The screenshot displays the Suresoft IDE interface with two tables showing coverage metrics for the file `CoverSample.java`.

File Coverage Table:

파일명	함수(%)	라인(%)	구문(%)	분기(%)	함수 호출(%)	엔트리(%)	엑시트(%)
CoverSample.java	66.67 (2/3)	55.56 (5/9)	54.29 (19/35)	50 (3/6)	60 (3/5)	50 (1/2)	40 (2/5)

Method Coverage Table:

함수명	라인(%)	구문(%)	분기(%)	함수 호출(%)	엔트리(%)	엑시트(%)
sum (int, int)	0 (0/1)	0 (0/4)	N/A	N/A	0 (0/1)	0 (0/1)
devide (int, int)	66.67 (2/3)	54.55 (6/11)	50 (1/2)	0 (0/1)	100 (1/1)	100 (1/1)
calculate (String, int, int)	60 (3/5)	65 (13/20)	50 (2/4)	75 (3/4)	N/A	33.33 (1/3)

Source Code:

```

1 public class CoverSample {
2
3     public int calculate(String op, int left, int right) {
4         if (op.equals("sum")) {
5             return sum(left, right);
6         }
7         else if (op.equals("devide")) {
8             return devide(left, right);
9         }
10        return 0;
11    }
12
13    private int sum(int left, int right) {
14        return left + right;
15    }
16
17    private int devide(int left, int right) throws IllegalArgumentException {
18        if (right == 0) {
19            throw new IllegalArgumentException("Division by zero");
20        }
21        return left / right;
22    }
23 }
24
  
```

4) 분기 커버리지

- calculate 함수에는 5번째 라인, 8번째 라인에 분기가 존재하고 각각 true, false 조건이 실행 가능함
- 2개의 분기문에 각각 2개의 실행가능 조건이 존재하므로 calculate 함수의 분기 커버리지 분모는 4가 됨
- 5라인의 분기는 False만 수행되었고 8라인의 분기는 True만 수행되었기 때문에 수행된 실행 분기는 2개임
- 따라서 calculate 함수의 분기 커버리지는 2/4 (50%) 값으로 계산됨

```
C:\Users\Wkhj\Documents\workspace-sts-3.9.2.RELEASE\Cover-Sample\src\CoverSample.java

수행
부분 수행
미수행
미수행 대상

모두(True, False) 수행
T True 수행
F False 수행

데드 코드
제외 라인
변경 라인

1
2 public class CoverSample {
3
4     public int calculate(String op, int left, int right) {
5         if(op.equals("sum")) {
6             return sum(left, right);
7         }
8         else if(op.equals("devide")) {
9             return devide(left, right);
10        }
11        return 0;
12    }
13
14    private int sum(int left, int right) {
15        return left + right;
16    }
17
18    private int devide(int left, int right) throws IllegalArgumentException {
19        if(right == 0) {
20            throw new IllegalArgumentException("Division by zero");
21        }
22        return left / right;
23    }
24 }
```

5) 함수호출 커버리지

- 소스코드 내에서 함수호출 구문이 얼마나 수행 되었는지 나타내는 값

전체파일

변경파일

파일명

파일명	함수(%)	라인(%)	구문(%)	분기(%)	함수 호출(%)	엔트리(%)	엑시트(%)
CoverSample.java	66.67 (2/3)	55.56 (5/9)	54.29 (19/35)	50 (3/6)	60 (3/5)	50 (1/2)	40 (2/5)

전체함수

변경함수

함수명

함수명	라인(%)	구문(%)	분기(%)	함수 호출(%)	엔트리(%)	엑시트(%)
sum (int, int)	0 (0/1)	0 (0/4)	N/A	N/A	0 (0/1)	0 (0/1)
divide (int, int)	66.67 (2/3)	54.55 (6/11)	50 (1/2)	0 (0/1)	100 (1/1)	100 (1/1)
calculate (String, int, int)	60 (3/5)	65 (13/20)	50 (2/4)	75 (3/4)	N/A	33.33 (1/3)

C:\Users\Wkhjw\Documents\workspace-sts-3.9.2.RELEASE\Cover-Sample\src\CoverSample.java

ASCII

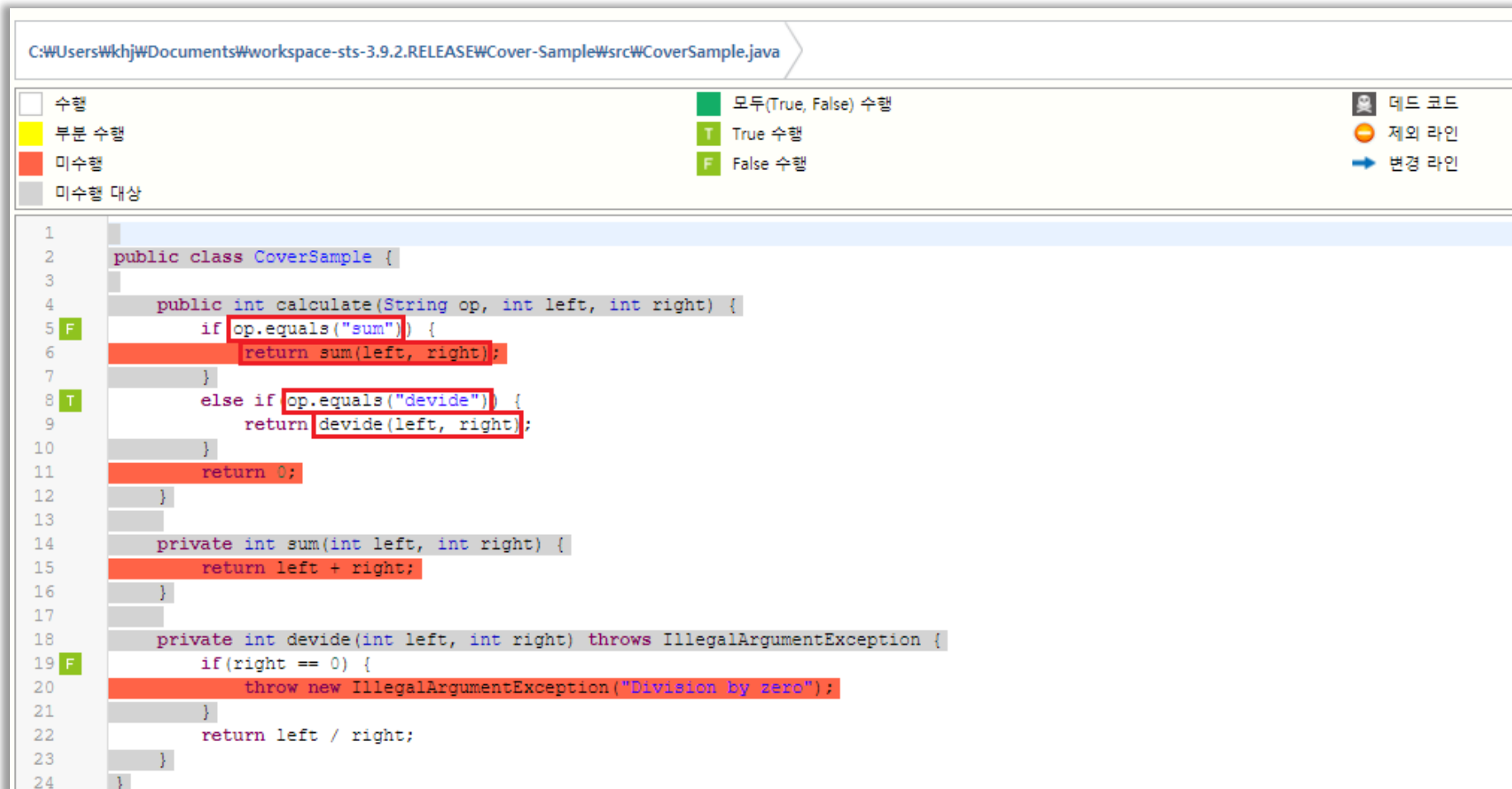
```

1  public class CoverSample {
2
3      public int calculate(String op, int left, int right) {
4          if (op.equals("sum")) {
5              return sum(left, right);
6          }
7          else if (op.equals("divide")) {
8              return divide(left, right);
9          }
10         return 0;
11     }
12
13     private int sum(int left, int right) {
14         return left + right;
15     }
16
17     private int divide(int left, int right) throws IllegalArgumentException {
18         if (right == 0) {
19             throw new IllegalArgumentException("Division by zero");
20         }
21         return left / right;
22     }
23 }
24

```

5) 함수호출 커버리지

- caculate 함수에는 총 4개의 함수호출 구문이 존재함
- 4개의 함수호출 구문중 3개가 수행되어 3/4 (75%) 값으로 계산됨



```
C:\Users\Wkhj\Documents\workspace-sts-3.9.2.RELEASE\Cover-Sample\src\CoverSample.java

수행
부분 수행
미수행
미수행 대상

모두(True, False) 수행
T True 수행
F False 수행

데드 코드
제외 라인
변경 라인

1
2 public class CoverSample {
3
4     public int calculate(String op, int left, int right) {
5         if (op.equals("sum")) {
6             return sum(left, right);
7         }
8         else if (op.equals("devide")) {
9             return devide(left, right);
10        }
11        return 0;
12    }
13
14    private int sum(int left, int right) {
15        return left + right;
16    }
17
18    private int devide(int left, int right) throws IllegalArgumentException {
19        if (right == 0) {
20            throw new IllegalArgumentException("Division by zero");
21        }
22        return left / right;
23    }
24 }
```

6) 엔트리 커버리지

- 소스코드 내부에서 해당 함수를 호출하는 구문의 수행된 정도를 표현하는 값
- 아래 코드에서는 divide 함수를 호출하는 구문이 9라인에 1개 존재하고, 수행 되었기 때문에 1/1 (100%)로 계산됨

전체파일

변경파일

파일명

파일명	함수(%)	라인(%)	구문(%)	분기(%)	함수 호출(%)	엔트리(%)	엑시트(%)
CoverSample.java	66.67 (2/3)	55.56 (5/9)	54.29 (19/35)	50 (3/6)	60 (3/5)	50 (1/2)	40 (2/5)

전체함수

변경함수

함수명

함수명	라인(%)	구문(%)	분기(%)	함수 호출(%)	엔트리(%)	엑시트(%)
sum (int, int)	0 (0/1)	0 (0/4)	N/A	N/A	0 (0/1)	0 (0/1)
divide (int, int)	66.67 (2/3)	54.55 (6/11)	50 (1/2)	0 (0/1)	100 (1/1)	100 (1/1)
calculate (String, int, int)	60 (3/5)	65 (13/20)	50 (2/4)	75 (3/4)	N/A	33.33 (1/3)

C:\Users\Wkhjw\Documents\workspace-sts-3.9.2.RELEASE\WCover-Sample\src\WCoverSample.java

ASCII

```

1  public class CoverSample {
2
3      public int calculate(String op, int left, int right) {
4          if (op.equals("sum")) {
5              return sum(left, right);
6          }
7          else if (op.equals("divide")) {
8              return divide(left, right);
9          }
10         return 0;
11     }
12
13     private int sum(int left, int right) {
14         return left + right;
15     }
16
17     private int divide(int left, int right) throws IllegalArgumentException {
18         if (right == 0) {
19             throw new IllegalArgumentException("Division by zero");
20         }
21         return left / right;
22     }
23 }
24

```

7) 엑시트 커버리지

- 함수의 종료구문의 실행여부를 계산하여 나타내는 값
- calculate 함수의 경우 3개의 return 구문이 존재하고, 그 중 1개가 실행되어 1/3(33.33%)로 계산됨

파일명	함수(%)	라인(%)	구문(%)	분기(%)	함수 호출(%)	엔트리(%)	엑시트(%)
CoverSample.java	66.67 (2/3)	55.56 (5/9)	54.29 (19/35)	50 (3/6)	60 (3/5)	50 (1/2)	40 (2/5)

함수명	라인(%)	구문(%)	분기(%)	함수 호출(%)	엔트리(%)	엑시트(%)
sum (int, int)	0 (0/1)	0 (0/4)	N/A	N/A	0 (0/1)	0 (0/1)
devide (int, int)	66.67 (2/3)	54.55 (6/11)	50 (1/2)	0 (0/1)	100 (1/1)	100 (1/1)
calculate (String, int, int)	60 (3/5)	65 (13/20)	50 (2/4)	75 (3/4)	N/A	33.33 (1/3)


```

1  public class CoverSample {
2
3      public int calculate(String op, int left, int right) {
4          if (op.equals("sum")) {
5              return sum(left, right);
6          }
7          else if (op.equals("devide")) {
8              return devide(left, right);
9          }
10         }
11         return 0;
12     }
13
14     private int sum(int left, int right) {
15         return left + right;
16     }
17
18     private int devide(int left, int right) throws IllegalArgumentException {
19         if (right == 0) {
20             throw new IllegalArgumentException("Division by zero");
21         }
22         return left / right;
23     }
24 }
  
```

Software for safe world

Thank you for your kind attention

www.suresofttech.com

8) 변경 라인 커버리지

- 이전 대상과 비교하여 수정/변경/추가된 라인의 실행 여부
- 소스 뷰 화면에서 이전 대상과 비교하여 수정/변경/추가된 라인의 경우 → **변경 라인** 으로 표기

전체파일

변경파일

파일명

파일명	구문(%)	분기(%)	함수(%)	라인(%)	함수 호출(%)	엔트리(%)	엑시트(%)	MC/DC(%)
main.c	100 (4/4)	N/A	100 (1/1)	100 (4/4)	100 (3/3)	N/A	100 (1/1)	N/A

전체함수

변경함수

함수명

함수명	수행 여부	구문(%)	분기(%)	라인(%)	함수 호출(%)	엔트리(%)	엑시트(%)	MC/DC(%)
main()	O	100 (4/4)	N/A	100 (4/4)	100 (3/3)	N/A	100 (1/1)	N/A

D:\WProjects\WProject1\WProject1\Wmain.c

EUC-KR

```

1  #include <stdio.h>
2
3  int main()
4  {
5      printf("Hello, world!\n");
6      → printf("Hello, world!\n");
7
8      → printf("라인추가!\n");
9
10     return 0;
11 }

```

9) 변경 함수 커버리지

- 이전 대상과 비교하여 수정/변경/추가된 함수의 실행 여부
- 이전 대상과 비교하여 수정/변경/추가된 함수의 경우 변경함수에서 해당 함수명을 표기

The screenshot displays the Suresoft code coverage tool interface. The top navigation bar includes tabs for '전체파일' (All Files), '변경파일' (Changed Files), '전체함수' (All Functions), and '변경함수' (Changed Functions), with the latter being the active tab. Below the navigation bar, there are two tables. The left table shows file-level coverage for 'main.c', and the right table shows function-level coverage for 'main()'. Both show 100% coverage across all metrics. The bottom section contains a legend for coverage status (e.g., Covered, Not Covered, Partially Covered) and a code editor showing the source code of 'main.c' with line 6 highlighted by a blue arrow, indicating it was executed.

파일명	구문(%)	분기(%)	함수(%)	라인(%)	함수 호출(%)	엔트리(%)	엑시트(%)	MC/DC(%)
main.c	100 (4/4)	N/A	100 (1/1)	100 (4/4)	100 (3/3)	N/A	100 (1/1)	N/A

함수명	수행 여부	구문(%)	분기(%)	라인(%)	함수 호출(%)	엔트리(%)	엑시트(%)	MC/DC(%)
main()	O	100 (4/4)	N/A	100 (4/4)	100 (3/3)	N/A	100 (1/1)	N/A

Legend:

- 커버됨 (Covered)
- 부분 수행 (Partially Covered)
- 수행되지 않음 (Not Covered)
- 측정 대상 아님 (Not Measured)
- 모두(True, False) 수행 (All executed)
- True만 수행 (True only executed)
- False만 수행 (False only executed)
- 다중 분기 부분 수행 (Multiple branch partially covered)
- 데드 코드 (Dead code)
- 제외 라인 (Excluded line)
- 변경 라인 (Changed line)

```

1  #include <stdio.h>
2
3  int main()
4  {
5      printf("Hello, world!\n");
6      printf("Hello, world!\n");
7
8      printf("라인추가!\n");
9
10     return 0;
11 }
  
```