

# Code Coverage and Test Reports with IntelliJ



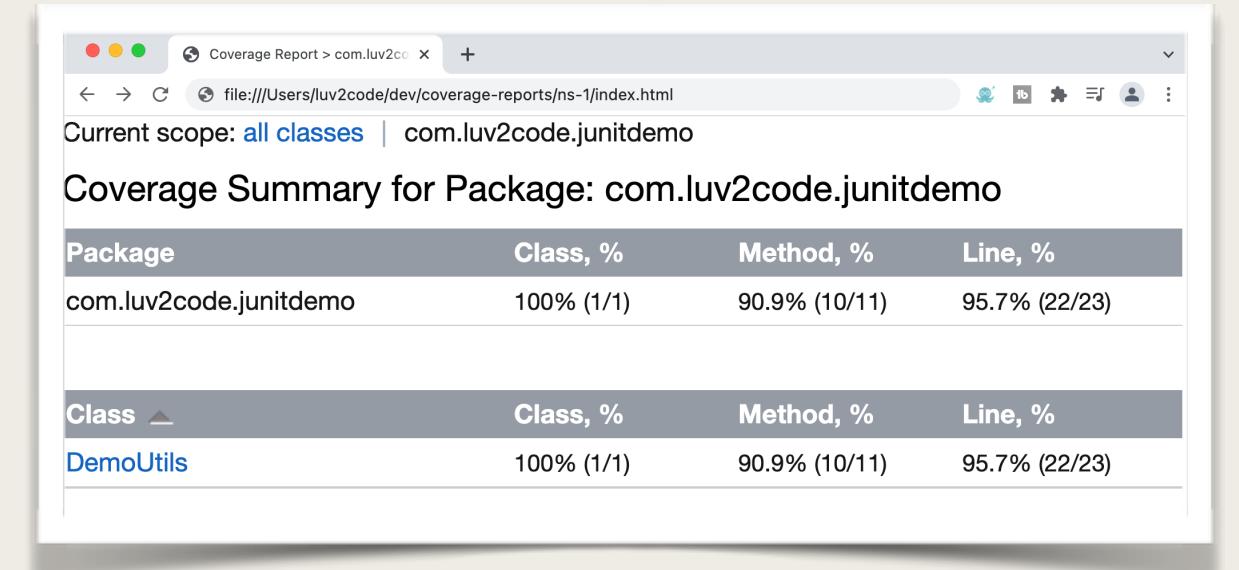
# Code Coverage

- Code coverage measures how many methods/lines are called by your tests
- Coverage is represented as a percentage: 50% coverage etc ...
- In general, the higher the coverage the better
  - However, 100% is not always attainable
  - On most teams, 70%-80% is acceptable
- Code coverage is only a metric and can be easily tricked with bad tests
- Use the metric as simply one data point in your software process



# IntelliJ Support for Code Coverage

- IntelliJ has built-in support for code coverage
- Can generate coverage reports in the IDE
- Also can generate HTML output for viewing in web browser
- If you are not using IntelliJ, don't worry
  - In later videos, I'll show you how to do the same using Maven commands
  - No IDE is required ... run Maven from command-line



# IntelliJ Support for Code Coverage

The screenshot shows the IntelliJ IDEA interface with a code coverage tool overlay. The coverage report for 'DemoUtilsTest' indicates 100% classes, 95% lines covered in 'all classes in scope'. The 'Element' table shows the package 'com.luv2code.junitdemo' with 100% class coverage (1/1), 90% method coverage (9/10), and 95% line coverage (22/23). Below the report, the 'DemoUtils' class code is displayed, with several lines highlighted in yellow, indicating they are covered by tests. A red callout box points to line 32, which contains the 'multiply' method, with the text 'multiply(...) method not covered by our unit tests'.

Coverage: DemoUtilsTest

100% classes, 95% lines covered in 'all classes in scope'

Element	Class, %	Method, %	Line, %
com.luv2code.junitdemo	100% (1/1)	90% (9/10)	95% (22/23)

```
public class DemoUtils {  
    private String academy = "Luv2Code Academy";  
    private String academyDuplicate = academy;  
    private String[] firstThreeLettersOfAlphabet = {"A", "B", "C"};  
    private List<String> academyInList = List.of("Luv", "2", "code");  
  
    public List<String> getAcademyInList() { return academyInList; }  
  
    public String getAcademy() { return academy; }  
  
    public String getAcademyDuplicate() { return academyDuplicate; }  
  
    public String[] getFirstThreeLettersOfAlphabet() { return firstThreeLettersOfAlphabet; }  
  
    public int add(int a, int b) { return a + b; }  
  
    public int multiply(int a, int b) { return a * b; }  
  
    public Object.checkNotNull(Object obj) {  
        if (obj == null) {  
            throw new NullPointerException("Object cannot be null");  
        }  
        return obj;  
    }  
}
```

multiply(...) method not covered by our unit tests

Show Context Actions

Paste ⌘V

Copy / Paste Special >

Column Selection Mode ⇧⌘8

Refactor >

Folding >

Analyze >

Go To >

Generate... ⌘N

Run 'DemoUtilsTest' ⌘^R

Debug 'DemoUtilsTest' ⌘^D

Run 'DemoUtilsTest' with Coverage (selected)

Modify Run Configuration...

Open...

Locate...

Git...

Compare with...

Create Gist...

Check Current File

# IntelliJ - Generate Code Coverage Report

The screenshot shows an IntelliJ browser window with the title "Coverage Report > com.luv2code.junitdemo". The URL is "file:///Users/luv2code/dev/coverage-reports/ns-1/index.html". The page displays coverage statistics for the package "com.luv2code.junitdemo".

**Coverage Summary for Package: com.luv2code.junitdemo**

Package	Class, %	Method, %	Line, %
com.luv2code.junitdemo	100% (1/1)	90.9% (10/11)	95.7% (22/23)

**Coverage Summary for Class: DemoUtils (com.luv2code.junitdemo)**

Class	Class, %	Method, %	Line, %
DemoUtils	100% (1/1)	90.9% (10/11)	95.7% (22/23)

multiply(...) method not covered  
by our unit tests

Current scope: all classes | com.luv2code.junitdemo

Coverage Summary for Class: DemoUtils (com.luv2code.junitdemo)

Class

DemoUtils

```
1 package com.luv2code.junitdemo;
2
3 import java.util.List;
4
5 public class DemoUtils {
6
7     private String academy = "Luv2Code Academy";
8     private String academyDuplicate = academy;
9     private String[] firstThreeLettersOfAlphabet = {"A", "B", "C"};
10    private List<String> academyInList = List.of("luv", "2", "code");
11
12    public List<String> getAcademyInList() {
13        return academyInList;
14    }
15
16    public String getAcademy() {
17        return academy;
18    }
19
20    public String getAcademyDuplicate() {
21        return academyDuplicate;
22    }
23
24    public String[] getFirstThreeLettersOfAlphabet() {
25        return firstThreeLettersOfAlphabet;
26    }
27
28    public int add(int a, int b) {
29        return a + b;
30    }
31
32    public int multiply(int a, int b) {
33        return a * b;
34    }
35
36    public Object checkNull(Object obj) {
37
38        if (obj != null) {
39            return obj;
40        }
41    }
42}
```

# IntelliJ - Generate Test Reports

The screenshot shows a web browser window titled "Test Results – DemoUtilsTest". The URL in the address bar is "file:///Users/luv2code/dev/test-reports/Test%20Results%20-%20DemoUtilsTest.html". The main content of the page is a test report for "DemoUtilsTest". The report summary states "DemoUtilsTest: 9 total, 9 passed" and "2.06 s". Below this, there is a "Collapse | Expand" link. The detailed test results table lists nine test cases, all of which have passed:

Test Case	Status	Time
Same and Not Same	passed	23 ms
Array Equals	passed	3 ms
True and False	passed	2 ms
Null and Not Null	passed	2 ms
Equals and Not Equals	passed	3 ms
Timeout	passed	2.02 s
Lines match	passed	4 ms
Iterable equals	passed	2 ms
Throws and Does Not Throw	passed	3 ms

# IntelliJ Documentation

- Additional features and support for testing

<https://www.jetbrains.com/help/idea/tests-in-ide.html>