

Objectives

- Software testing issue: when have I tested enough?
- Coverage

Oct 20, 2008

Sprenkle - CS209

1

Review

- What is *unit testing*?
- What are the benefits of unit testing?
- What are the steps in a JUnit Test Case?

Oct 20, 2008

Sprenkle - CS209

2

Review: Example Test Cases for Calculator Program

- Basic Functionality
 - Addition
 - Subtraction
 - Multiplication
 - Division
 - Order of operations
- Invalid Input
 - Letters, not-operation characters (&,\$, ...)
- "Tricky" Cases
 - Divide by 0
 - Negative Numbers
 - Long sequences of operands, operators
 - VERY large, VERY small numbers

Oct 20, 2008

Sprenkle - CS209

3

Software Testing Issues

- How do we know if the calculator program is correct?
 - How do we know that we've exposed all the bugs?
 - How confident are we in its correctness?
- How do we know if we've tested enough?
 - Our customers want this product soon but we need product to be correct
 - Harder to fix after it has been released

Oct 20, 2008

Sprenkle - CS209

4

Software Testing Issues

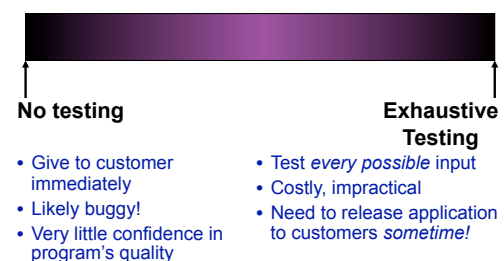
- How do we know if the calculator program is correct?
 - How do we know that we've exposed all the bugs?
 - How confident are we in its correctness?
- How do we know if we've tested enough?
 - Time? It's been a couple hours/days/...
 - Number of test cases executed? A lot!
 - I asked my brother and he's really smart and he says that it's enough

Oct 20, 2008

Sprenkle - CS209

5

Testing Continuum

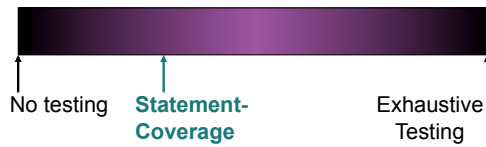


Oct 20, 2008

Sprenkle - CS209

6

Testing Continuum



- Focused on run-time bugs
 - Need to execute **all code**
- Cover (i.e., execute) all **statements** in the program

Oct 20, 2008

Sprenkle - CS209

7

Statement Coverage

- Cover all statements in the program

Test Suite:
num=5

```

✓ public String exampleMethod(int num) {
✓   String string = null;
✓   if (num < 10) {
      string = "" + condition;
    }
  // remove the leading & trailing whitespace
  return string.trim();
}

```

Is this method bug-free?

Oct 20, 2008

Sprenkle - CS209

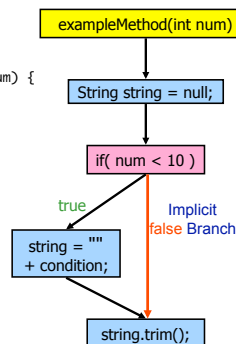
8

Program Flow

```

public String exampleMethod(int num) {
  String string = null;
  if (num < 10) {
    string = "" + condition;
  }
  return string.trim();
}

```



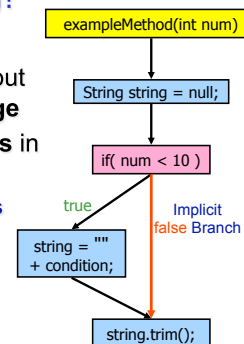
Oct 20, 2008

Sprenkle - CS209

9

What Went Wrong?

- Test suite had 100% statement coverage but missed a **branch/edge**
- Try covering all **edges** in program's flow
 - Also covers all **nodes**
 - Called **Branch Coverage**



Oct 20, 2008

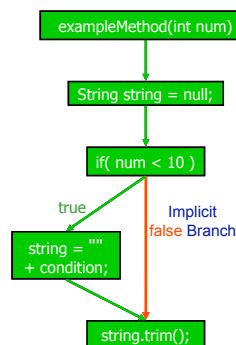
Sprenkle - CS209

10

Branch Coverage

- Cover all **branches** in the program

Test Suite:
num=5,
num=10



Oct 20, 2008

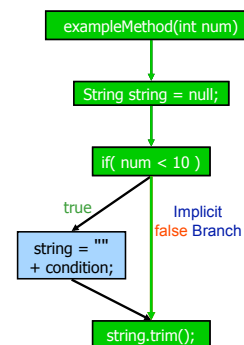
Sprenkle - CS209

11

Branch Coverage

- Cover all **branches** in the program

Test Suite:
num=5,
num=10



Oct 20, 2008

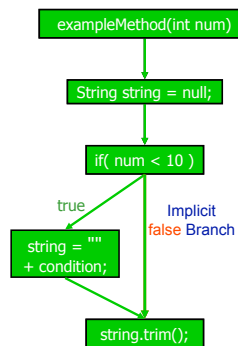
Sprenkle - CS209

12

Branch Coverage

- Cover all **branches** in the program

Test Suite:
num=5,
num=10



Oct 20, 2008

Sprenkle - CS209

13

Interlude: Eclipse Debugger

Oct 20, 2008

Sprenkle - CS209

14

Example 2

```

public String exampleMethod(int a) {
    String str = "d";
    if ( a < 7 ) {
        a *= 2;
        str += "riv";
    } else {
        str = "co" + str;
    }

    if( a > 10 ) {
        str += "ing";
    } else {
        str += "es";
    }
    return str.substring(6);
}

```

Oct 20, 2008

Sprenkle - CS209

15

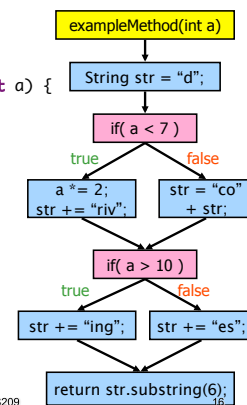
Example 2

```

public String exampleMethod(int a) {
    String str = "d";
    if ( a < 7 ) {
        a *= 2;
        str += "riv";
    } else {
        str = "co" + str;
    }

    if( a > 10 ) {
        str += "ing";
    } else {
        str += "es";
    }
    return str.substring(6);
}

```

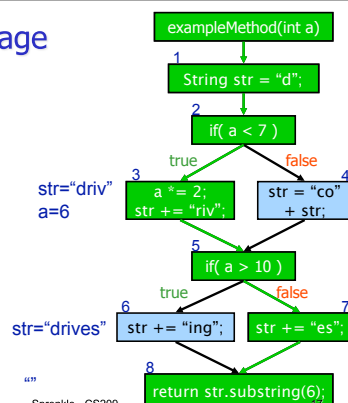


Oct 20, 2008

Sprenkle - CS209

Branch Coverage

Test Suite:
a=3,
a=30

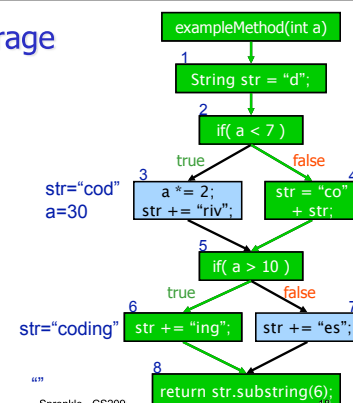


Oct 20, 2008

Sprenkle - CS209

Branch Coverage

Test Suite:
a=3,
a=30

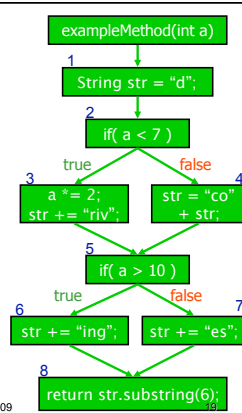


Oct 20, 2008

Sprenkle - CS209

Branch Coverage

Test Suite:
a=3,
a=30

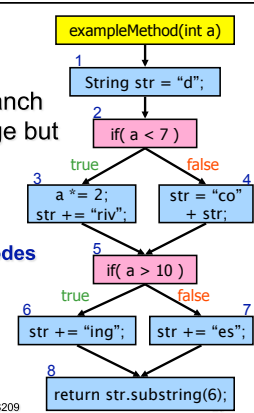


Oct 20, 2008

Sprengle - CS209

What Went Wrong?

- Test suite had 100% branch (and statement) coverage but missed a **path**
- Try to cover all **paths** in program's flow
 - Also gets all **branches, nodes**
 - Called **Path Coverage**

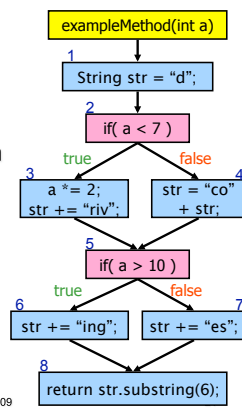


Oct 20, 2008

Sprengle - CS209

Path Coverage

- Cover all **paths** in program's flow
- How many paths through this method?

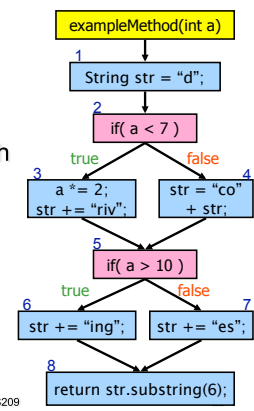


Oct 20, 2008

Sprengle - CS209

Path Coverage

- Cover all **paths** in program's flow
- How many paths through this method?
 - 1-2-3-5-6-8
 - 1-2-3-5-7-8
 - 1-2-4-5-6-8
 - 1-2-4-5-7-8
- What test cases would give us path coverage?



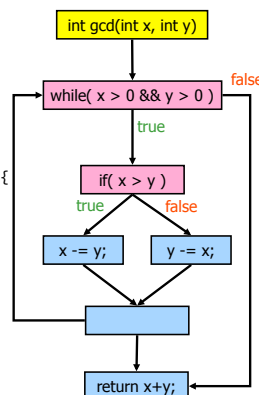
Oct 20, 2008

Sprengle - CS209

Example 3

```

/**
 * Euclid's algorithm to
 * calculate greatest
 * common divisor
 */
public int gcd( int x, int y ) {
    while ( x > 0 && y > 0 ) {
        if ( x > y ) {
            x -= y;
        } else {
            y -= x;
        }
    }
    return x+y;
}
  
```



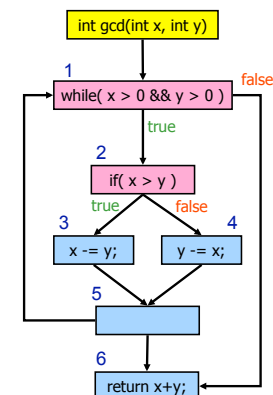
Oct 20, 2008

Sprengle - CS209

23

Path Coverage

- How many paths through this method?
 - Too many to count, test them all!

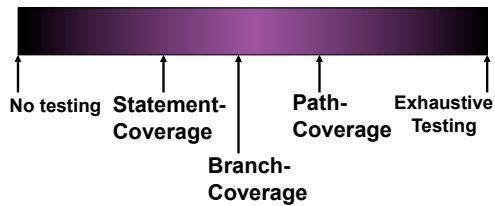


Oct 20, 2008

Sprengle - CS209

24

Testing Continuum




Oct 20, 2008

Sprenkle - CS209

25

Comparison of Coverage

				
No testing	Statement	Branch	Path	Exhaustive Testing
Coverage Criterion	Advantages		Disadvantages	
Statement				
Branch				
Path				

Oct 20, 2008

Sprenkle - CS209

26

Comparison of Coverage

Coverage Criterion	Advantages	Disadvantages
Statement	Practical	Weak, may miss many faults
Branch	Practical, Stronger than Statement	Weaker than Path
Path	Strongest	Infeasible, too many paths to be practical

Oct 20, 2008

Sprenkle - CS209

27

Uses of Coverage Criteria

- "Stopping" rule → sufficient testing
 - Avoid unnecessary, redundant tests
- Measure test quality
 - Dependability estimate
 - Confidence in estimate
- Specify test cases
 - Describe additional test cases needed

Oct 20, 2008

Sprenkle - CS209

28

Coverage Criteria Discussion

- Is it always possible for a test suite to cover all the statements in a given program?
 - No. Could be infeasible statements
 - Unreachable code
 - Legacy code
 - Configuration that is not on site
- Do we need the test suite to cover 100% of statements/branches to believe it is adequate?
 - 100% coverage does not mean correct program
 - But < 100% coverage does mean testing inadequacy

Oct 20, 2008

Sprenkle - CS209

29

True/False Quiz

- A program that passes all test cases in a test suite with 100% path coverage is bug-free.
 - False.
 - The test suite may cover a faulty path with data values that don't expose the fault.
 - Towards Exhaustive Testing

Oct 20, 2008

Sprenkle - CS209

30

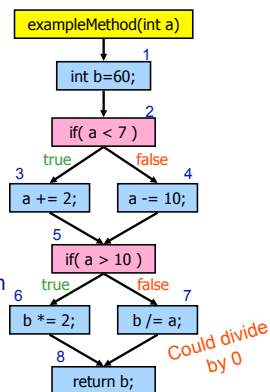
Example

Test Suite:

3-7: a=3,
4-6: a=30,
3-6: a=6,
4-7: a=9

But, error shows up with

3-7: a=0
4-7: a=10



Oct 20, 2008

Sprenkle - CS209

31

True/False Quiz

- When you add test cases to a test suite that covers all statements so that it covers all branches, the new test suite is more likely to be better at exposing faults.

- True.
- You're adding test cases and covering new paths, which may have faults.

Oct 20, 2008

Sprenkle - CS209

32

Choose Your Test Suite

Statement-adequate Test Suite

Branch-adequate Test Suite

- Branch-adequate suite is not necessarily better than Statement-adequate suite
 - Statement-adequate suite could cover buggy paths that Branch-adequate suite doesn't

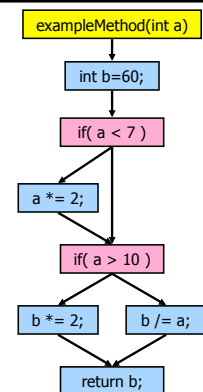
Oct 20, 2008

Sprenkle - CS209

33

Example

- TS1 (Statement-Adequate):
 - a=0, 6
- TS2 (Branch-Adequate):
 - a=3, 30
- Statement-adequate will find fault but branch-adequate won't
 - Covers the path that exposes the fault



Oct 20, 2008

Sprenkle - CS209

34

Software Testing: When is Enough Enough?

- Need to decide when tested enough
 - Balance goals of releasing application, high quality standards
- Can use program coverage as "stopping" rule
 - Also measure of confidence in test suite
 - Statement, Branch, Path and their tradeoffs
 - Use coverage tools to measure statement, branch coverage
- Still, need to use some other "smarts" besides program coverage for creating test cases

Oct 20, 2008

Sprenkle - CS209

35