Objectives

- Testing
- Coverage

1

Review

1. What is our workflow in Git—even more important since we're working on a team project?
2. True or False. Our team's GitHub repository and git are all my team needs to effectively collaborate.
3. What are the steps to a JUnit test case?
   - How do we implement them?
4. How is writing/running the tests for the project different from the tests for the mutant lab?

2

Review: Collaboration: Workflow

1. Create a branch for your work
   - Commit periodically
   - Write descriptive comments so your team members know what you did and why
2. Switch back to master
3. Pull master branch
4. Merge your branch into the master branch
   - Handle merge conflicts
   - Commit
5. Push master branch

Are These Effective Tests?

```
@Test
public void testThirdShortest() {
    String[] words = { "a", "ab", "abc" };
    String actual = mutant.thirdShortest(words);
    assertEquals(3, actual.length());
}
```

```
@Test
public void testExceptionThrown() {
    String[] words = { "a" };
    assertThrows(Exception.class, () -> {
        mutant.thirdShortest(words);
    });
}
```

Test Discussion

- They are correct tests
  - They will reveal bugs
- However, they are *weak* tests
  - Cover necessary invariants, but they are not sufficient to expose failures

```
@Test
public void testThirdShortest() {
    String[] words = { "a", "ab", "abc" };
    String actual = mutant.thirdShortest(words);
    assertEquals("abc", actual);
}
```

Check the actual result

```
@Test
public void testExceptionThrown() {
    String[] words = { "a" };
    assertThrows(IllegalArgumentException.class,
    () -> {
        mutant.thirdShortest(words);
    });
}
```

Expect the exact exception

5

Testing More Than One Possible Answer

- `thirdShortest` only returns one answer (a String) but there could be multiple different correct answers
  - We can discuss if this is the best design but …
- Example test

```
@Test
public void testMoreInArray2() {
    String[] words = { "a", "b", "bc", "ab", "bye", "and" };
    String result = mutant.thirdShortest(words);
    assertTrue(result.equals("bye") || result.equals("and"));
}
```


6

Is This An Effective Test?

```
@Test
public void testAll() {
    String[][] tests = { { "a", "ab", "abc" },
        { "1", "12", "12345", "12345345", "234oi34iuwer" },
        { "cba", "abc", "bca", "a", "a", "a", "ab", "ab", "ab" } };
    assertEquals(mutant.thirdShortest(tests[0]), "abc");
    assertEquals(mutant.thirdShortest(tests[1]), "12345");
    assertTrue(mutant.thirdShortest(tests[2]).equals("cba") ||
        mutant.thirdShortest(tests[2]).equals("abc") ||
        mutant.thirdShortest(tests[2]).equals("bca"));
    assertThrows(IllegalArgumentException.class, () -> {
        mutant.thirdShortest(null) });
    assertThrows(IllegalArgumentException.class, () -> {
        mutant.thirdShortest(new String[]{}); });
    assertThrows(IllegalArgumentException.class, () -> {
        mutant.thirdShortest(new String[]{ "hey" }); });
    assertThrows(IllegalArgumentException.class, () -> {
        mutant.thirdShortest(new String[]{ "hey", "there" }); });
    String[] words = { "abcds", "b", "bc", "ab", "bye", "and" };
    String[] original = { "abcds", "b", "bc", "ab", "bye", "and" };
    result = mutant.thirdShortest(words);
    assertTrue(result.equals("bye") || result.equals("and"));
    assertEquals(Arrays.asList(words), Arrays.asList(original));
        …
```

7

Is This An Effective Test?

May be effective but hard to use
Tests are not independent
Will be hard to pinpoint bugs

```
@Test
public void testAll() {
    String[][] tests = { { "a", "ab
        { "1", "12", "12345", "1234
        { "cba", "abc", "bca", "a",
    assertEquals(mutant.thirdShortest(tests[0]), "abc");
    assertEquals(mutant.thirdShortest(tests[1]), "12345");
    assertTrue(mutant.thirdShortest(tests[2]).equals("cba") ||
        mutant.thirdShortest(tests[2]).equals("abc") ||
        mutant.thirdShortest(tests[2]).equals("bca"));
    assertThrows(IllegalArgumentException.class, () -> {
        mutant.thirdShortest(null) });
    assertThrows(IllegalArgumentException.class, () -> {
        mutant.thirdShortest(new String[] {}); });
    assertThrows(IllegalArgumentException.class, () -> {
        mutant.thirdShortest(new String[] { "hey" }); });
    assertThrows(IllegalArgumentException.class, () -> {
        mutant.thirdShortest(new String[] { "hey", "there" }); });
    String[] words = { "abcds", "b", "bc", "ab", "bye", "and" };
    String[] original = { "abcds", "b", "bc", "ab", "bye", "and" };
    result = mutant.thirdShortest(words);
    assertTrue(result.equals("bye") || result.equals("and"));
    assertEquals(Arrays.asList(words), Arrays.asList(original));
        …
```

8

Guidance for Writing Tests

- Group tests in methods, classes
  - Class could be by behavior, by error conditions, …
- Test methods should focus on one behavior
  - If test case fails, should be helpful in narrowing down where the problem is
- See examples on course schedule

9

Software Testing Issues

- How should you test? How often?
  - Code may change frequently
  - Code may depend on others' code
  - A lot of code to validate
- How do you know that an output is correct?
  - Complex output
  - Human judgment?
- What caused a code failure?

➥ Need a *systematic*, *automated*, *repeatable* approach

10

Software Testing Issues

- How do we know if our code is correct?
  - How do we know that we've exposed all the faults?
  - 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

Software Testing Issues

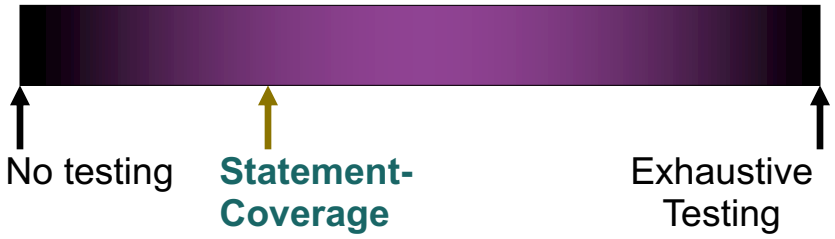
- How do we know if our code is correct?
  - How do we know that we've exposed all the faults?
  - How confident are we in its correctness?
- How do we know if we've tested enough?
  - Passes all of our TDD test suite
    - But did we come up with *all* the necessary test cases?
  - Time? It's been a couple hours/days/…
  - Number of test cases executed? A lot!
  - I asked my sister and she's really smart and she says that it's enough

Testing Continuum

- Give to customer immediately
- Likely buggy!
- Very little confidence in program's quality

- Test *every possible* input
- Costly, impractical
- Need to release application to customers *sometime!*

Testing Continuum



- Need to execute **all code**
- Cover (i.e., execute) all **statements** in the program

Analogy: Map coverage

15

Statement Coverage

- Cover all statements in the program

Test Suite:

num=5

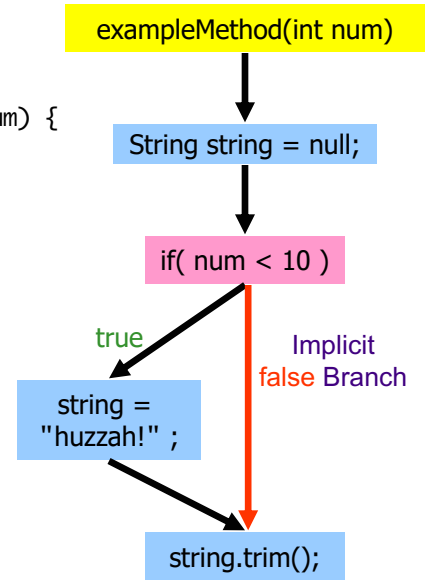
```
public String exampleMethod(int num) {
✓ 1   String string = null;
✓ 2   if (num < 10) {
✓ 3       string = "huzzah!";
      }
      // remove leading & trailing whitespace
✓ 4   return string.trim();
}
```

Is this method bug-free?

16

Program Flow

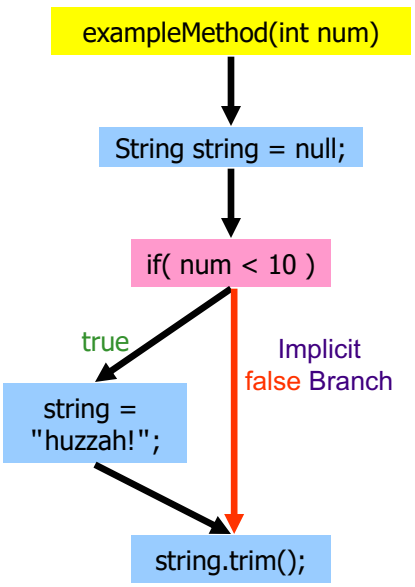
```
public String exampleMethod(int num) {
    String string = null;
    if (num < 10) {
        string = "huzzah!";
    }
    return string.trim();
}
```



17

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**



18

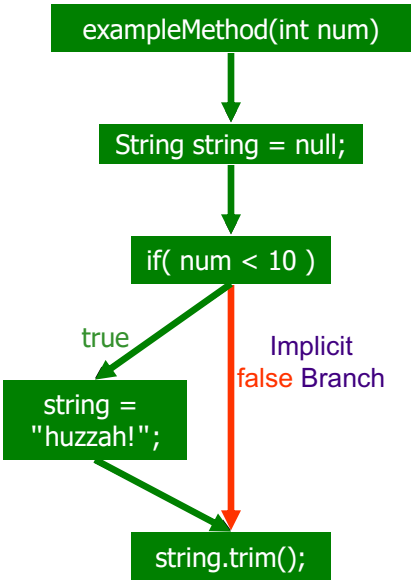
Branch Coverage

- Cover all **branches** in the program

Test Suite:

num=5,

num=10



19

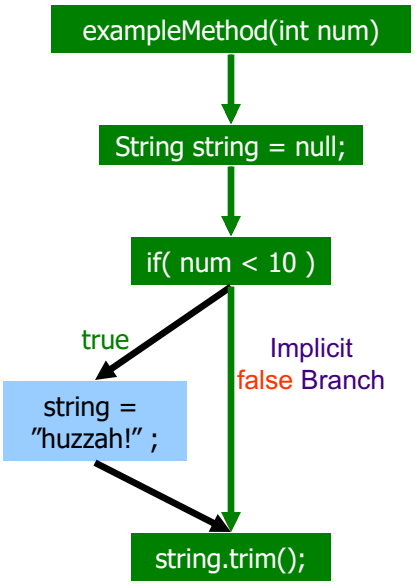
Branch Coverage

- Cover all **branches** in the program

Test Suite:

num=5,

num=10



20

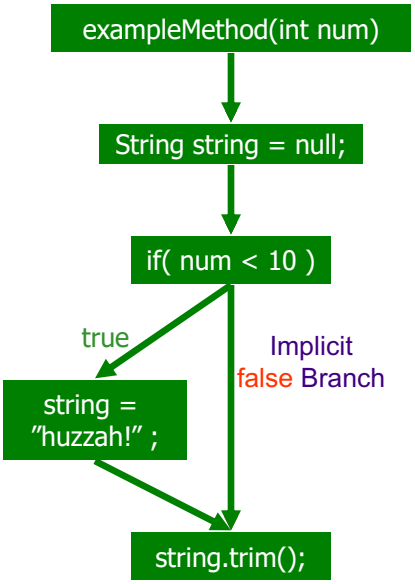
Branch Coverage

- Cover all **branches** in the program

Test Suite:

num=5,

num=10



Example 2

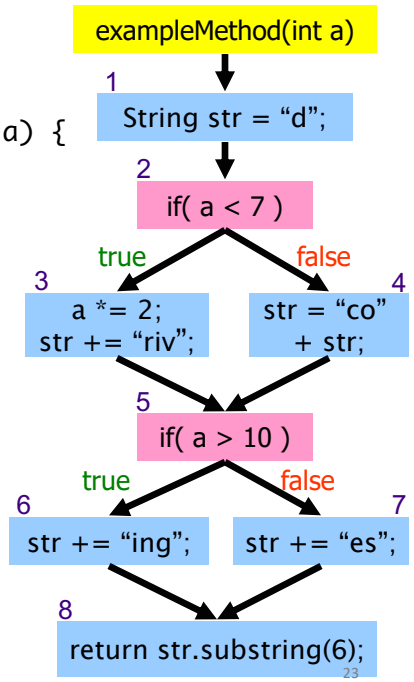
```
public static String exampleMethod(int a) {
   String str = "d";
   if ( a < 7 ) {
      a *= 2;
      str += "riv";
   } else {
      str = "co" + str;
   }

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

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);
}
```



Branch Coverage

Test Suite:

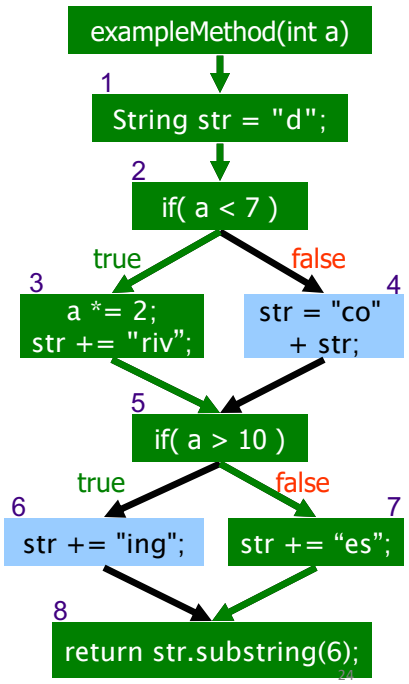
a=3,

a=30

str="driv"
a=6

str="drives"

""



Branch Coverage

Test Suite:

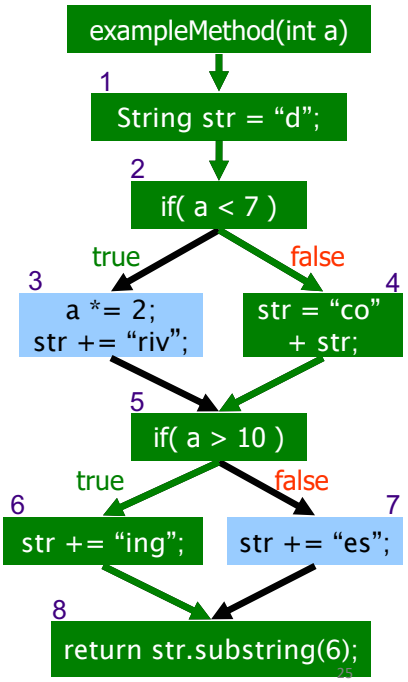
a=3,

a=30

str="cod"
a=30

str="coding"

""



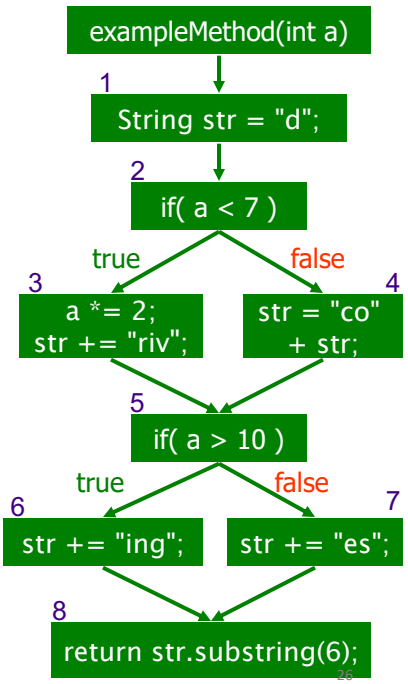
25

Branch Coverage

Test Suite:

a=3,

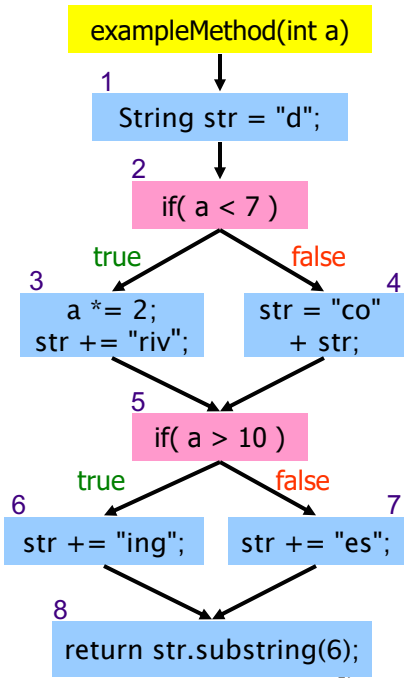
a=30



26

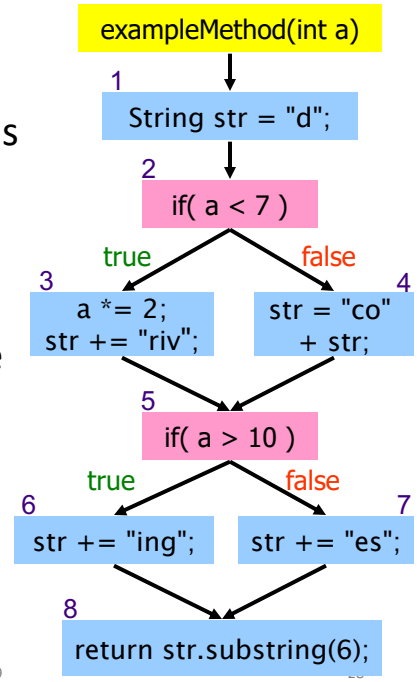
What Went Wrong?

- 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**



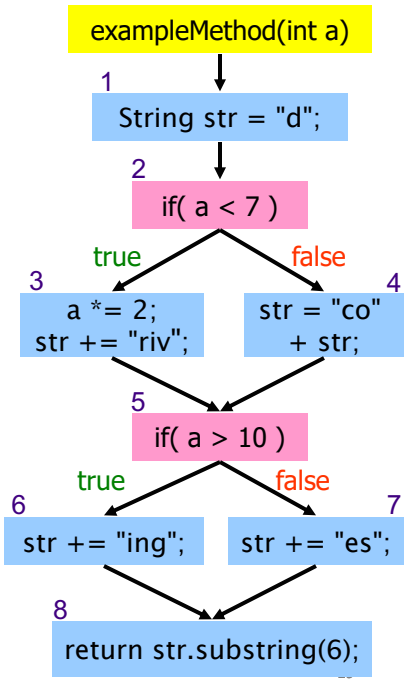
Path Coverage

- Cover all **paths** in program's flow
- How many paths through this method?
- What test cases would give us path coverage?



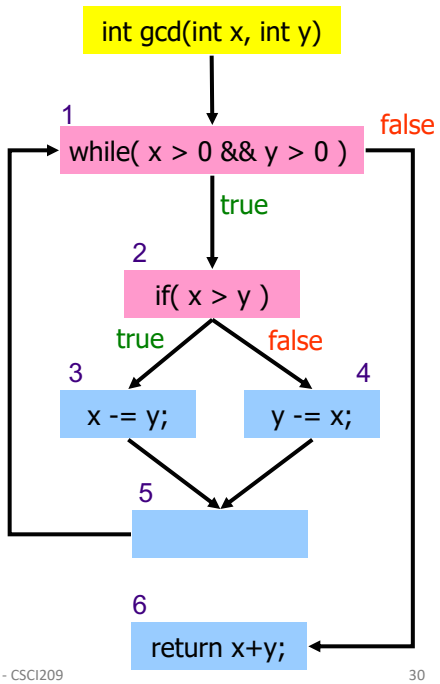
Path Coverage

- 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?



Example 3

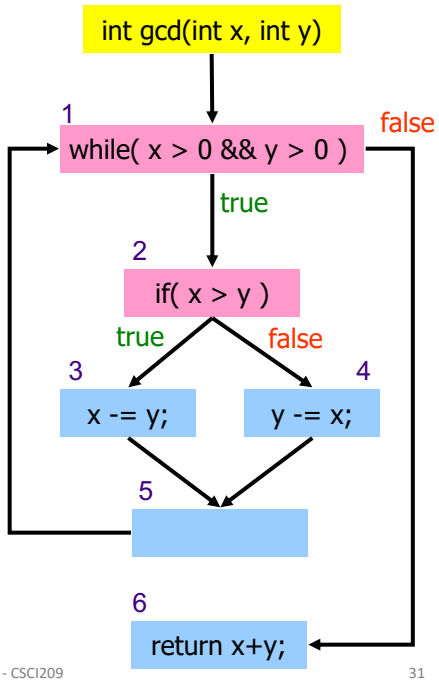
```
/**
 * Euclid's algorithm to
 * calculate greatest
 * common divisor
 */
public int gcd( int x, int y ) {
    while ( x > 0 && y > 0 ) {
        if( x > y ) {
            x -=y ;
        } else {
            y -=x;
        }
    }
    return  x+y;
}
```



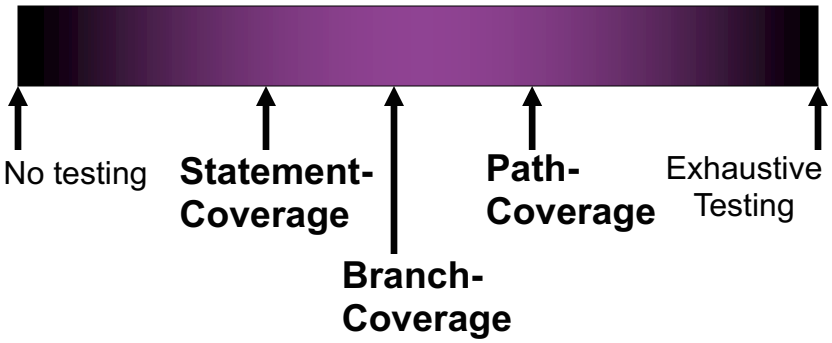
Path Coverage

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

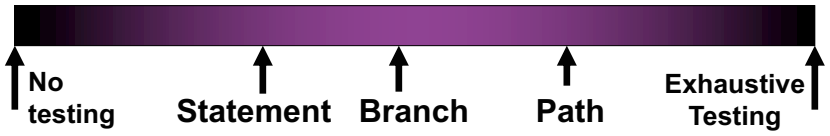
1-6
1-2-3-5-1-6
1-2-4-5-1-6
1-2-3-5-1-2-3-5-1-6
1-2-4-5-1-2-4-5-1-6
1-[2-(3|4)-5-1]*-6



Testing Continuum



Comparison of Coverage Criteria



Coverage Criterion	Advantages	Disadvantages
Statement		
Branch		
Path		

Looking Ahead

- Friday
  - Coverage tools, Debugger
- Monday
  - Testing project