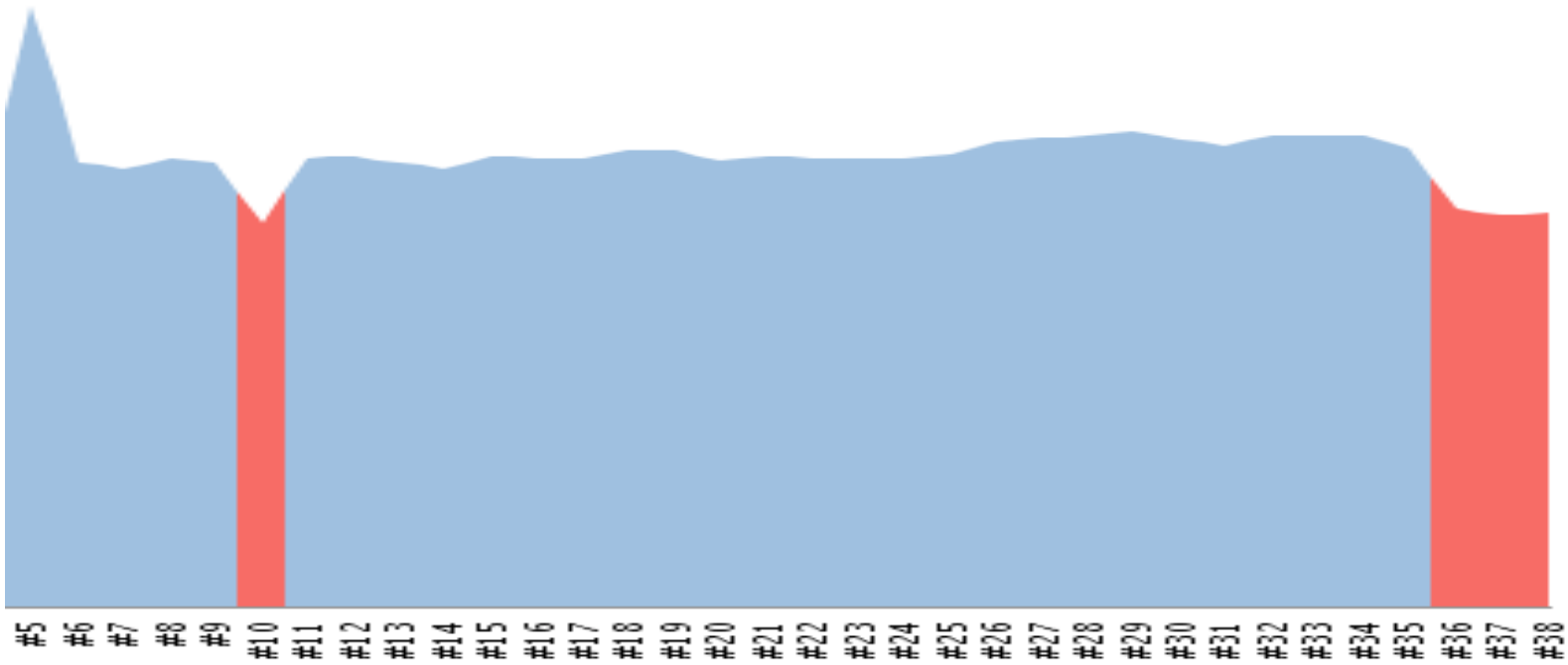
Software Quality FS 2016

Discussion Exercise 2

Eya Ben Charrada
charrada@ifi.uzh.ch

University of Zurich[UZH]

ImageJ
Builds



ImageJ

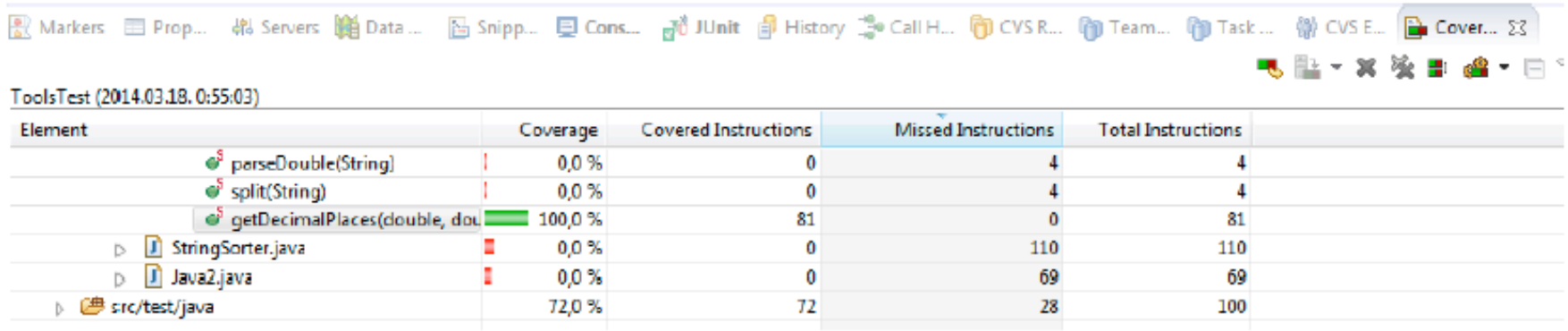
- Reports changes before closing a ticket
- Comments

Testing strategies

- Black-box:
  - Boundary values
  - Equivalence partitioning
- White-box:
  - Coverage (branch, statement,…)
- Documentation challenge

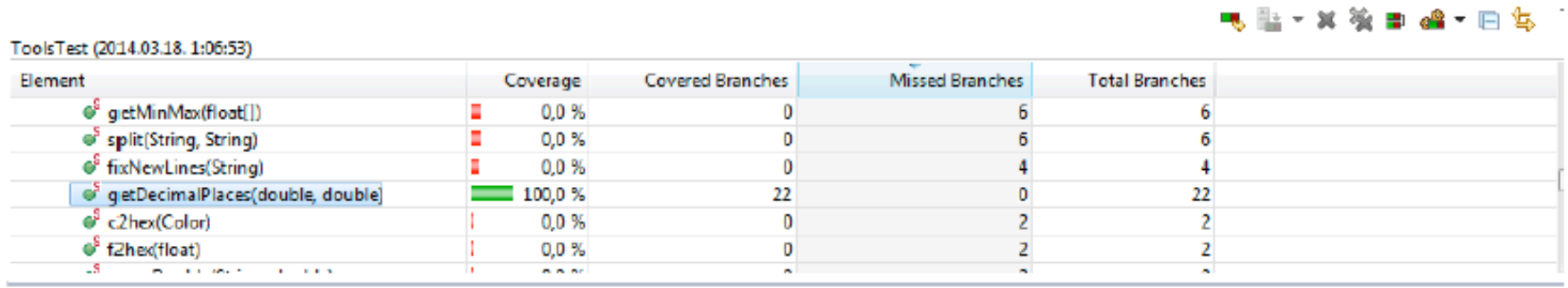
Testing
Coverage Tools

Instruction metric coverage summary:



Element	Coverage	Covered Instructions	Missed Instructions	Total Instructions
parseDouble(String)	0,0 %	0	4	4
split(String)	0,0 %	0	4	4
getDecimalPlaces(double, dou	100,0 %	81	0	81
StringSorter.java	0,0 %	0	110	110
Java2.java	0,0 %	0	69	69
src/test/java	72,0 %	72	28	100

Branch metric coverage summary:



Element	Coverage	Covered Branches	Missed Branches	Total Branches
getMinMax(float[])	0,0 %	0	6	6
split(String, String)	0,0 %	0	6	6
fixNewLines(String)	0,0 %	0	4	4
getDecimalPlaces(double, double)	100,0 %	22	0	22
c2hex(Color)	0,0 %	0	2	2
f2hex(float)	0,0 %	0	2	2

JUnit

```
@Test
public void testCanonicalColorNames(){
        assertEquals(Color.red, Colors.decode("red", Color.gray));
        assertEquals(Color.blue, Colors.decode("blue", Color.gray));
        assertEquals(Color.cyan, Colors.decode("cyan", Color.gray));
}
```

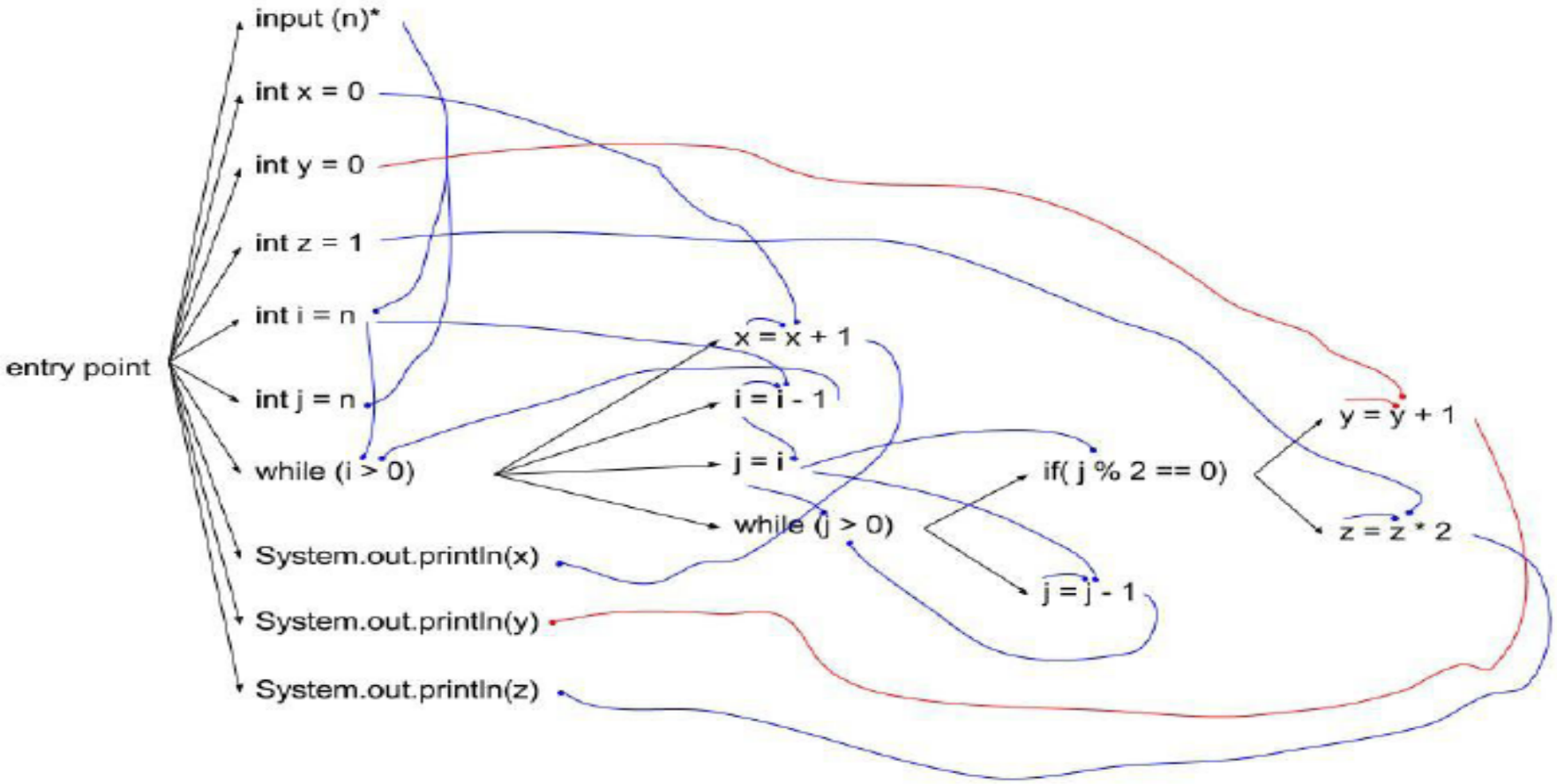
Only the first failure is reported by a test method

JUnit

```
public class MyTestCase {

    @Before
    public void setUp() {
    // Set up for the test
    }
    @Test
    public void testCondition1() {
    assertTrue(condition1);
    }
    @Test
    public void testCondition2() {
    assertTrue(condition2);
    }
}
```

Dependencies
What's wrong?



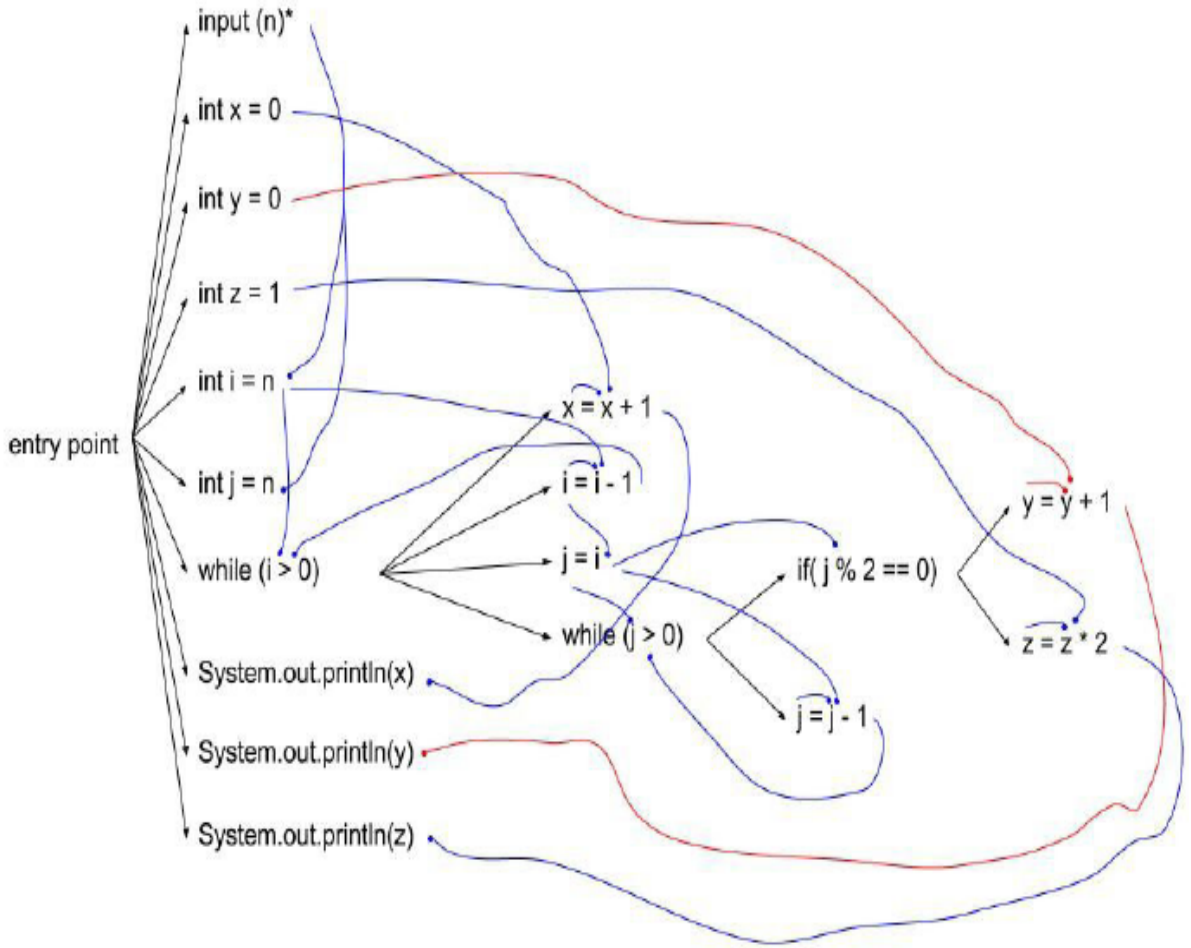
Dependencies
Slicing

Forward slice: Which parts may be affected?

Backward slice: A version of the original program with some parts missing, can be compiled and executed.

[Source: http://www.cs.ucl.ac.uk/staff/mharman/exe1.html]

Dependencies
Slicing




```
public static void calculateXYZ(int n){
  int x = 0;
  int y = 0;
  int z = 1;
  int i = n;
  int j = n;
  while (i>0){
    x = x + 1;
    i = i - 1;
    j = i;
    while (j>0){
      if(j%2 == 0)
        y = y + 1;
      else
        z = z * 2;
      j = j- 1;
    }
  }
  System.out.println(x);
  System.out.println(y);
  System.out.println(z);
}
```

Hypothesizing about a defect

Hypothesis	Test Cases	Result	Finding
Uneven numbers don't work	Input: 11 Input: 77 Input: 37 Input: 35	Output: 1*1 Output: 7*1, 1*1 Output: 1*1 Output: 5*1, 1*1	True

Hypothesizing about a defect

H5: The program does not work for the odd numbers	7 9 27 455 929	✗ ✓ ✓ ✗ ✗	Many odd numbers were tested within the range and it did not derive the correct result for a lot of odd numbers, except it gave correct result for the ones which are perfect square or cube numbers.	H5: *Failure*
H6: The program does not work for the even numbers	4 18 512 756 998	✓ ✗ ✓ ✗ ✗	Many even numbers were tested within the range and it did not derive the correct result for a lot of even numbers, except it gave correct result for the ones which are perfect square or cube numbers.	H6: *Failure*
H7: The program does not work for the decimal point numbers	7.2 15.7 88.9 763.8 999.8	✗ ✗ ✗ ✗ ✗	Many decimal point numbers were tested within the range and it did not derive the correct result for any decimal number	H7: *Failure*

Exam

- **Location**: BIN 2.A.10
- **Duration**: 90 minutes
- **Scope**:
  - Lecture's slides
  - Exercises
- **Cheat sheet**: 1 double-sided handwritten A4 page

Exam
Structure

- 1/3 knowledge questions
- 1/3 application
- 1/3 essay and writings

Sample exam is available on the lecture's website