Software Quality FS 2014

Discussion Exercise 2

Eya Ben Charrada

charrada@ifi.uzh.ch

2.B.17

University of Zurich[UZH]

ImageJ

- A few build fails
- One ticket resolved as *worksforme*
- Commit comments

SVN
structure

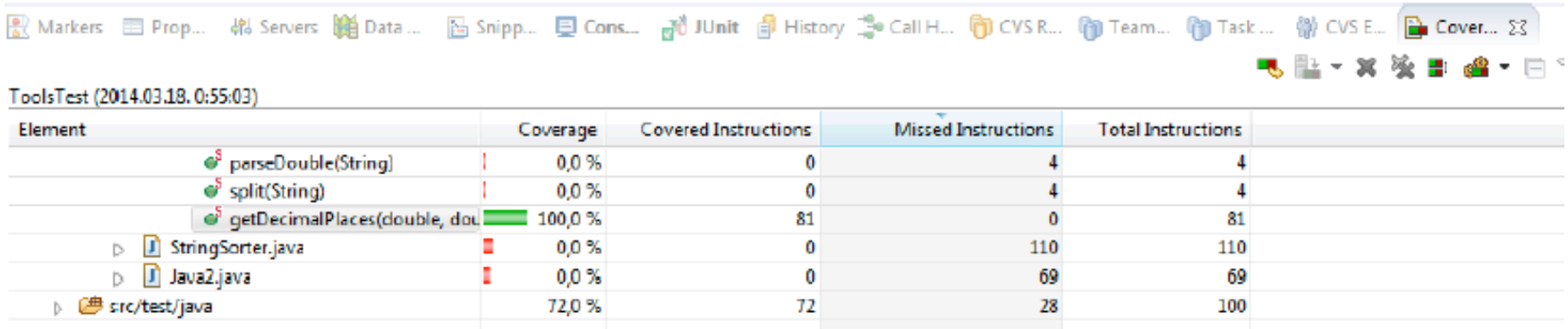
- Trunk
- Branches
- Tags

Testing strategies

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

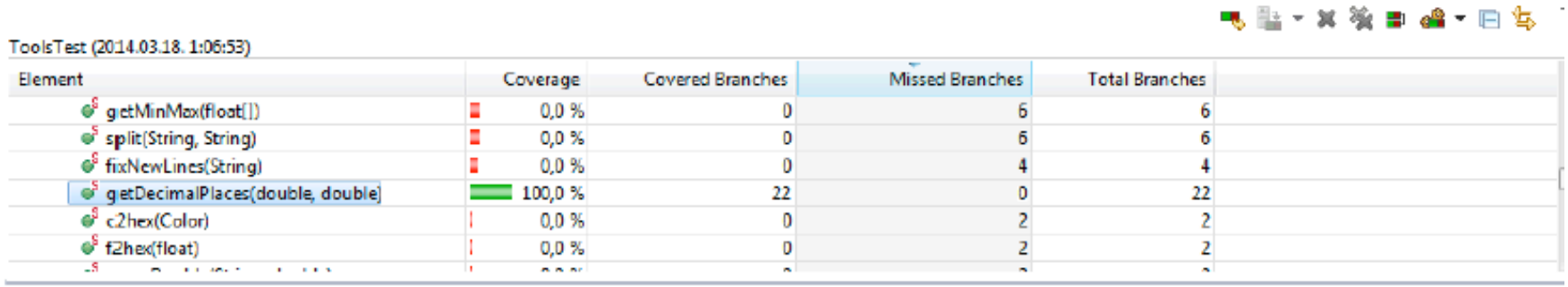
Coverage

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 testHexToColor() {
        ……
        assertEquals("ColorTest 1 successes", 8, successor);
        ……
        // Test 2: if # does not lead to conflicts
        assertEquals("ColorTest 2 successes", 8, successor);
        // Test 3: if there is just some input it should lead to null
        assertNull("ColorTest 3 successes",
colorTester.hexToColor("Just some Input"));
}
```

Only the first failure is reported by a test method

JUnit

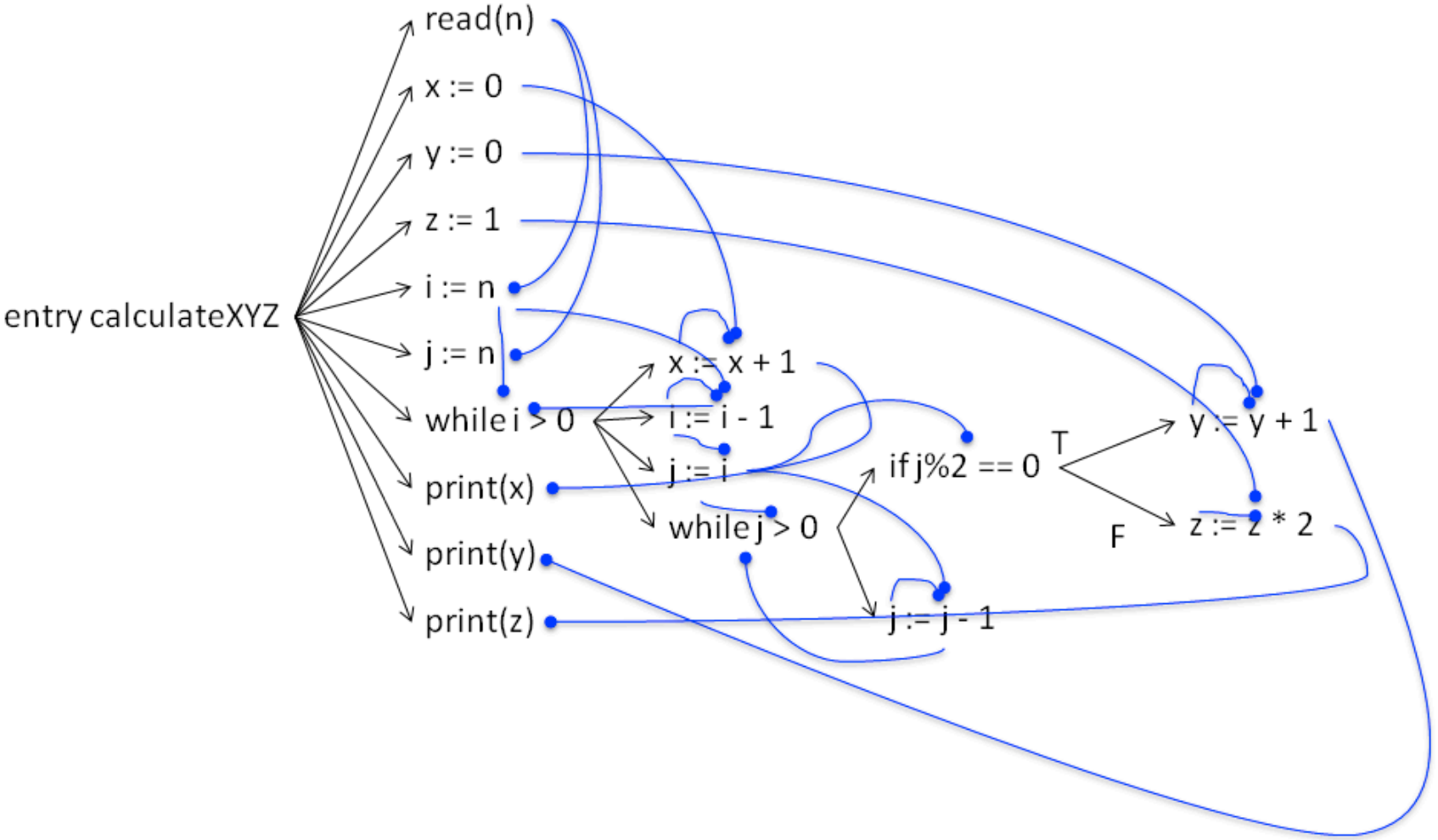
```
public class MyTestCase {

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

JUnit

```
for (int i = 0; i < input.length; i++) {
    if (colorTester.hexToColor(input[i]).equals(output[i])) {
    successor++;
    }
}
assertEquals("ColorTest 2 successes", 8, successor);
```

Dependencies

What's wrong?



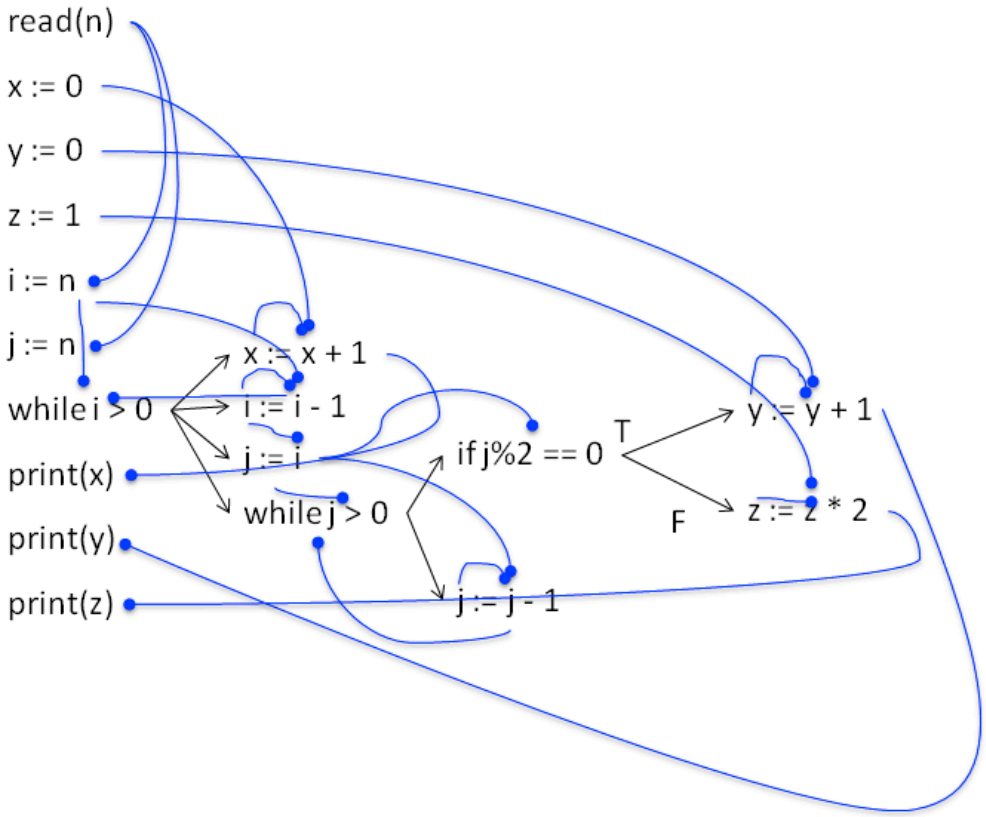
Dependencies
Slicing

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 8: For numbers with two or more trailing zeroes (i.e. which are divisible by 100) the program returns the inverse of the number but starting with 1 instead of 0
Prediction: Entering 34560000 yields 10006543 as a result
Experiment:
java -jar Inverse.jar 34560000
Output: 10006543
-> Hypothesis proven

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