

# Software Quality FS 2010

## Exercise 2 - Discussion

Cédric Jeanneret

Requirements Engineering Research Group

Department of Informatics

University of Zurich

<http://www.ifi.uzh.ch/reqg/people/jeanneret>



University of Zurich

Department of Informatics

# Outline

---

- Frequent problems in exercise 2
- Introduction to exercise 3
- Formalities for the exam

# Exercise 2

---

- In general, well solved
- Good discipline within the development environment
  - 1 broken build, fixed within 10 minutes
  - 1 ticket left open
  - (useful) commit comments
- ImageJ's quality in use VS ImageJ's internal quality
- <http://imagejdev.org/>

# Exercise 2.2

## Modularity and Testing

---

### Modularity:

- Allows decomposition of a system into simpler pieces & understanding that system in terms of these pieces
- Confines the search for a fault / an enhancement to a single module
  - “Heavy reading and browsing of the original source code was required to even understand how to setup a test and then run it. ”*
- Drives the testing process: unit tests, integration tests, system tests
- Implementation units  $\neq$  architectural components
- Allows the composition of systems from pieces (**reuse**)
- **MVC** is only one possible pattern for decomposing applications

# Exercise 2.3

## JUnit: 1 Test case = 1 Test method

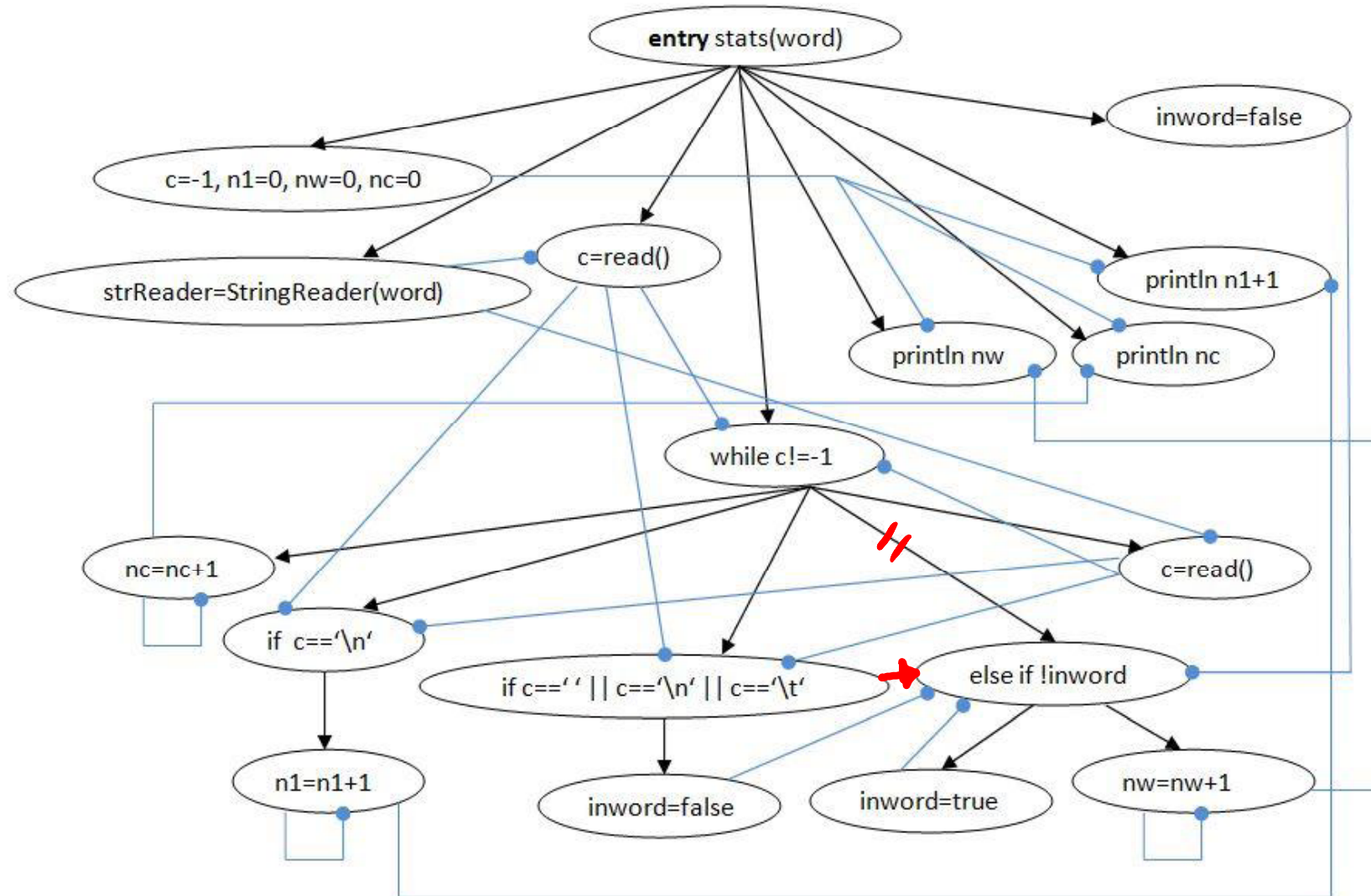
The screenshot displays an IDE window with the following components:

- Code Editor:** Shows the source code for `C2HexTest.java`. The code includes a class with three test methods: `testPairwise()`, `testBoundaries()`, and `testWithoutAlphaValue()`. Each test method contains multiple assertions using `assertEquals` to verify the output of the `Tools.c2hex` method for various color inputs.
- JUnit Runner:** Located on the right, it shows the execution results. The runner indicates that the tests finished after 0.03 seconds, with 3/3 runs, 0 errors, and 0 failures. A green progress bar is visible.
- Test Results:** A tree view shows the test methods and their execution times: `testPairwise (0.010 s)`, `testBoundaries (0.000 s)`, and `testWithoutAlphaValue (0.000 s)`.
- Failure Trace:** A section at the bottom right, currently empty, is used to display any test failures.

```
public class C2HexTest {  
  
    int min = 0;  
    int med = 121;  
    int max = 255;  
  
    @Before  
    public void setUp() {  
  
    }  
  
    @After  
    public void tearDown() {  
  
    }  
  
    @Test  
    public void testPairwise() {  
        assertTrue(Tools.c2hex(new Color(min, min, max, min)).equals("#0000FF"));  
        assertTrue(Tools.c2hex(new Color(med, max, med, max)).equals("#79FF79"));  
        assertTrue(Tools.c2hex(new Color(max, med, min, med)).equals("#FF7900"));  
        assertTrue(Tools.c2hex(new Color(min, med, max, max)).equals("#0079FF"));  
        assertTrue(Tools.c2hex(new Color(med, min, min, max)).equals("#790000"));  
        assertTrue(Tools.c2hex(new Color(max, min, med, min)).equals("#FF0079"));  
        assertTrue(Tools.c2hex(new Color(min, max, med, med)).equals("#00FF79"));  
        assertTrue(Tools.c2hex(new Color(med, med, max, med)).equals("#7979FF"));  
        assertTrue(Tools.c2hex(new Color(max, max, min, min)).equals("#FFFF00"));  
        assertTrue(Tools.c2hex(new Color(max, max, max, max)).equals("#FFFFFF"));  
        assertTrue(Tools.c2hex(new Color(med, med, med, min)).equals("#797979"));  
        assertTrue(Tools.c2hex(new Color(min, min, min, med)).equals("#000000"));  
    }  
  
    @Test  
    public void testBoundaries() {  
        assertTrue(Tools.c2hex(new Color(min, min, min, min)).equals("#000000"));  
        assertTrue(Tools.c2hex(new Color(max, max, max, max)).equals("#FFFFFF"));  
    }  
  
    @Test  
    public void testWithoutAlphaValue() {  
        assertTrue(Tools.c2hex(new Color(med, med, med)).equals("#797979"));  
    }  
}
```

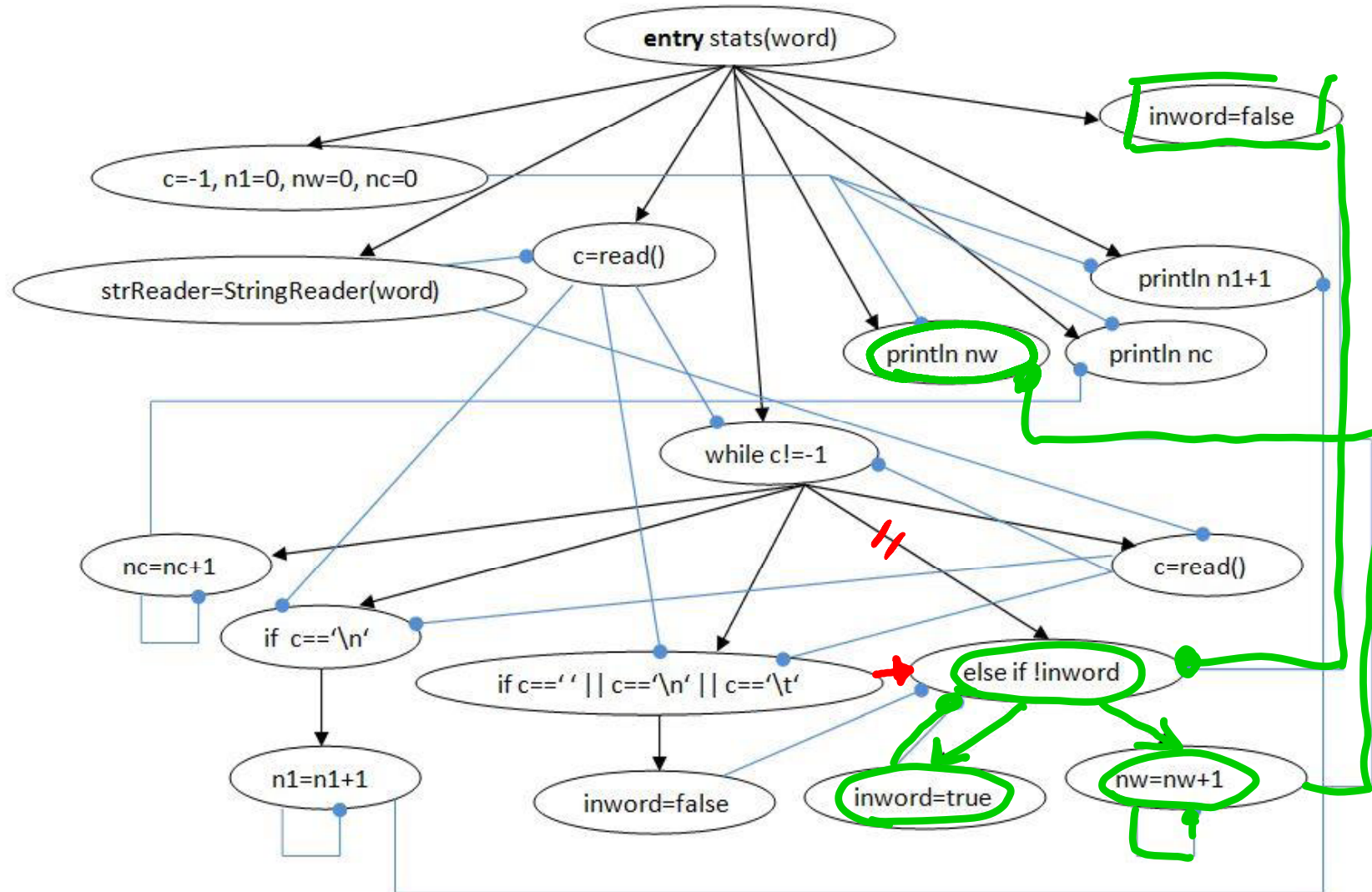
# Exercise 2.5

## Static Analysis – PDG



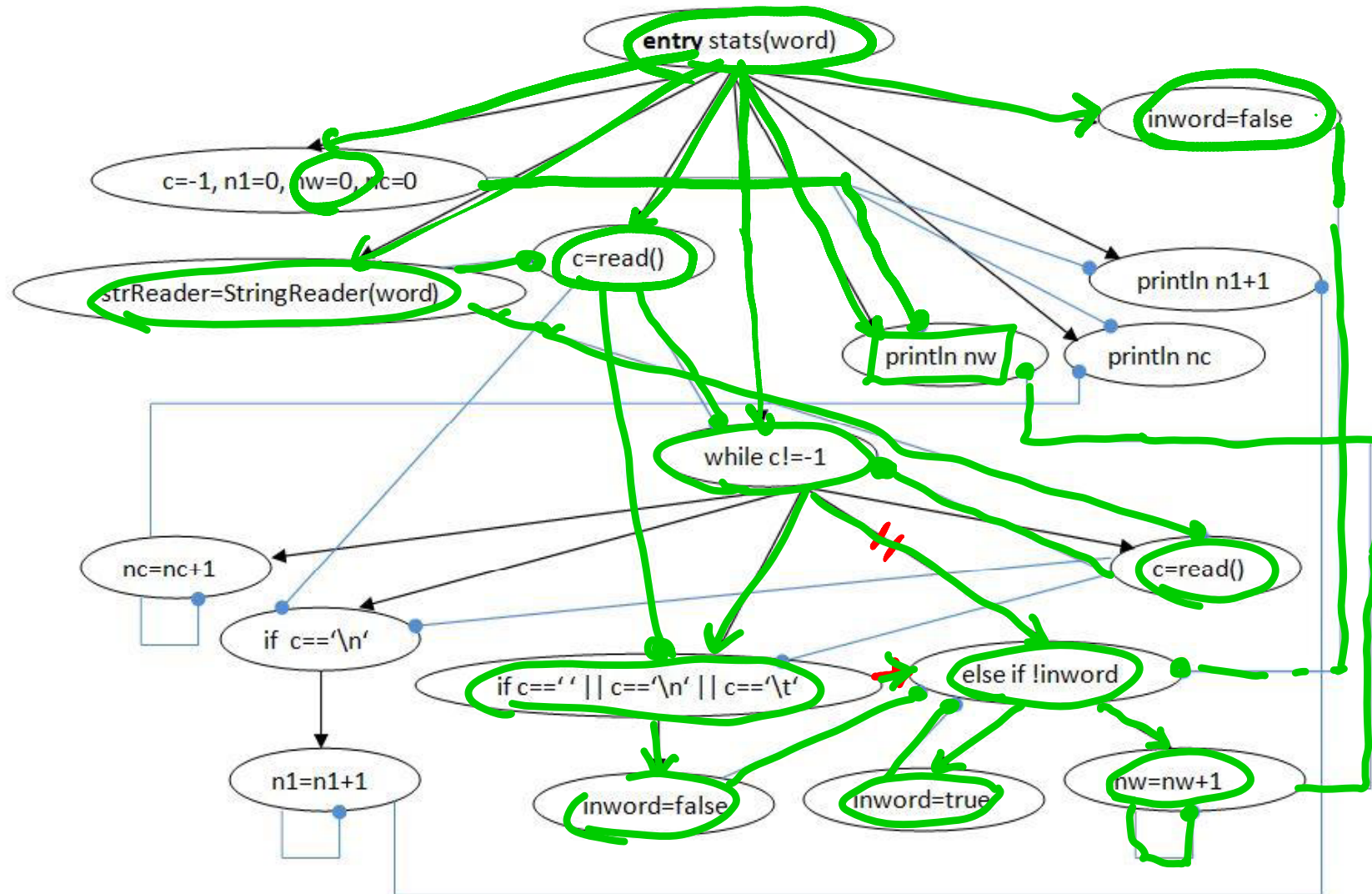
# Exercise 2.5

## Static Analysis – Forward Slice



# Exercise 2.5

## Static Analysis – Backward Slice





# Exercise 2.6

## Hypothesizing about a defect

---

Hypothesis 1: All numbers are translated with a unit (1) less.

#	Test Input	Expected Output	Actual Output	Test Result
1	9	IX	IX	PASS
2	16	XVI	XV	FAIL
3	21	XXI	XXI	PASS

All tests should fail. **Reject** hypothesis 1, because of test cases 1 and 3.

# Exercise 3

---

- Defining & Achieving Quality
  - Use of standard ISO / IEC 9126-1
    - Printed copies available in my office (BIN 2.B.17)
- Wiki contributions
  - Tutorials and sample solutions
  - Peer-reviewed during the assignment
    - Stay on schedule!

# Exam

---

Location: BIN 2.A.10

Date: Monday June 7, 2pm

Duration: 90 minutes

Language: German

Structure: 1/3 MCQ, 1/3 Case study and 1/3 Essay

Sample exam is available on the lecture's website

Scope: Lecture's slides + Exercises

Cheat sheet: 1 double-sided handwritten A4 page