



UNIVERSITY OF GOTHENBURG

Exercise 4: Structural Testing

Gregory Gay DIT635 - February 21, 2020





Finish In-Class Activities First!





The Planning System Returns

- Everybody likes meetings.
 - Not true but we need to book them.
- We don't want to double-book rooms or employees for meetings.
- System to manage schedules and meetings.







Structural Testing

- You already tested this system based on the functionality. Now we want to fill in the gaps.
- Goal: 100% Statement Coverage (Line Coverage) of all classes except Main and the exceptions.
 - First, measure coverage of your existing tests
 - Then, fill in any gaps with additional tests targeting the missed code.





Measuring Coverage

- The easiest way: use an IDE plug-in.
 - Eclipse: EclEmma <u>https://www.eclemma.org/</u>
 - IntelliJ: IntelliJ IDEA code coverage runner: <u>https://www.jetbrains.com/help/idea/code-coverage.html</u>
- Command line:
 - Emma, Cobertura offer executable tools.
 - JaCoCo available as a Maven plug-in: <u>https://medium.com/capital-one-tech/improve-java-code-</u> <u>with-unit-tests-and-jacoco-b342643736ed</u>





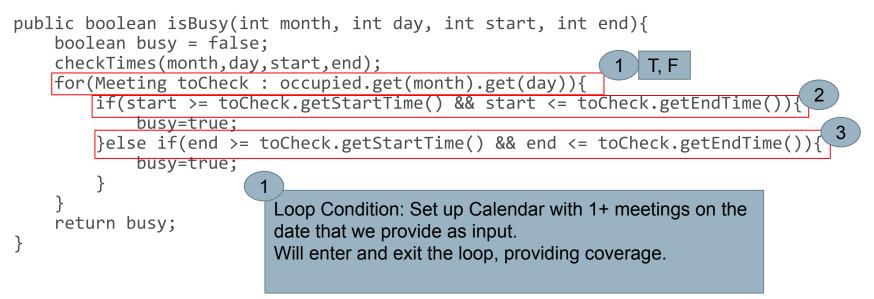


- If tests from last week don't get 100% line coverage.
- Target methods from each class using one of the coverage criteria from class.
 - Recommendation: Branch Coverage
 - Skip Main and exception.
- If you find code that cannot be covered, explain why.





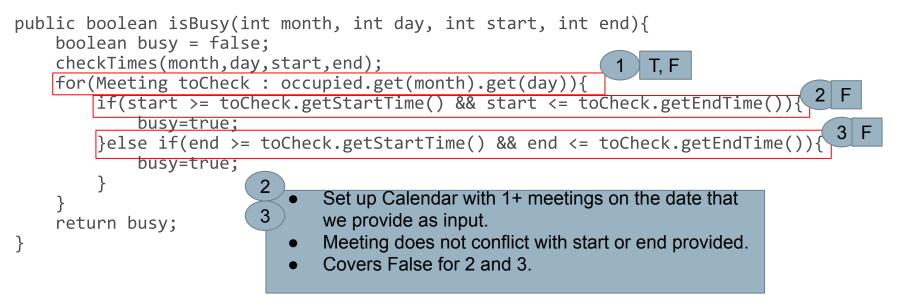








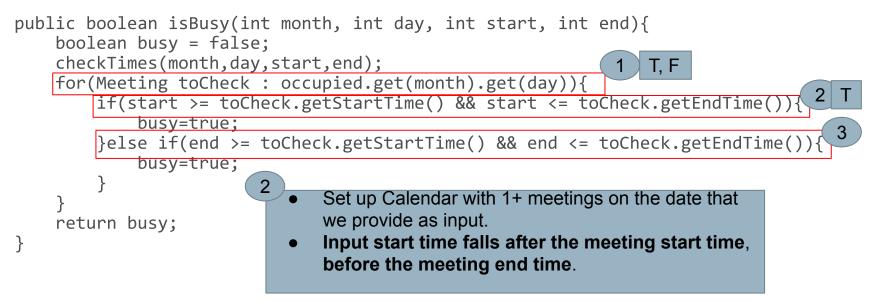








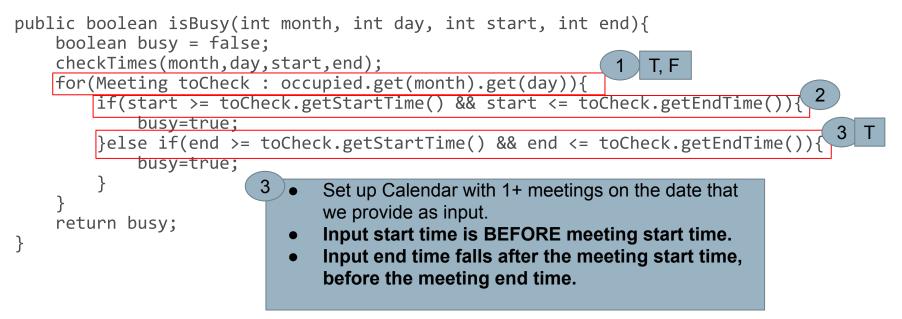














@Test

}

}

```
public void testIsBusyCoverage_1TF_2F_3F() {
      // Meeting with no conflict with our dates.
      Meeting noConflict = new Meeting(1,13,1,3);
      Calendar calendar = new Calendar();
      // Add meeting to calendar
      try {
             calendar.addMeeting(noConflict);
             // Enter a time that has no conflict.
             // Covers branches 1TF, 2F, 3F
             boolean result = calendar.isBusy(1, 13, 14, 16);
             assertFalse("Should cause no conflict", result);
      } catch(TimeConflictException e) {
```

fail("Should not throw exception: " + e.getMessage());

- Set up Calendar with 1+ meetings on the date that we provide as input.
- Meeting does not conflict with start or end provided.





@Test

```
public void testIsBusyCoverage_1TF_2T() {
```

```
Meeting noConflict = new Meeting(1,13,1,3);
```

```
Calendar calendar = new Calendar();
```

// Add meeting to calendar

try {

}

}

```
calendar.addMeeting(noConflict);
```

```
// Start time will fall after meeting start time
```

// and before meeting end time

```
// Covers branches 1TF, 2T
```

boolean result = calendar.isBusy(1, 13, 2, 3);

assertTrue("Should be a conflict with start time", result);

```
} catch(TimeConflictException e) {
```

```
fail("Should not throw exception: " + e.getMessage());
```

- Set up Calendar with 1+ meetings on the date that we provide as input.
- Input start time falls after the meeting start time, before the meeting end time.





@Test

```
public void testIsBusyCoverage_1TF_2F_3T() {
```

```
Meeting noConflict = new Meeting(1,13,2,4);
```

```
Calendar calendar = new Calendar();
```

// Add meeting to calendar

try {

}

}

```
calendar.addMeeting(noConflict);
```

// Start time will fall before meeting start time

```
    Set up Calendar with 1+ meetings
on the date that we provide as input.
```

- Input start time is BEFORE meeting start time.
- Input end time falls after the meeting start time, before the meeting end time.

// End time will fall after meeting start time, before end time

```
// Covers branches 1TF, 2F, 3T
```

```
boolean result = calendar.isBusy(1, 13, 1, 3);
```

```
assertTrue("Should be a conflict with end time", result);
```

```
} catch(TimeConflictException e) {
```

```
fail("Should not throw exception: " + e.getMessage());
```



UNIVERSITY OF GOTHENBURG



UNIVERSITY OF TECHNOLOGY