



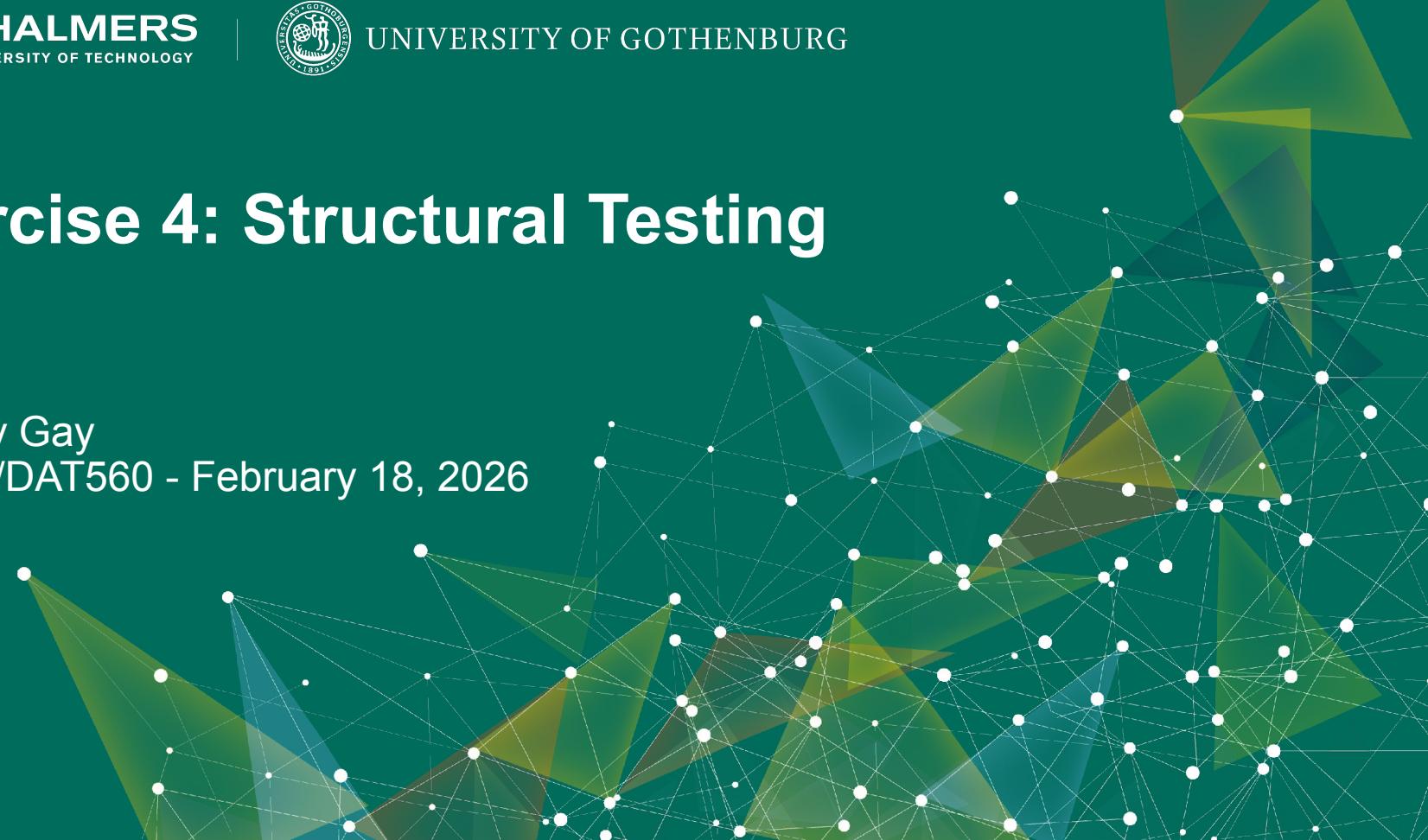
CHALMERS
UNIVERSITY OF TECHNOLOGY



UNIVERSITY OF GOTHENBURG

Exercise 4: Structural Testing

Gregory Gay
DIT636/DAT560 - February 18, 2026





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.
 - IntelliJ: IntelliJ IDEA code coverage runner:
<https://www.jetbrains.com/help/idea/code-coverage.html>
 - VSCode:
https://code.visualstudio.com/docs/debugtest/testing#_test-coverage
- Command line:
 - Cobertura
 - JaCoCo available as a Maven plug-in:
<https://medium.com/capital-one-tech/improve-java-code-with-unit-tests-and-jacoco-b342643736ed>



Activity

- 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: Target Branch Coverage
 - **Skip PlannerInterface and exception.**
- If you find code that cannot be covered, explain why.
- If you feel some code doesn't need covered, explain.



Example

From Calendar:

```
public boolean isBusy(int month, int day, int start, int end){  
    boolean busy = false;  
    checkTimes(month,day,start,end);  
    for(Meeting toCheck : occupied.get(month).get(day)){  
        if(start >= toCheck.getStartTime() && start <= toCheck.getEndTime()){  
            busy=true;  
        }else if(end >= toCheck.getStartTime() && end <= toCheck.getEndTime()){  
            busy=true;  
        }  
    }  
    return busy;  
}
```

1 T, F

2

3

1 Loop Condition: Set up Calendar with 1+ meetings on the date that we provide as input.
Will enter and exit the loop, providing coverage.



Example

From Calendar:

```
public boolean isBusy(int month, int day, int start, int end){  
    boolean busy = false;  
    checkTimes(month,day,start,end);  
    for(Meeting toCheck : occupied.get(month).get(day)){  
        if(start >= toCheck.getStartTime() && start <= toCheck.getEndTime()){  
            busy=true;  
        }else if(end >= toCheck.getStartTime() && end <= toCheck.getEndTime()){  
            busy=true;  
        }  
    }  
    return busy;  
}
```

1 T, F 2 F 3 F

2 3

- Set up Calendar with 1+ meetings on the date that we provide as input.
- Meeting does not conflict with start or end provided.
- Covers False for 2 and 3.



Example

From Calendar:

```
public boolean isBusy(int month, int day, int start, int end){  
    boolean busy = false;  
    checkTimes(month,day,start,end);  
    for(Meeting toCheck : occupied.get(month).get(day)){  
        if(start >= toCheck.getStartTime() && start <= toCheck.getEndTime()){  
            busy=true;  
        }else if(end >= toCheck.getStartTime() && end <= toCheck.getEndTime()){  
            busy=true;  
        }  
    }  
    return busy;  
}
```

1 T, F

2 T

3

2

- 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.**



Example

From Calendar:

```
public boolean isBusy(int month, int day, int start, int end){  
    boolean busy = false;  
    checkTimes(month,day,start,end);  
    for(Meeting toCheck : occupied.get(month).get(day)){  
        if(start >= toCheck.getStartTime() && start <= toCheck.getEndTime()){  
            busy=true;  
        }else if(end >= toCheck.getStartTime() && end <= toCheck.getEndTime()){  
            busy=true;  
        }  
    }  
    return busy;  
}
```

1 T, F

2

3 T

3

- 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.**



```
@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(result, "Should cause no conflict");  
  
    } 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(result, "Should be a conflict with start time");  
    } 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  
        // 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(result, "Should be a conflict with end time");  
    } 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 is BEFORE meeting start time.**
- **Input end time falls after the meeting start time, before the meeting end time.**



UNIVERSITY OF
GOTHENBURG



CHALMERS
UNIVERSITY OF TECHNOLOGY