Class Testing

Helen Dodd

2nd December 2006
Table of Contents

- Background of Class Testing
  - What it is
  - Where it fits in
  - Who does it
  - Goals

- Windshield Wiper Example
  - Pseudocode
  - Methods for testing the program
  - Test Driver pseudocode
  - Pros/Cons
What Is Class Testing?

- Unit Testing for Object-Oriented Systems
  - Test all features of a class object
  - Units should be tested in isolation
  - Test sequences of methods
- Inheritance presents problems in testing
  - Flattened classes
- Units
  - The smallest chunk that can be compiled by itself
  - A single procedure/function
  - Something so small it would be developed by one person
- Classes and Methods = Units?
Functional or Structural?

- **Functional Testing**
  - Test methods as black boxes
  - Tests based on specification

- **Structural Testing**
  - ‘Set’ and ‘Get’ methods for attributes
When? Who? What?

- Unit Level
- Extreme Programming…
  - Test cases designed before coding begins
- More normally…
  - Test cases designed after coding is complete
- Tester ≠ Programmer
Goals

- Check attributes get set correctly
  - Initialised to the right value, eg:
    \[
    \text{sizeIndex} = [ 31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 30 ]
    \]
- Find errors in calculation
  - \( + \) instead of \( * \)
- Wrong method calls
  - \text{year.increment()}\) instead of \text{month.increment()}
- Redundant code
- Incorrect boundary values
  - \text{for (int} i = 0; i \leq 5; i++)\) VS \text{for (int} i = 0; i < 5; i++)\)
- Error Messages
- Program efficiency is not so important
Windshield Wiper Example

<table>
<thead>
<tr>
<th>c2.Lever</th>
<th>OFF</th>
<th>INT</th>
<th>INT</th>
<th>INT</th>
<th>LOW</th>
<th>HIGH</th>
</tr>
</thead>
<tbody>
<tr>
<td>c2.Dial</td>
<td>n/a</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>n/a</td>
<td>n/a</td>
</tr>
<tr>
<td>a1.Wiper</td>
<td>0</td>
<td>4</td>
<td>6</td>
<td>12</td>
<td>30</td>
<td>30</td>
</tr>
</tbody>
</table>

- Lever has four positions: OFF, INT, LOW and HIGH
- Dial is only relevant when lever is on INT
Windshield Wiper Pseudocode

```java
class windshieldWiper {
    private wiperSpeed
    private leverPosition
    private dialPosition

    windshieldWiper(wiperSpeed, leverPosition, dialPosition)

    getWiperSpeed()
    setWiperSpeed()

    getLeverPosition()
    setLeverPosition()

    getDialPosition()
    setDialPosition()

    senseLeverUp()
    senseLeverDown()

    senseDialUp()
    senseDialDown()

    End class windshieldWiper
```

- Maintain the state of lever and dial
- 'Sense' methods for lever and dial
- Get/Set operations for each variable
senseLeverUp()
    Case leverPosition Of
        Case 1: Off
            leverPosition = Int
        Case dialPosition Of
            Case 1: 1
                wiperSpeed = 4
            Case 2: 2
                wiperSpeed = 6
            Case 3: 3
                wiperSpeed = 12
        EndCase 'dialPosition
        Case 2: Int
            leverPosition = Low
            wiperSpeed = 30
        Case 3: Low
            leverPosition = High
            wiperSpeed = 60
        Case 4: High
            (impossible; error condition)
    EndCase 'leverPosition
Methods for testing Windshield Wiper

- Test the get/set methods
  - wiperSpeed
  - leverPosition
  - dialPosition
- Driver (main program)
- Test individual methods
- Test classes that don’t rely on others first
- Then test classes that use the already tested classes
- Stubs
  - Dummy subprograms
Test senseLeverUp

class testSenseLeverUp
    wiperSpeed
    leverPos
    dialPos
    testResult 'boolean
main()
testCase = instantiate windshieldWiper(0,Off,1)
windshieldWiper.senseLeverUp()
leverPos = windshieldWiper.getLeverPosition()
If leverPos = Int
    Then testResult = Pass
    Else testResult = Fail
EndIf
End 'main

<table>
<thead>
<tr>
<th>Test Case</th>
<th>Preconditions</th>
<th>Method</th>
<th>Expected Value of leverPos</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>windshieldWiper(0,Off,1)</td>
<td>senseLeverUp()</td>
<td>INT</td>
</tr>
<tr>
<td>2</td>
<td>windshieldWiper(0,Int,1)</td>
<td>senseLeverUp()</td>
<td>LOW</td>
</tr>
<tr>
<td>3</td>
<td>windshieldWiper(0,Low,1)</td>
<td>senseLeverUp()</td>
<td>HIGH</td>
</tr>
<tr>
<td>4</td>
<td>windshieldWiper(0,High,1)</td>
<td>senseLeverDown()</td>
<td>LOW</td>
</tr>
<tr>
<td>5</td>
<td>windshieldWiper(0,Low,1)</td>
<td>senseLeverDown()</td>
<td>INT</td>
</tr>
<tr>
<td>6</td>
<td>windshieldWiper(0,Int,1)</td>
<td>senseLeverDown()</td>
<td>OFF</td>
</tr>
</tbody>
</table>
Test windshield wiper

<table>
<thead>
<tr>
<th>Event Sequence</th>
<th>User Action</th>
<th>System Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>move lever to INT</td>
<td>Wiper speed is 4</td>
</tr>
<tr>
<td>2</td>
<td>move dial to 2</td>
<td>Wiper speed is 6</td>
</tr>
<tr>
<td>3</td>
<td>move dial to 3</td>
<td>Wiper speed is 12</td>
</tr>
<tr>
<td>4</td>
<td>move lever to LOW</td>
<td>Wiper speed is 20</td>
</tr>
<tr>
<td>5</td>
<td>move lever to INT</td>
<td>Wiper speed is 12</td>
</tr>
<tr>
<td>6</td>
<td>move lever to OFF</td>
<td>Wiper speed is 0</td>
</tr>
</tbody>
</table>
Test windshieldWiper Sequences

class testScenario
  wiperSpeed
  leverPos
dialPos
  step1OK 'boolean
  step2OK 'boolean
  step3OK 'boolean
  step4OK 'boolean
  step5OK 'boolean
  step6OK 'boolean

main()
  testCase = instantiate windshieldWiper(0,Off,1)
  wiperSpeed = windshieldWiper.getWiperSpeed()
  If wiperSpeed = 4
    Then step1OK = Pass
    Else step1OK = Fail
  EndIf

  wiperSpeed = windshieldWiper.getWiperSpeed()
  If wiperSpeed = 6
    Then step2OK = Pass
    Else step2OK = Fail
  EndIf

  wiperSpeed = windshieldWiper.getWiperSpeed()
  If wiperSpeed = 12
    Then step3OK = Pass
    Else step3OK = Fail
  EndIf

windshieldWiper.setWiperSpeed()

Pros/Cons of writing a test driver

- Automated testing
- Drivers and Stubs = Overhead
- Manual Testing
In Summary...

- Background of Class Testing
  - What it is
  - Where it fits in
  - Who does it
  - Goals

- Windshield Wiper Example
  - Pseudocode
  - Methods for testing the program
  - Test Driver pseudocode
  - Pros/Cons
Any Questions?