COMP I 10/L Lecture 9

Kyle Dewey

Outline

- @Test vs.assertEquals
- Boolean operations
 - & &
 - | |
 - !
- Complex if conditions

@Test vs. assertEquals

@Test vs. assertEquals

- @Test defines a test
- assertEquals checks a condition
- Can have a @Test containing no assertEquals
 - Test always passes
- Can have multiple assertEquals per @Test
 - Test passes if all assertEquals are ok

-Generally we want to define one assertEquals per @Test, but sometimes this is inconvenient

Example: MultiAssert.java MultiAssertTest.java

You're already familiar with operations returning boolean

You're already familiar with operations returning boolean

3 < 6

You're already familiar with operations returning boolean

You're already familiar with operations returning boolean

Can chain boolean expressions with AND (&&). Semantics: only true if both sides are true.

Can chain boolean expressions with AND (&&). Semantics: only true if both sides are true.

3 > 1 & 1 < 5

Can chain boolean expressions with AND (&&). Semantics: only true if both sides are true.

3 > 1 & 1 < 5

true

Can chain boolean expressions with AND (&&). Semantics: only true if both sides are true.

3 > 1 && 1 < 5 true

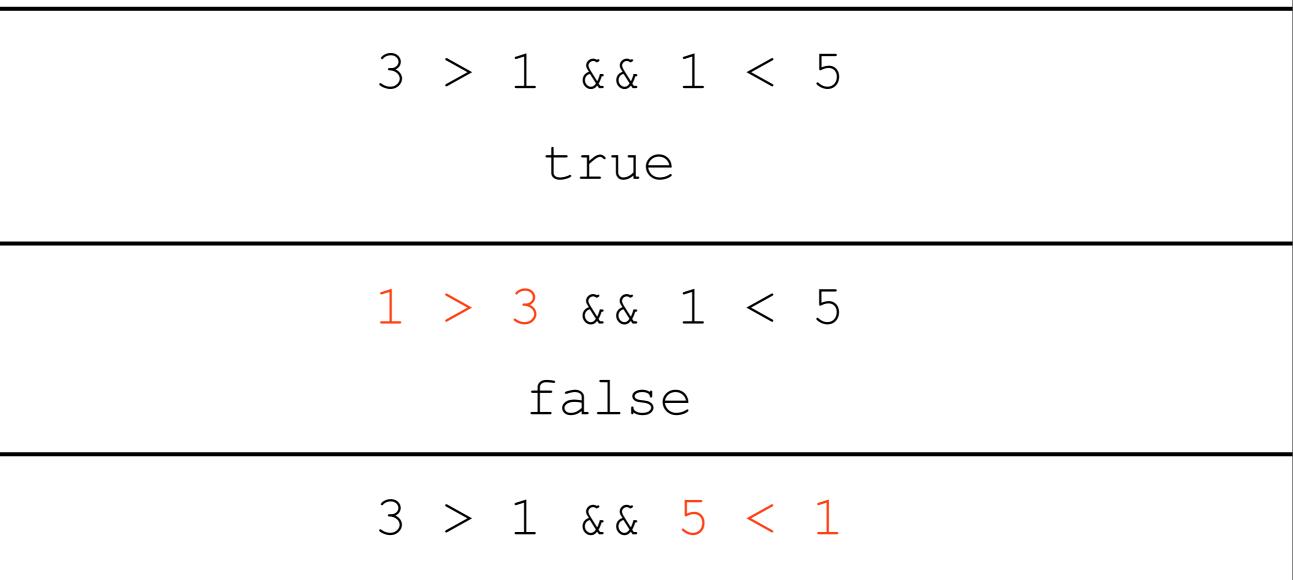
1 > 3 & 1 < 5

Can chain boolean expressions with AND (&&). Semantics: only true if both sides are true.

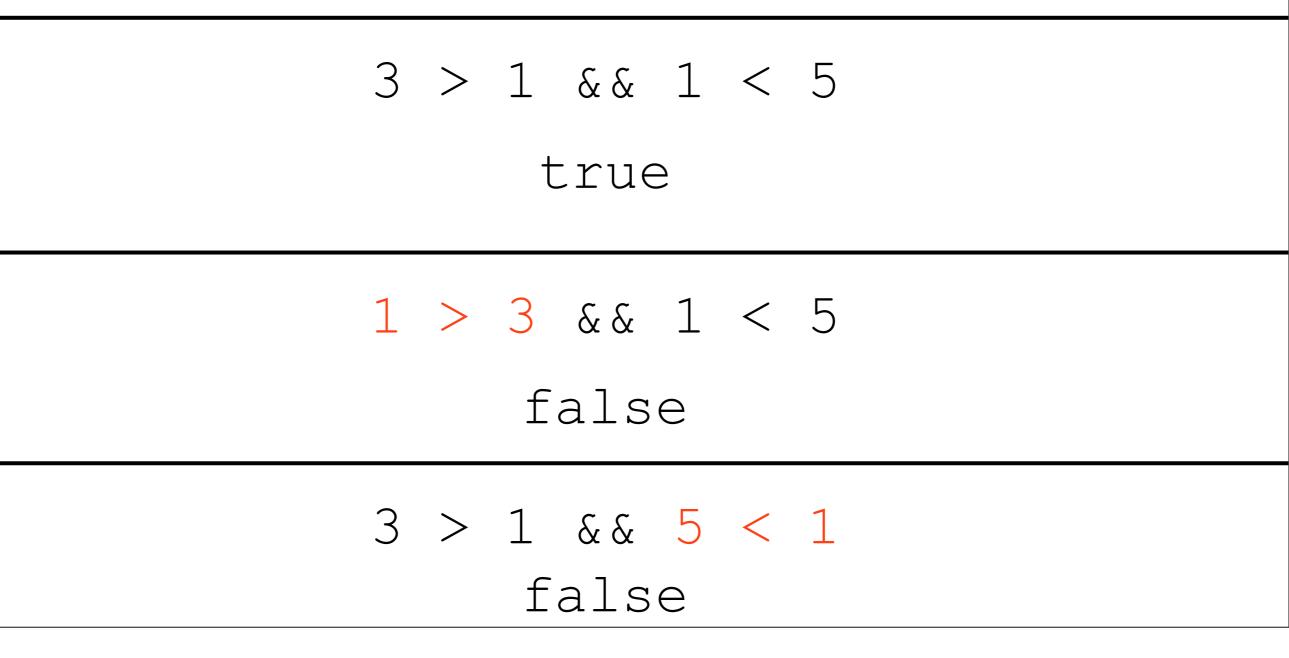
3 > 1 && 1 < 5 true

1 > 3 && 1 < 5 false

Can chain boolean expressions with AND (&&). Semantics: only true if both sides are true.



Can chain boolean expressions with AND (&&). Semantics: only true if both sides are true.



Example: And.java

boolean expressions can also be combined with OR (||) Semantics: true if either side is true.

boolean expressions can also be combined with OR (||) Semantics: true if either side is true.

3 > 1 || 5 < 1

boolean expressions can also be combined with OR (||) Semantics: true if either side is true.

3 > 1 || 5 < 1 true

boolean expressions can also be combined with OR (||) Semantics: true if either side is true.

3 > 1 || 5 < 1 true

2 < 1 || 8 < 9

boolean expressions can also be combined with OR (||) Semantics: true if either side is true.

3 > 1 || 5 < 1 true

2 < 1 || 8 < 9 true

boolean expressions can also be combined with OR (||) Semantics: true if either side is true.

3 > 1 || 5 < 1 true

2 < 1 || 8 < 9 true

2 < 1 || 9 < 8

boolean expressions can also be combined with OR (||) Semantics: true if either side is true.

3 > 1 || 5 < 1 true

2 < 1 || 8 < 9 true

2 < 1 || 9 < 8 false

Example: Or.java

Can negate a boolean expression with not (!).
Semantics: !true == false and !false == true.

Can negate a boolean expression with not (!).
Semantics: !true == false and !false == true.

! (1 < 2)

Can negate a boolean expression with not (!).
Semantics: !true == false and !false == true.

!(1 < 2) false

Can negate a boolean expression with not (!).
Semantics: !true == false and !false == true.

!(1 < 2) false

! (1 > 7)

Can negate a boolean expression with not (!).

Semantics: !true == false and !false == true.

!(1 < 2) false

! (1 > 7)

true

Can negate a boolean expression with not (!).

Semantics: !true == false and !false == true.

!(1 < 2) false

true

$$! (1 < 2 \& \& 1 > 3)$$

Can negate a boolean expression with not (!).
Semantics: !true == false and !false == true.

!(1 < 2) false

true

!(1 < 2 && 1 > 3) true

Example: Not.java

Putting it Together: ComplexConditional.java

Uses of & & and | | usually mean more tests are appropriate

Uses of & & and | | usually mean more tests are appropriate

- if (x == 1 || x == 5) { return 7;
- } else if (x > 7 && x <= 20) {
 return 8;</pre>
- } else {
 return 55;

}

Uses of & & and | | usually mean

more tests are appropriate

Test: x = 1

- if (x == 1 || x == 5) {
 return 7;
- } else if (x > 7 && x <= 20) {
 return 8;</pre>
- } else {
 return 55;

}

Uses of && and || usually mean

more tests are appropriate

Test: x = 1 Test: x = 5

- if (x == 1 | | x == 5) {

return 7;

- } else if (x > 7 && x <= 20) { return 8;
- } else { return 55;

}

Uses of && and || usually mean

more tests are appropriate

Test: x = 1 Test: x = 5
if (x == 1 || x == 5) {
 return 7; Test: x = 8
} else if (x > 7 && x <= 20) {
 return 8;
} else {
 return 55;
}</pre>

Uses of & & and | | usually mean

more tests are appropriate

Test: x = 1 Test: x = 5
if (x == 1 || x == 5) {
 return 7; Test: x = 8
} else if (x > 7 && x <= 20) {
 return 8;
} else {
 return 55; Test: x = 21
}</pre>

Putting it Together: ComplexConditionalTest.java