Advanced Conditionals

Mr. Fahrenbacher Presents

The Rules

- Don't EVER put a semicolon at the end of an if-case
- If you want multiple lines to be in an if-case, you need braces
- If you want to see if two things are equal, you must use two equals signs (==)

Other Rules

- You can't compare more than one thing at the same time

```
if(x == 3 || 4)
{
}
```

Syntax Error

```
if(x == 3 || x == 4)
{
}
```

Works

Negation Recap

!(x > 3) ⟶

!(x >= 5) ⟶

!(x == 1) ⟶

!(x != 2) ⟶

Negation Recap

!(x > 3) ⟶ x <= 3

!(x >= 5) ⟶

!(x == 1) ⟶

!(x != 2) ⟶

Negation Recap

!(x > 3) ⟶ x <= 3

!(x >= 5) ⟶ x < 5

!(x == 1) ⟶

!(x != 2) ⟶

Negation Recap

!(x > 3) ⟶ x <= 3

!(x >= 5) ⟶ x < 5

!(x == 1) ⟶ x != 1

!(x != 2) ⟶

Negation Recap

!(x > 3) ⟶ x <= 3

!(x >= 5) ⟶ x < 5

!(x == 1) ⟶ x != 1

!(x != 2) ⟶ x == 2

Demorgan's Law

- Rule for equivalency and negations
  - I did not go to both the store and the bank
    - !(S && B)
  - I did not go to the store, or I did not go to the bank
    - !S || !B
  - Alternate Version: !(S || B) == !S && !B

Demorgan's Law

Demorgan's Law

- Negations distribute over parentheses

Demorgan's Law

- Negations distribute over parentheses
- The negation of an AND is an OR (and vice-versa)

Demorgan's Law

- Negations distribute over parentheses
- The negation of an AND is an OR (and vice-versa)
- !(x > 3 && x <= 5) → x <= 3 || x > 5

Demorgan's Law

- Negations distribute over parentheses
- The negation of an AND is an OR (and vice-versa)
  - !(x > 3 && x <= 5) → x <= 3 || x > 5
  - !(x == 3 || (x >= -2 && x != 1))

Demorgan's Law

- Negations distribute over parentheses
- The negation of an AND is an OR (and vice-versa)
  - !(x > 3 && x <= 5) → x <= 3 || x > 5
  - !(x == 3 || (x >= -2 && x != 1))
    - x != 3 && ! (x >= -2 && x != 1)

Demorgan's Law

- Negations distribute over parentheses
- The negation of an AND is an OR (and vice-versa)
  - !(x > 3 && x <= 5) → x <= 3 || x > 5
  - !(x == 3 || (x >= -2 && x != 1))
    - x != 3 && ! (x >= -2 && x != 1)
    - x != 3 && (x < -2 || x == 1)

Short Circuit Eval

- Short-circuit evaluation is the Java equivalent of this mathematical problem:
  - 0 * (5sin(π/4) + ln(2) - e^log(42))
  - You should know the answer almost immediately!

Short Circuit Eval

- int x = 3;
- if(x == 3 || (_____))

- No matter what is inside the parenthesis, when x was equal to 3, the whole expression is true
- Whatever is in the parenthesis will NOT be evaluated.

Short Circuit Eval

- int x = 3;
- if(x == 3 || (6 / (x - 3) < 2))

- If both statements were evaluated, the second would generate a division by 0 error
- Short Circuit Avoids the problem

Short Circuit Eval

- int x = 3;
- if((6 / (x - 3) < 2) || x == 3)

- This code will not short-circuit properly, as it evaluates the left side first, generating the division by 0 error before checking the second case.

ShortCircuit And

- int x = 3;
- if(x != 3 && (5 / (x - 3) < 2))

- As soon as x is 3, the first part of the expression is false and the AND statement is short-circuited.

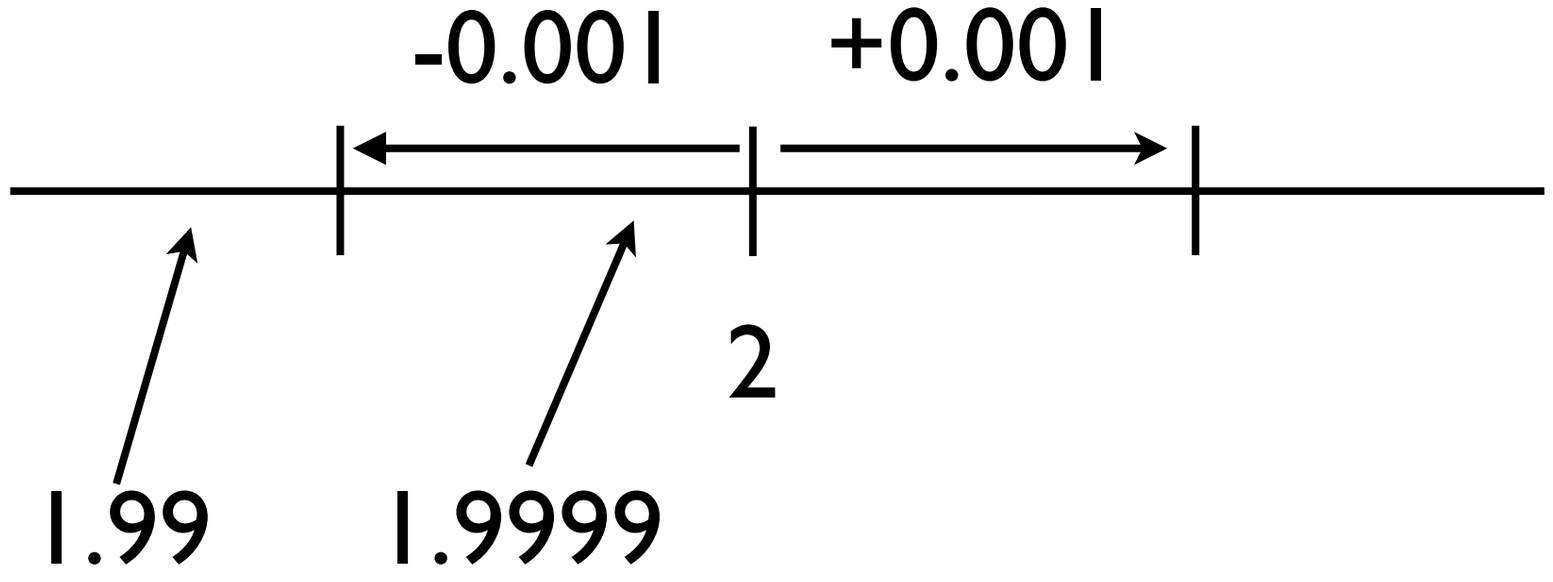
Comparing Doubles

- Decimals values are imprecise
- 0.1 + 0.1 + 0.1 = 0.30000000000000004
- if(0.1 + 0.1 + 0.1 == 0.3)...
  - This is always false!
  - Must be a better way

Interval Testing

- Want to see if a number is close enough (within a tolerance)

tolerance: 0.001



Interval Testing

- You can check for distance between two numbers using subtraction and absolute value

```
double tolerance = 0.0001;
double x = ...;

if(Math.abs(x - 2) <= tolerance)
{
  //x is close enough to 2
}
```