

*ITEC1620*  
*Object-Based Programming*

Lecture 6  
Boolean Algebra

## *Review*

---

- Branching selects from two paths
- Two paths → two states
  - `true`            (yes)
  - `false`           (no)
- Diamond contains a condition
  - A condition is a `true-false` question

## *Relational Operators*

---

- How to turn integers into `true/false`?
  - Greater than `>`
  - Less than `<`
  - Equal to `==`
  - Not equal to `!=`
  - Greater than or equal to `>=`
  - Less than or equal to `<=`

## *Relational Operators II*

---

- “missing” operators

- $a !> b$        $\rightarrow$       a not greater than b
  - $a \leq b$        $\rightarrow$       a less than or equal to b
- $a !< b$        $\rightarrow$       a not less than b
  - $a \geq b$        $\rightarrow$       a greater or equal to b

## *Relational Operators III*

---

- equivalent operators

■  $a > b$                        $\rightarrow$       a greater than b

➤  $b < a$                        $\rightarrow$       b less than a

■  $a \geq b$                        $\rightarrow$       a greater or equal to b

➤  $b \leq a$                        $\rightarrow$       b less than or equal to a

## *Compound Conditions*

---

- Allow us to put two (or more) sub-conditions into a condition
  - AND
  - OR

## AND - &&

---

- The expression is TRUE if and only if both input variables are TRUE

	TRUE 1	FALSE 0
TRUE 1	TRUE 1	FALSE 0
FALSE 0	FALSE 0	FALSE 0

## *OR - //*

---

- The expression is TRUE if either input variable is TRUE

	TRUE 1	FALSE 0
TRUE 1	TRUE 1	TRUE 1
FALSE 0	TRUE 1	FALSE 0



## *Inclusive and Exclusive OR*

---

- Computers use inclusive OR
  - Stop the bus if passengerA OR passengerB wants to get off
- Exclusive OR is different
  - You can get \$1000 cash back or 0% financing

## *Inversion (NOT) – !*

---

- The expression is TRUE if and only if the input variable is FALSE

in	out
TRUE	FALSE
1	0
FALSE	TRUE
0	1

## *Using Inversion*

---

- In Java syntax, the branch always occurs on true
  - if (true), then do the branch
  - while (true), stay in the loop

```
//boolean done;  
while (!done)
```

## *Using Compound Conditions*

---

- Some math requires compound conditions
  - $a < b < c$ 
    - $a < b \ \&\& \ b < c$
  - $a == b == c$ 
    - $a == b \ \&\& \ a == c$

## *Short-cut Evaluations*

---

if (divisor != 0 && amount/divisor < 10)

- What happens if divisor == 0?
  - amount/divisor would cause an error
  - Do not want amount/divisor evaluated

## *Short-cut Evaluations II*

---

```
if (divisor != 0)
    if (amount/divisor < 10)
```

- Bad style, and non-transferable

```
while (divisor != 0)
    ?
```

## *Short-cut Evaluations III*

---

- Fix – Java stops evaluating the terms of the condition as soon as the result of the expression is known
  - false AND something must be false
    - Stop evaluating the expression
  - true OR something must be true
    - Stop evaluating the expression

*Questions?*

---



## *Short-cut Evaluations IV*

---

boolean a = true, b = false, c = true, d = false;

`( (a || (b && c)) && ((c && (d || b)) ) || (d && c)`

- Circle the evaluated terms and what is the overall value?

- 
- a is true
    - true OR something is true, skip b && c
    - true AND something is something...
  - c is true
    - true AND something is something...
  - d is false
    - false OR something is something...

- 
- b is false

- Left side is false

- d is false

- false AND something is false

- Right side is false

$((a) \mid (b \ \&\& \ c)) \ \&\& \ (((c) \ \&\& \ (d) \mid (b))) \ \mid \ ((d) \ \&\& \ c)$

## *Short-cut Evaluations V*

---

- How does Java evaluate?
  - Left to right, until value known
- What about brackets?
  - Like math, used to determine sub-values
    - $3 * (2 + 4)?$
    - $0 * (13494 + 23847 + 34975 + 23847)?$

## *Short-cut Evaluations VI*

---

boolean a = false, b = true, c = false, d = true;

`(!b || (a && d)) || ((b || a) && d) || ((!b && c) || b)`

- Circle the evaluated terms and what is the overall value?

## *Designing if Statements*

---

- In a Java program, there are four boolean variables – itec1000, itec1010, itec1620, and itec1630. The value of each variable is true if that course is taken in first year, and false otherwise
- Write an if statement that will set the boolean variable ok to true if the chosen courses are valid (i.e. at least 9 credits and all prerequisites)

---

```
if (itec1620 && ((itec1000 && itec1010) ||  
                (itec1010 && itec1630) ||  
                (itec1000 && itec1630))
```

```
    ok = true;
```

```
else
```

```
    ok = false;
```

---

```
int count = 0;
if (itec1000)
    count++;
if (itec1010)
    count++;
if (itec1630)
    count++;
if (itec1620 && count >= 2)
    ok = true;
else
    ok = false;
```



## *Designing if Statements II*

---

- In a Java program, there are three boolean variables – movie, dinner, and clothes. The value of each variable is true if money is spent on that item, and false otherwise
- Write an if statement that will set the boolean variable inBudget to true if the chosen items cost less than \$60 total
- Costs: movie - \$30, dinner - \$40, clothes - \$50

---

## *Readings and Assignments*

---

- Text sections (5<sup>th</sup>, 6<sup>th</sup>, or 7<sup>th</sup> edition)
  - 5.1, 5.3
- Tutorial – Evaluating Conditions
- Tutorial – Designing if Statements