BOOLEAN VARIABLES AND CONDITIONAL OPERATORS

CS302 – Introduction to Programming
University of Wisconsin – Madison
Lecture 6

By Matthew Bernstein – matthewb@cs.wisc.edu

Boolean Variables

- A **Boolean variable** is a primitive data type that can store one of two possible values: true or false
- Example:

```
boolean failed = true;
```

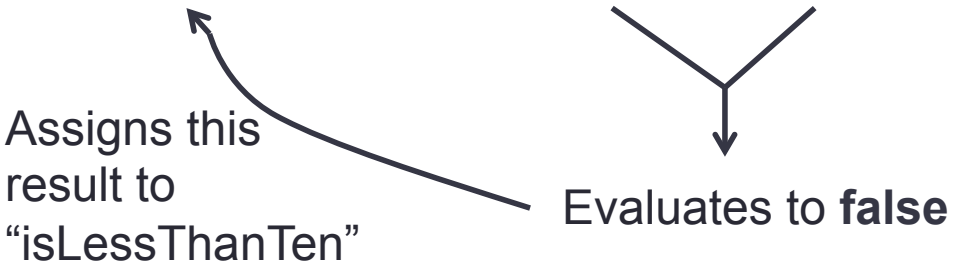
- You can then use this Boolean variable later in your program:

```
// Only executed if failed has been set to true
if (failed)
{
        …
}
```

Relational Operators return Boolean Variables

- Remember the Relational Operators (<, >, <=, >=, !=, ==). These all return a Boolean value
- Example:

```
int x = 10;
boolean isLessThanTen = x < 10;
```



Conditional Operators

- A **Conditional Operator** operates on Boolean variables and yields a new Boolean variable
- **AND (&&)** returns true if both operands are true. If either operand is false, this operator returns false
- Example:

```
x && y
```

- **OR (||)** return true if either or both operands are true. Returns false if both operands are false
- Example:

```
x || y
```

- **NOT (!)** return true if operand is false, return false if operand is true

```
!x
```

Putting It All Together

- We usually combine relational operators and conditional operators in “if” statements
- NOTE: Relational operators have a higher precedence than conditional operators
- Example:

```
if (altitude > 12 && altitude < 31)
{
        System.out.println(“Stratosphere”)
}
```

More examples…

- For what range of the variable “altitude” will this if statement’s condition evaluate to “true”?

```
if ( !(altitude > 12 && altitude < 31) )
{
        System.out.println(“Not Stratosphere”)
}
```

- Answer:

  Any value less than twelve or any value greater than 31

Common Error

- What is wrong with the following “if” statement?

```
if (0 <= temp <= 100)
{
        …
}
```

It Should Look Like…

```
if (0 <= temp && temp <= 100)
{
        …
}
```

Common Error

- What is wrong with the following “if” statement?

```
if (input == 1 || 2)
{
        …
}
```

It Should Look Like…

- The “||” operator operates on two Boolean variables:

```
if (input == 1 || input == 2)
{
        …
}
```

De Morgan's Law

- Rules for simplifying complicated logical conditions:

!(A && B) is the same as !A || !B

!(A || B) is the same as !A && !B

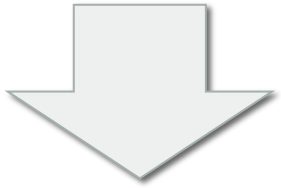
Applying De Morgan's Law

- How would we make the following "if" statement less confusing using De Morgan's Law?

!(country.equals("USA") && !state.equals("AK") && !state.equals("HI"))

Solution

!(country.equals(“USA”) && !state.equals(“AK”) && !state.equals(“HI”))



Apply De Morgan’s Law

!country.equals(“USA”) || !!state.equals(“AK”) || !!state.equals(“HI”))

Cancel out “!!”

!country.equals(“USA”) || state.equals(“AK”) || state.equals(“HI”))

Order of Operator Precedence

Operator Precedence

Operators	Precedence
postfix	*expr*++ *expr*--
unary	++*expr* --*expr* +*expr* -*expr* ~ !
multiplicative	* / %
additive	+ -
shift	<< >> >>>
relational	< > <= >= instanceof
equality	== !=
bitwise AND	&
bitwise exclusive OR	^
bitwise inclusive OR	|
logical AND	&&
logical OR	||
ternary	? :
assignment	= += -= *= /= %= &= ^= |= <<= >>= >>>=

http://docs.oracle.com/javase/tutorial/java/nutsandbolts/operators.html

Example of Operator Precedence

Demonstrations of operator precedence:

```
boolean someVariable = 3 * 2 == 5 + 1

boolean anotherVariable = 3 < 2 == 5 > 1;

if (7 % 4 < 5 && 6 > 9 - 5)
{
        System.out.println("This was executed");
}
```

Cool Link

- Barcelona Supercomputing Center: Simulating the human heart:
- http://www.youtube.com/watch?v=tKD2hfF27rM&feature=youtu.be