



CS 149

Professor: Kevin Molloy

(adapted from slides originally developed by Alvin Chao)



Relational Operators

Relational operator – compare two values, evaluate to true or false

Relational Operator	Meaning
>	is greater than
<	is less than
>=	is greater than or equal to
<=	is less than or equal to
==	is equal to
!=	is not equal to



Boolean Expressions

- The primitive data type *boolean* has two values: **true** and **false**. Boolean expressions are built using *relational operators* and *conditional operators*.
- An expression that evaluates to true or false
Examples:
 - $X < 7$
 - $A == B$
 - $3 != 4$



Relational Operators and Types

- Relational operators work as expected when comparing integers
- Also possible to compare integers with floating point values
`2 < 3.0 // true`
- Be careful with floats and doubles!
`(.1 + .1 + .1) == .3 // false!!!!`
- Chars:
`'a' < 'b' // true`
`'0' < '1' // true`
`'Z' < 'a' // true, upper-case less than lower-case`
`'9' < 'A' // true, numbers less than letters`



String Comparison

- They don't work *at all* with strings:
“HELLO” < “THERE” // **Syntax error!!!**
“HELLO” == otherString // **Won't work as expected!!!**
- To compare string you will use the method `.equals()`. If we have two strings `a = “Hello”` and `b = “hello”` the expression
- `a.equals(b)` would yield false.



If Statements

- Syntax:

```
if (boolean_expression)
    statement_or_block
else
    statement_or_block
```

- Examples:

```
if (performance > 80)
    bonusPay += 1000;
```

```
if (performance > 80)
    bonusPay += 1000;
else
    System.out.println("You are fired.");
```



Danger...

- What's wrong with this code?

```
if (performance > 80)
    System.out.println("Nice work!");
    bonusPay += 1000;
```



Prevention

- Style guide / Checkstyle says use braces and proper indentation.

We use braces here:

```
if (performance > 80) {  
    bonusPay += 1000;  
}
```

To prevent the mistake from the previous slide.

```
if (performance > 80) {  
    System.out.println("Nice work!");  
    bonusPay += 1000;  
}
```


Empty Blocks = bad style

- These are all functionally equivalent, which is better?

```
if (performance <= 80) {  
  
} else {  
    bonusPay += 1000;  
}
```

```
if (performance > 80) {  
    bonusPay += 1000;  
} else {  
  
}
```

```
if (performance > 80) {  
    bonusPay += 1000;  
}
```

Decisions Model 1

Fill in the rest of the table, the first four lines are completed.

Interactions	Value displayed	Relational operator
<code>int three = 3</code>	none	none
<code>int four = 4</code>	none	none
<code>System.out.println(four)</code>	4	none
<code>three > four</code>	false	>
<code>boolean isLarger = three > four</code>		
<code>System.out.println(isLarger)</code>		
<code>three == four</code>		
<code>three < four</code>		
<code>three <= four</code>		
<code>three = four</code>		
<code>three == four</code>		



Decisions Model 1

1. On line 5 for the first model: `boolean isLarger = three > four`
 - a) What three actions are performed in this single line of code?
 - b) Write two lines of code, ending with semicolons, that would perform these same actions (but in two lines instead of a single line).
2. List the four unique boolean expressions used in the model.
3. The `!=` operator means “not equals”. Give an example of a boolean expression that uses `!=` and evaluates to false.
4. Explain why the same boolean expression `three == four` resulted with two different boolean values in this Model.
5. What is the difference between `=` and `==` in Java?
6. Here are the six relational operators that can be used in a boolean expression. `==, >, <, >=, <=, !=`

Conditionals Model 2

Boolean expressions may also use conditional operators to implement basic logic. Relational operators are always executed first, so there is generally no need for parentheses.

Operator	Meaning
!	Not
&&	And
	Or

If all three operators appear in the same expression, Java will evaluate the ! first, then &&, and finally ||. If there are multiples of the same operator, they are evaluated from left to right.

Example Variables:

```
int a = 3;
int b = 4;
int c = 5;
boolean funny = true;
boolean weird = false;
```

Example Expressions:

```
a < b && funny
a < b && b < c
c < a || b < a
funny && a < c
!funny || weird
```



Conditionals Model 2

Example Variables:

```
int a = 3;
int b = 4;
int c = 5;
boolean funny = true;
boolean weird = false;
```

1. What do these example expressions evaluate to (true or false)?

Example Expressions:

```
a < b && funny
a < b && b < c
c < a || b < a
funny && a < c
!funny || weird
```



Conditional Operators

Example Variables:

```
int a = 3;
int b = 4;
int c = 5;
boolean funny = true;
boolean weird = false;
```

Give different examples of boolean expressions that:

- a) uses a, b, and !, and evaluates to false
- b) uses b, c, and !, and evaluates to true
- c) uses any variables, but evaluates to false
- d) uses any variables, but evaluates to true

Using your answers from the previous question, write the boolean expression `p && q` where `p` is your first answer and `q` is your second answer.

- a) Your expression:
- b) Result of `p && q`:

- **Acknowledgements**

Parts of this activity are based on materials developed by Chris Mayfield and Nathan Sprague.

</end>