Logical & Relational Expressions: expressions with *boolean* result

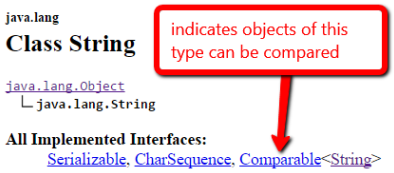
Relational operators:

Operator	Meaning	Opposite
==	equals	!=
!=	does note equal	==
>	is greater than	<=
<	is less than	>=
>=	is greater than or equal	<
<=	is less than or equal	>

- Relational operators are binary – operate on two operands (no more, no less)
- Relational operators apply only to primitive (built-in) data types (e.g. int or char): use with objects (e.g. Strings) will compile but cause logic errors

Comparing objects

- Not all objects are comparable: how would you compare one Scanner with another? Look for the word "Comparable" in the class heading (or in the API, as shown here):



- Methods for comparing Strings:

Method (from String class)	Example	Means

boolean equals(Object anObject)
// although the parameter says
// Object, we always pass a
// String to this method
	`String s,t;`
	`boolean b;`
	`s = "wobble",`
	`t = "WOBBLE";`
	`b = s.equals(t);`
		Variable b gets false, since the Strings are not equal; equals method does character-by-character comparison (case-sensitive)

boolean equalsIgnoreCase(String anotherString)	`b = s.equalsIgnoreCase(t);`
	`// using same variables`
	`// as above`
		b gets true; method does character-by-character comparison without regard to case

int compareTo(String anotherString)	`int c;` `c = s.compareTo(t);`	c gets a value greater than 0, indicating that s is greater than t
	`c = t.compareTo(s);`	c gets a value less than 0, indicating that t is less than s
	`c = s.compareTo("wobble");`	c gets 0, indicating the two Strings are equal

Comparing floating-point numbers

- Remember, doubles (and floats) are *approximations* of precise values
- For example, 0.01 and 1.0/100 may not be equal! (see DoubleTrouble)
- Need to decide how close is close enough (see DoubleLessTrouble)

Combining relational expressions

Suppose we have these declarations:

```
int x = 3, y = 2, z = 1;
boolean b = x > y > z;
```

What is the value of b? (Answer: Probably not what you think!)

Logical operators: ***correct*** way to combine relational expressions

Operator	Meaning
!	NOT: reverses truth value of expression
&&	AND: truth value determined by combination of sub-expressions; true only if all sub-expressions are true
||	OR: truth value determined by combination of sub-expressions; false only if all sub-expressions are false

Correct versions of expression above:

x > y && x > z // means: x is greater than both y & z

x > y && y > z // means the same thing, but uses transitive property

DeMorgan's Laws (where A and B are relational or logical expressions):

!(A && B) == !A || !B

!(A || B) == !A && !B