What is a Boolean expression?

**A boolean is an expression that
Evalutes to true or false.**

A boolean expression is used in a conditional.

**Boolean expression consist of
relational operators**

10 == 10	**true**
5 < 7	**true**
10 < 2	**false**
10 != 10	**false**

Relational Operators

A relational operator compares two values and determines the relationship between them

In Java		In Math
==	equality	=
>	greater than	>
<	less than	<
>=	greater than or equal to	≥
<=	less than or equals to	≤
!=	inequality	≠

Points to larger numbers

Points to smaller numbers

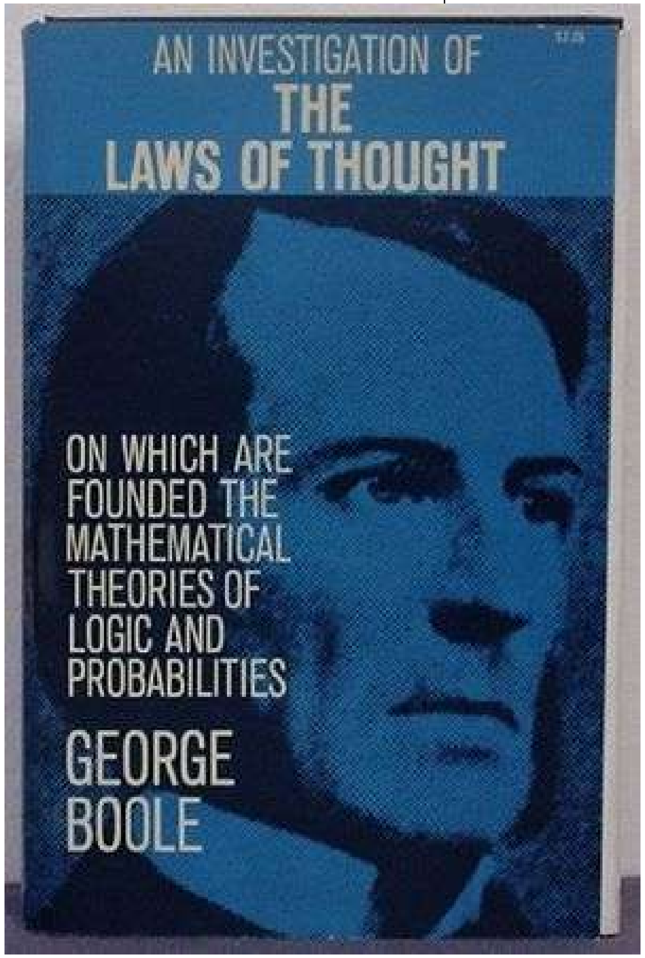
You've used relational operators **before. You just have to learn new** syntax**. Syntax is the grammar used in a language. Think of it as the rules you use in Java.**

Boolean Logic

Boolean logic is a form of mathematics in which the only values used are true and false.

Boolean logic is the basis of all modern computing.

There are three basic operations in Boolean logic – AND, OR, and NOT.


AN INVESTIGATION OF
THE
LAWS OF THOUGHT
ON WHICH ARE FOUNDED THE MATHEMATICAL THEORIES OF LOGIC AND PROBABILITIES
GEORGE BOOLE

100th Anniversary Edition

Logical Operators

- Java provides logical operators.

Operator	Meaning	Kind
&&	AND	Binary two expressions
||	OR	Binary two expressions
!	NOT	Unary one

Logic operators are used to evaluate two conditions.

```
if(x > 10 && y < 20)                    if(x > 10 || y < 20)
```

Writing boolean statements with && AND

- And operator will be true only if both expressions evaluate to true.

if(x < 10 && y > 20) both must be met

a	b	outcome
true	true	true
true	false	false
false	true	false
false	false	false

Writing boolean statements with && AND

int x = 2   int y = 90

```
if(x < 10 && y < 97)

   T              T
```

Condition would produce True

```
if(x > 10 && y < 97)

  False           True
```

Condition would produce False

Short circuit evaluation.

- (If one were false the whole thing would be false.)

Note: Java uses short-circuit (lazy) evaluation. That means in an or evaluation if the first part is true the evaluation stops and the result is true; likewise with an and evaluation with false as the first part the evaluation stops and the result is false.

Writing an or || boolean statement:

The outcome will be true as long as one of the expressions evaluates to true.

if(x < 10 || y > 20)   Only one must be true

a	b	outcome
true	true	true
true	false	true
false	true	true
false	false	false

Boolean Operators

- int x = 2 int y = 90
- **Writing an or || boolean statement:**
- (x < 10 || y < 97) Condition would produce True

  True True

- (x > 10 || y < 97) Condition would produce True

  False True

Boolean Operators Not !

- It reverses the value of a boolean expression

```
if(!(x < 10 || y >20))
```

a	outcome
True	False
False	True

Boolean Operators Not !

int x = 2   int y = 90

Writing an && with ! boolean statement:

```
    !(x < 10) && (y < 97)                Condition would produce False
         !True         True
! true = false   && True    =  False



    !(x<10 && y < 97)
       !(true  &&  true)
             !true    =  false
```

Writing Boolean Statements

Rewrite each condition below in valid Java syntax (give a boolean expression):

1. x > y > z (x>y && x > z);
2. x and y are both less than 0 (x<0 && y<0);
3. neither x nor y is less than 0 !(x<0 && y<0);

(!(x<0) && (!(y<0));

if Statements

- **Selection statements** (also known as decision statements or a conditional in programming.

- **if statements** as one kind of selection statement.

```
Basic if statement

if (number == 3)
{
   System.out.println("The value of number is 3");
   System.out.println("Goodbye");
}
```

The if statement

```
if (  boolean expression placed here  )
{
   do something 1;
   do something 2;
}
```

Several if statements

```
int x = 109;

if(x<100)
{
  System.out.println("x < 100");
}

if(x>100)
{
  System.out.println("x > 100");
}
```

OUTPUT

```
x > 100
```

if Statements

Improper structure of if

```
if(grade < 70)
 System.out.println(“You failed”);

if(grade < 80)
   System.out.println(“You passed”);
```

grade = 50;

Both if statements will execute.

When you use if statements, every if that is true will execute.

The if statement

```
int satScore = 1800;


if(satScore >= 1700)
{
   System.out.println(“College Bound!");
}

if(satScore<1700)
{
   System.out.println(“Try Again!");
}
```

OUTPUT

```
College Bound!
```

The if statement

```
int satScore = 1800;


if(satScore >= 1700)
{
   System.out.println("College Bound!");
}

if(satScore<1500)
{
   System.out.println("Try Again!");
}
```

OUTPUT

```
College Bound!
```

Conditional Statements

Programming style

Note that if there is only a single statement in the if or else block, curly brackets are not needed. If there is more than one statement in one of these blocks, the curly brackets are required.

```
if (boolean condition)
    statement;
else
    statement;
```

Curly brackets optional

```
if (boolean condition)  {
    statement;
    statement;
}
else {
    statement;
    statement;
}
```

Curly brackets required

Conditional Statements

Improper structure. Will execute every one that is true

```
public void grade(int  testScore)  {
    if (testScore >= 90)
      System.out.println("Your grade is A");
    if (testScore >= 80)
      System.out.println("Your grade is B");
    if (testScore >= 70)
      System.out.println("Your grade is C");
    else
      System.out.println("Your grade is F");
  }
```

testScore = 90;

Print:

```
Your grade is A
Your grade is B
Your grade is C
```

Boolean logic operators

```
//properly structured with boolean logic operators
  public void grade2(int  testScore)  {
   if (testScore >= 90)
     System.out.println("Your grade is A");

   if (testScore >= 80 && testScore < 90)
     System.out.println("Your grade is B");

   if (testScore >= 70 && testScore < 80)
     System.out.println("Your grade is C");

  if(testScore < 70)
     System.out.println("Your grade is F");

 }
```

```
//properly structured with if else if
  public void grade3(int  testScore)  {
   if (testScore >= 90)
     System.out.println("Your grade is A");

  else if (testScore >= 80)
     System.out.println("Your grade is B");

  else if (testScore >= 70)
     System.out.println("Your grade is C");
   else

     System.out.println("Your grade is F");

 }
```

```
//improper structure  Needs curly braces around ifs

   public void grade4(int  testScore)  {

   if (testScore >= 90)
     System.out.println("Your grade is A");
     System.out.println("First if statement");
   if (testScore >= 80 && testScore < 90)
     System.out.println("Your grade is B");
   System.out.println("Second if statement");
   if (testScore >= 70 && testScore < 80)
     System.out.println("Your grade is C");
     System.out.println("Third if statement");
   if(testScore <70)
    System.out.println("Your grade is F");
    System.out.println("Last if statement");
 }
```

Only the first statement goes with the if. Control goes to the next statement.

testScore = 70

```
First if statement
Second if statement
Your grade is C
Third if statement
Last if statement
```

Put curly braces after the if and at the end of the block that goes with the if.

```
public void grade5(int  testScore)  {
     if (testScore >= 90){
     System.out.println("Your grade is A");
     System.out.println("First if statement");}
     if (testScore >= 80 && testScore < 90){
     System.out.println("Your grade is B");
     System.out.println("Second if statement");}
     if (testScore >= 70 && testScore < 80){
     System.out.println("Your grade is C");
     System.out.println("Third if statement");}
     if(testScore < 70){
     System.out.println("Your grade is F");
     System.out.println("Last if statement"); }
  }
```

```
public void whatPrints2(int a, int b)
 {
if(a<10)
 System.out.println("Happy");
if(b>10)
 System.out.println("Boo!");
else
  System.out.println("Halloween");
 }
```

```
a = 5    b = 11        Happy      Boo
a = 5    b = 5         Happy     Halloween
a = 12   b = 11         Boo
```

Nested if statements and Control

```
public void whatPrints(int e, int f)
 {
if(e>90)
   if(f>10)
      System.out.println("go");
   else
      System.out.println("run");
else
   System.out.println("fly");
   System.out.println("nogo");
 }
```

e = 95   f = 12

go nogo

e = 95   f = 5

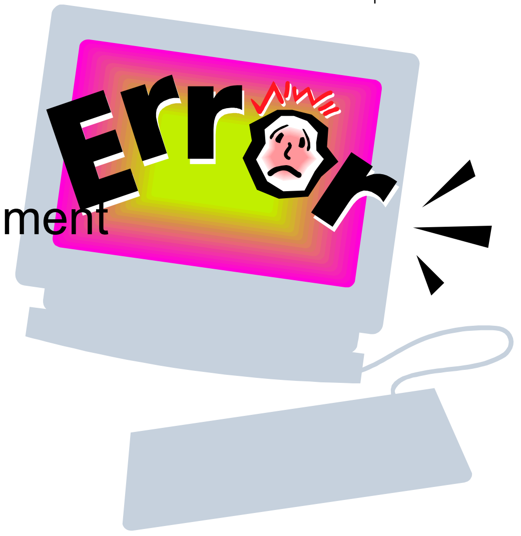
run   nogo

e = 85   f = 15

fly   nogo

common errors

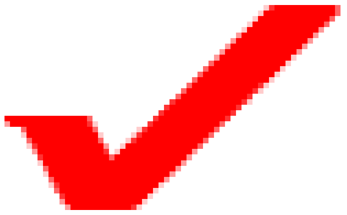
```
If(total >= 25);
{
}
```



Cannot put a semicolon after the if statement

Basic structure of an if statement

```
if(total >= 25)
{
}
```



Avoid Common Errors!

1. if should be lowercase!

   `If(num == 3)` Wrong!

2. Do not type a semicolon after the boolean expression.

   `if(num == 3);` Wrong!

3. Always use the "double equals" symbol == (i.e. comparison operator) rather than the assignment operator in control expressions.

   `if(num = 3)` Wrong!

4. Never put a ; before an open { brace

   ```
   ;{  //illegal
    };  //legal
   ```

Coding Bat theEnd

Given a string, return a string length 1 from its front if FRONT is true. if it is false return a string length 1 from the back. The string will be non-empty.

theEnd("Hello", true) → "H"
theEnd("Hello", false) → "o"
theEnd("oh", true) → "o"

Steps to solve

1. First char if front is true
2. Last char if front is false

```
public String theEnd(String str, boolean front) {
```

Coding Bat endsLy

Given a string, return true if it ends in "ly".

endsLy("oddly") → true
endsLy("y") → false
endsLy("oddy") → false

Steps:

1.if the chars at the last two index locations are ly return true.
2.Method in String called .equals(string)

```
public boolean endsLy(String str) {
```

Coding Bat twoChar

Given a string and an index, return a string length 2 starting at the given index. If the index is too big or too small to define a string length 2, use the first 2 chars. The string length will be at least 2.

twoChar("java", 0) → "ja"
twoChar("java", 2) → "va"
twoChar("java", 3) → "ja“
twoChar("Hello", -7) → "He"
twoChar("Hello", 99) → "He

What would make it return the first two chars.

- If index is too big or small for length of 2
- index < 0
- str.length()-index < 2

Return string of 2 at index
str.substring(index, index +2);

```
public String twoChar(String str, int index) {
```

hasBad

- Given a string, return true if "bad" appears starting at index 0 or 1 in the string, such as with "badxxx" or "xbadxx" but not "xxbadxx". The string may be any length, including 0. Note: use .equals() to compare 2 strings.

hasBad("badxx") → true
hasBad("xbadxx") → true
hasBad("xxbadxx") → false

Conditions for returning true
-bad is at index 0
-bad is at index 1

-str.indexOf(“bad”) == 0;
-str.indexOf(“bad) == 1;

```
public boolean hasBad(String str) {
```