# Introduction

If you have 3 people, combine Facilitator & Process Analyst.

If you have 4 people, split up into two pairs to program (Spokesperson and Process Analyst serve as drivers, Facilitator and Quality Control serve as navigators), but merge back to larger group of four to answer questions.

|  |  |
| --- | --- |
| **Team Roles** | **Team Member** |
| **Facilitator**: reads the questions aloud, keeps track of time and makes sure everyone contributes appropriately. |  |
| **Spokesperson**: talks to the instructor and other teams. Compiles and runs programs when applicable. |  |
| **Quality Control**: records all answers & questions, and provides team reflection to team & instructor. |  |
| **Process Analyst**: Considers how the team could work and learn more effectively. |  |

## Notes

**Spokesperson & Process Analyst:** You will need a computer with BlueJ open and ready to use.

**Facilitator:**

* Note the start time, end time, and total amount of time it takes to complete each model
  + Model 1:
  + Model 2:

## Model I. (15 min) Boolean Values

A boolean is a primary data type that only has two values: true or false. A Boolean expression is a comparison statement that results in one of the two Boolean values.

In BlueJ’s Main method, only some statements in BlueJ will display the ***value*** of the ***expression*** you have entered. Fill in the below table (the first four rows are completed for you).

|  |  |  |
| --- | --- | --- |
| **Type into BlueJ’s Main method** | **Value displayed (if any):** | **Boolean operator used (if any):** |
| int three = 3; | none | none |
| int four = 4; | none | none |
| System.out.println(four); | 4 | none |
| System.out.println(three > four); | false | > |
| boolean isLarger = three > four; | false | > |
| System.out.println(isLarger); | false | > |
| System.out.println(three == four); | false | == |
| System.out.println(three < four); | true | < |
| System.out.println(three <= four); | true | <= |
| System.out.println(three = four); | 4 | = |
| System.out.println(three == four); | false | == |

**Critical Thinking Questions (sorry they begin at 7 - just go with it)**

1. Examine the first line of Java code in the above model.
   1. What two actions are performed in this single line of code?

Establishes a variable and assigns a value to that variable

* 1. Give two lines of code that would perform these same two actions (but in two lines instead of a single line).

Int three = x;

Int x = 3;

1. What is the boolean variable that is declared in this Model?

False

1. List the four unique Boolean expressions in this Model.

=

<

>

==

1. Experiment with a fifth operator, the != operator. Give an example of a Boolean expression that uses the != and evaluates to false.

System.out.print(3 != 3)

1. Explain why the same Boolean expression three == four executed as two different Boolean values in this Model.
2. What is the difference between = sign and the == operator?

“=” is for assigning a value to another variable but “==” is there to act as a comparison operator between the 2 values

1. Boolean expressions are built from *relational operators* like < and >. List the six relational operators that can be used in a Boolean expression (five have been discussed so far, but you should be able to derive the sixth from the other five).

<

>

==

!=

<=

>=

## Model II. (25 min) Boolean Expressions

Many different programming constructs rely on a **Boolean expression**. Boolean expressions use Boolean logical operators (!, &&, ||) to build a condition that evaluates to either true or false. If all three logical operators appear in the same expression, Java will evaluate the ! first, then &&, and finally ||. If there are multiples of the same operators (for example, two && operators), they are evaluated left to right.

The ! (NOT) operator takes the opposite of a Boolean expression. Type the following, *one line at a time*, in BlueJ’s Main method. Remember that BlueJ will only print the value of an expression if you print it.

|  |
| --- |
| > int a = 3, b = 4, c = 5; // a multiple assignment  > a < b;  > !(a < b); |

**Critical Thinking Questions**

1. Examine the first line of Java code in the above model.
   1. Which variables are being declared in the single line (give the variable names)?

a,b ,c

* 1. Are the variables being declared to the same type?

yes

* 1. Are the variables being initialized to the same value?

No, each variable is being described as a integer

1. Give an example of a Boolean expression that:
   1. uses a and b, and evaluates to true

a < b

* 1. uses b and c, and evaluates to true

!(b > c)

* 1. uses any variables, but evaluates to false

!(b < c)

1. Using your answer to CTQ 15a and 15b, create a Boolean expressions that calculates the result of checking p && q expression, where p is your answer to 15(a) and q is your answer to 15(b). Type your expression in BlueJ’s Main method and print it out.
   1. Give the expression here:

boolean p = a<b;

boolean q = !(b > c)

* 1. Give the result of p && q (either true or false).

true

1. Now complete the following table, recording the output of all eight combinations of the Boolean expressions using the && (AND) and || (OR) operators.

|  |  |  |  |
| --- | --- | --- | --- |
| **p** | **q** | **p && q** | **p || q** |
| true | true | true | true |
| true | false | false | true |
| false | true | false | true |
| false | false | false | false |

1. Earlier you typed the expression !(a > c). Try typing the same expression without parentheses ()’s. How is ***operator precedence*** (order of operator) causing the error that you see?

“ Bad operand type int for unary operand ‘ ! ’ ”

1. Assuming a, b, and c correspond to the values in the model, explicitly give the result of each operation in the correct sequence (order of operation) and specify the final output. (Do not check your work in BlueJ yet.) In other words, show your work as you evaluate the following expression in the same order that Java would evaluate this expression:

!(a > c) && b > c

|  |  |  |  |
| --- | --- | --- | --- |
|  | Boolean  operation  ! > && | expression | result  (true or false) |
| 1st | > | a > c | false |
| 2nd | > | b > c | false |
| 3rd | && | (a > c) && b > c | true |
| 4th | ! | ! ( a > c) && b > c | false |

1. Suppose you were not sure about the order of operations for ! and &&. Add parenthesis ()’s to the Boolean expression from the previous expression in two different ways, so that:
   1. the ! is evaluated before the &&

(! ( a > c))

* 1. What does your expression in (a) evaluate to?

true

* 1. the && is evaluated before the !

! ((a < c) && b < c )

* 1. What does your expression in (c) evaluate to?

true

1. Reexamine the table from CTQ 17 that you filled out at the start of this Model for evaluating && and ||. Looking only at p and && columns, when is it necessary to examine q to determine how p && q should be evaluated?

Whenever the q is not equal to p.

1. Given that the Java operators && and || are ***short circuit logical operators*** (evaluating only what is necessary), which Boolean expression (the expression that is more likely to be true or the expression that is more likely to be false) should you place on the left of each operator?
   1. left of the && expression

True

* 1. left of the || expression

False

1. Assume you have two count variables called count1 and count2, which begin at zero and increase upwards. Write a Boolean expression that is true when one of two counts has reached 100 or higher.

boolean huned = (count1 || count2 == 100)

Consider the sentence: “I will play soccer if it’s sunny and I’ve finished my homework.” Does the “and” in English work the same as the && operator in Java? Complete a ***truth table*** with the three columns and four rows: sunny, homework finished, and play soccer and compare it with the table in the last Model.

|  |  |  |
| --- | --- | --- |
| **sunny?** | **homework?** | **Play Soccer** |
| true |  |  |
| true |  |  |
| false |  |  |
|  |  |  |