Name _____ Period _____

Boolean Expressions

Your Tasks (Mark these off as you go)

- ☐ Interpret a Boolean expression
- ☐ Apply the not operator
- ☐ Write compound Boolean statements
- ☐ Declare and initialize Boolean variables
- ☐ Apply the JAVA ternary operator
- ☐ Define key vocabulary
- ☐ Receive credit for this lab guide

☐ Interpret a Boolean expression

- A ***Boolean value*** is simply a computer science-y term that means **a true/false value**.
- A ***Boolean expression*** is a statement that *evaluates* to a *Boolean value* (a single true/false).

To determine whether two values are the same or not the same, or whether one value is greater or less than another value requires a *comparison operator*.

Below is a list of comparison operators commonly used in computer science.

`==` is equal to `!=` is not equal to `>` is greater than `<` is less than `>=` is greater than *or* equal to `<=` is less than *or* equal to	To the left are 6 common **comparison operators**. Each compares a value on the left with a value on the right and returns a Boolean value – **true** or **false**. Most of these do what you would expect. Why these symbols: **==**, **!=**, **<=**, and **>=**? 1. We use `==` because the single equal sign `=` is the assignment operator. We need something different to indicate we want to compare two values instead of assigning one to the other. 2. We use `!=`, `<=`, and `>=` because they only require ASCII symbols. Historically the mathematical symbols `≠`, `≤` and `≥` were hard or impossible to produce on some computer systems. The `!` is universally read as "not".

Refer to the following variable declarations, then indicate what is printed

```
int a = 8;
int b = 9;
int c = a;
char d = 'h'; (note, char datatypes are stored as numbers behind the scenes)
char e = 'i';
```

`System.out.print(a == b);`	
`System.out.print(a > b);`	
`System.out.print(a < b);`	
`System.out.print(d == e);`	
`System.out.print(d > e);` (HINT: does h come after i? Or the other way around)	

☐ Apply the not operator

The not (or negation) operator is used to indicate the opposite. For example, what is the opposite of true?... False of course. The "not" operator is indicated with the explanation point, !. Below are some examples.

Code	Output
`System.out.print(!true);`	`false`
`System.out.print(!false);`	`true`
`System.out.print(!(3 < 5));`	`false`
`System.out.print(!(1 == 0));`	`true`

Refer to the following variable declarations to evaluate what is printed

```
int x = 79;
int y = 46;
int z = -3;
int a = 13;
int b = 40;
```

Code	Output
`System.out.print(!!true);`	
`System.out.print(!(b < 10));`	
`System.out.print(!!( b == 40 ));`	
`System.out.print(!(x != 3));`	
`System.out.print(!!!!false);`	

☐ Write compound Boolean statements

It is also possible to write compound Boolean statements. Truth tables are useful for evaluating the outcome when Boolean statements are combined.

a	b	a AND b		a	b	a OR b
false	false	false		false	false	false
false	true	false		false	true	true
true	false	false		true	false	true
true	true	true		true	true	true

Below are examples of how to use the above truth tables to evaluate a compound Boolean statement.

Statement	True or False
`(( x < 10 ) AND ( y = 97 )`	First part is true, second part is true, entire statement is true
`(( x < 10 ) AND ( y = -3 )`	First part is true, second part is false, entire statement is false
`(( x < 10 ) AND ( y ≠-3 )`	First part is true, second part is true, entire statement is true
`(( x < 10 ) OR ( y = 97 ))`	Either part is true, entire statement is true
`(( x < 10 ) OR ( y = -3 ))`	Either part is true, entire statement is true

The above examples illustrate how true and false statements are evaluated, however the syntax is not correct. For java to correctly read the syntax the “AND” and “OR” statements must be replaced with the correct symbols as shown below,

Statement	Java syntax explanation
`(( x < 10 ) && ( y == 97 )`	*And* is replaced with &&, = is replaced with “==”
`(( x < 10 ) && ( y == -3 )`	*And* is replaced with &&, = is replaced with “==”
`(( x < 10 ) && ( y != -3 )`	*And* is replaced with &&, ≠ is replaced with “!=”
`(( x < 10 ) || ( y == 97 ))`	*OR* is replaced with ||, = is replaced with "=="
`(( x < 10 ) || ( y == -3 ))`	*OR* is replaced with ||, = is replaced with "=="

Re-write the following Boolean expressions using proper JAVA syntax, then predict the outcome

```
int w = 13;
int x = 79;
int y = 46;
int z = -3;
```

Code	JAVA syntax	Outcome
((x < 10) and (y = 46))		
((x > 10) and (y = w * 2))		
((x > 10) and (z ≠ w-16))		
((x > y/2) or (y = 5))		
((w*2 > y) or (w ≠ x))		

☐ Declare and initialize Boolean variables

Boolean variables can be declared using the boolean keyword and by setting the value of the variable to either true or false. Boolean expressions can also be assigned to boolean type variables.

Code	Output

int a = 10;
int b = 5;
boolean c = true
boolean d = false
boolean e = (a > b);
boolean f = (c != d);
boolean g = (a == 2*b);
System.out.println(e);
System.out.println(f);
System.out.println(g);
	true
	true
	true

Declare two integers, *num1* and *num2*, and initialize them to a different random number 1 – 100 (inclusive). Declare a Boolean variable, *check*, and initialize this variable to whether or not the variables you declared above are the same.

☐ Apply the JAVA ternary operator

The Java ternary operator provides an abbreviated syntax to evaluate a true or false condition, and return a value based on the Boolean result.

The syntax of the Java ternary operator is as follows,

```
(condition) ? (return if true) : (return if false);
```

Below are a few applications,

Code	Output
`System.out.println(Math.random() < .5 ? true:false);`	true of Math.random() is less than .5, otherwise false

`int a = 5;`
`int b = 10;`
`System.out.println(a == b ? "same":"different");`
	Different

`int a = (int)(Math.random() * 100);`
`int guess = userInput.nextInt();`
`System.out.println(a == guess ? “you guessed it”:“try again”`
	you guessed it or try again depending on the values of a and guess

Consider the random numbers you generated above. Prompt the user for the sum of the two numbers. Then create a scanner that could be used to get their answer. Use the ternary operator to inform the user whether or not they got the answer correct.

A palindrome is a word the reads the same whether it is written forward or backwards. We can also apply palindromes to numbers.

int num	Output
111	true
221	false
202	true
312	false
000	true

Write an algorithm that could be used to check whether a 3 digit integer is a palindrome. If the number is a palindrome, true should be printed. If the number is not, false should be printed. If statements may not be used in your solution.

☐ Define key vocabulary

Boolean

not operator

Ternary operator

☐ Receive Credit for this lab guide

Submit this portion of the lab to Pluska to receive credit for the lab guide.