

Name: _____

CS 139 – Act07A – Logical Expression/String Equivalence

Boolean expressions, String equivalence, and problem solving with if statements.

Objectives

At the end of this exercise, students will:

- Practice with evaluating Boolean expressions
- Understand how Strings are compared
- Use logical operators in decision structures
- Be prepared to do more complex coding involving decision structures

Getting ready

1. All team members - record your notes on your own response sheet.
2. Hand in each individual paper.

Part 1 – Logical operators

Logical operators take boolean operands. In other words, the logical operators are used to compare two boolean values.

In Java the logical operators are:

Logical Operator	Meaning
&&	And (and at the same time)
	Or (one or the other)
!	Not (negation or reversal)

Truth tables help us evaluate the logical operators. To read the table, look at the table for the && operator. If both operands are true, the result is true. If either operand is false the result is false. NOTE: Memorize the truth tables.

&&	true	false
true	true	false
false	false	false

	true	false
true	true	true
false	true	false

!	true	false
	false	true

Examples: !true is false
true && false is false
3 > 12 || 13 == 12 is true

1. BOARD: Evaluating logical expressions. Using the variables, types, and values in the chart below, evaluate each expression and write your result as true or false. You may use a calculator.

Variable	Data type	value
blue	char	'b'
red	char	'R'
yes	boolean	true
no	boolean	false
hiVal	int	999
loVal	int	-999
code	char	'@'
grade	double	89.5
amount	double	50.00

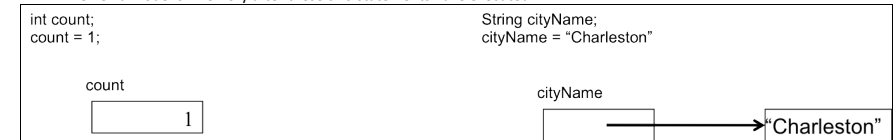
Expression	Result
a. (blue > red) && yes	
b. (blue <= red) no	
c. (yes == no) (code > blue)	
d. yes no	
e. no && true	
f. !(yes && no)	
g. (yes (no && (blue > red))) (grade <= 100)	
h. (amount <= hiVal) && (amount >= loVal)	
i. (amount <= hiVal) (amount >= loVal)	
j. ((amount + 10000) <= hiVal) (amount >= loVal)	
k. (code < red) no	

Part 2 Comparing Strings – Memory models of reference types

```
int count;
count = 3;
```

```
String cityName;
cityName = Charleston;
```

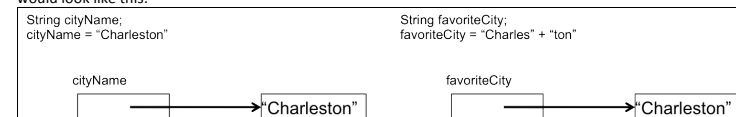
2. Given a model of memory after these two statements have executed:



how are these two data types (int and String) fundamentally different?

3. What will the model look like if we add in the declaration, `String favoriteCity;` and the assignment, `favoriteCity = "Charleston";` (Hint, there is only one "Charleston" object since "Charleston" is a literal and will be used in both cases.) Draw the change on the model above.
4. What will the model look like if we add in the declaration, `int total;` and the assignment, `total = 1;` Show your models to the instructor before moving on.
5. An int is an example of a primitive type in Java. A String is an example of a reference type in Java. Provide your own definition of primitive data type and reference data type in the space below.

6. The two Strings, "Charleston" and "Charles" + "ton" result in two different objects in Java. If we draw the memory model, it would look like this:



7. What do you think the result of `cityName == favoriteCity` will be? (Hint, what do you think is being compared?)

Part 3 Comparing Strings – Correctly comparing values of reference types

The code below illustrates the correct way to compare two objects for equality.

```
/**
 * This program correctly compares two String objects using
 * the equals method.
 */
public class StringCompare
{
    public static void main(String [] args)
    {
        String name1 = "Mark";
        String name2 = "Ma" + "rk";
        String name3 = "Mary";

        // Compare "Mark" and "Mark"
        if (name1.equals(name2))
        {
            System.out.println(name1 + " and " + name2 +
                " are the same.");
        }
        else
        {
            System.out.println(name1 + " and " + name2 +
                " are the NOT the same.");
        }

        // Compare "Mark" and "Mary"
        if (name1.equals(name3))
        {
            System.out.println(name1 + " and " + name3 +
                " are the same.");
        }
        else
        {
            System.out.println(name1 + " and " + name3 +
                " are the NOT the same.");
        }
    }
}
```

8. Reference types have a method called, `equals`, that lets us compare those types. For `Strings`, the method returns `true` if the two `Strings` contain the same characters in the same order and `false` otherwise. You see an example of method calls to `equals` in this example.
9. So, when we run this code, what would you expect to be displayed by the `System.out.println` calls?

10. If we want to compare two strings, but don't care about the case of the letters, you can also use `equalsIgnoreCase` which returns `true` if the characters are the same regardless of case. So `"abc".equalsIgnoreCase("ABC")` will return `true`.
11. READ Chapter 3.6 for more information about comparing `Strings`.
12. Do the Checkpoint 3.20 and 3.22.

3.20

3.22

13. Optional (you should read about the `compareTo` method in this same section). Do Checkpoint 3.21.

3.21

Part 4 Problem solving using logical operators

Individually, write code to solve Programming Problem #8. Then compare your answers.

14. Thought questions; be prepared to discuss in class.
 - a. Did everyone in the group solve the problem in the same way?
 - b. If not, will all correctly determine the discount?
 - c. Is any one solution better than another in this case?