Lab 05: Expressions
CSE-020-01, Fall 2019 **Points: 20**

Overview

This week we will practice writing Boolean expressions that express the desired logic. The result will be either true or false. First, let us cover the common mistakes people are doing in previous labs. You will be asked to spot these errors and fix them as part of this week's lab.

Variable Dependency

Last lab we talked about variables and their usage. Before we use a variable in any capacity, we have to declare it with a name and a data type. Any attempt to use something before their declaration or definition is illegal as the compiler does not know what you are referring to when trying to figure out the meaning of a variable using its name.

Reuse – You can read and write to a variable many times after declaration. However, there are several illegal operations and usage of variables:

- Declaring a variable more than once.
- Declaring a variable with a new data type.
- Using a variable before a declaration.
- Assigning a value that is not the same data type as the variable.

Type Casting

We have been introduced to several data types. Some of them are as follows:

- `char` – character literal (16-bits)
- `short` – 16-bits integer
- `int` – 32-bits integer
- `float` – 32-bits float point number
- `double` – 64-bits floating point number

Type casting is a way to make sure we assign the correct type to variables or how the compiler is interpreting a particular number. As we saw in previous lab, `65` can be printed as number `65` or character `'A'`. If we print `65` as a `float` then it comes out as `65.0`. Also, results of arithmetic operations depend on the data type of the inputs.

Association – type casting is applied to the nearest expression. General form is: `(type) expression`. For instance:

- `(int) 4.0 / 3.0` – `int` is applied to `4.0` only
- `(int) (4.0 / 3.0)` – `int` is applied to the result of the division

Getting Started

After starting Eclipse, create a new project called **Lab20_5**. Import `Errors4.java`, `Errors5.java`, `Errors6.java` and `BooleanOp.java` files from the Lab 05 assignment page into the project and load them. Some errors this time are logical and not compile error. So even if your program runs, there might still be errors and things that are out of order from what they should be. Utilize your knowledge from previous lab exercises and make these programs behave in a way that makes sense to the user. Some errors require fixing the executable statements, but others require changing their order in the program.

Part 1: Finding and Fixing Errors

Your job is to find the correct behavior of `Errors4.java`, `Errors5.java` and `Errors6.java`. The expected behavior of these programs is as follows:

- `Errors4.java` – It should prompt the user for two `int`s to enter then outputs their sum as an `int`
- `Errors5.java` – An interview program that asks the user two questions. First it asks for their weight in kg and second it asks for their age. The program then outputs their age in dog years and weight in lbs as `int`s without any decimal.
- `Errors6.java` – This program should ask the user for 2 `int`s, 2 `float`s and 2 `short`s. It should then calculate their average and print it as the corresponding data types of the inputs. It works very similar to `Manipulate.java` from Lab 04, but now it calculates the average.

Your changes should follow the convention below:

```
System.out.println("This is corrected statement"); // inserted double quotes at
                                                  // at the end of output
                                                  // string
// System.out.println("This is old erroneous statement);
```

You'll convert erroneous lines of code inside the files into comments (required) and then have the corrected line inserted with comments describing the fix, as shown above. You're free to add additional comments as notes to yourself to explain the nature of the error(s) you fixed. This way it'll be easy to recall the changes you made when the TA asks you demonstration, or when you subsequently review this lab.

Part 2: (Assessment) Logic Check

Create a Word document or text file named `Part2` that contains answers to the following:

1. How many types of errors did you encounter?
2. What was your testing strategy of the modified program after it runs?

Part 3: BooleanOp

Complete the `println` statements with the appropriate expression as intended by the string output. Some examples to help you get started are given in the program:

```
System.out.println(a + " is greater than " + b + ": " + (a > b));
```

Your job is to change all the statement with UNKNOWN to their proper Boolean expression as implied by the printout:

```
System.out.println(a + " is less than " + b + ": " + " UNKNOWN");
```

All the Boolean operations you will need for this exercise are:

- Greater than: `>`
- Less than: `<`
- Equal: `==`
- Not equal: `!=`
- Greater than or equal to: `>=`
- Less than or equal to: `<=`
- AND: `&&`
- OR: `||`

Operations Hierarchy or precedence is as follows:

1. `()` parentheses have highest precedence
2. `*`, `/`, `%` are next in evaluation
3. `+`, `-`
4. `<`, `<=`, `>`, `>=`
5. `==`, `!=`
6. `&&`, `||` are last to be evaluated
7. Always evaluate left to right

Part 4: Expressions Evaluation

Create a or text file named `Part4` that explains the results of the following expressions (you may want to show each step of evaluation):

- `4 + 5 * 6 + 7 / 8`
- `4 + 5 * 6 < 7 / 8`
- `4 + 5 > 6 && 7 < 8`
- `1 && 0`

What to hand in

When you are done with this lab assignment, submit all your work through CatCourses.

Before you submit, make sure you have done the following:

- Attached the file named `Part2` containing answers to the assessment questions.
- Attached the fixed up `Errors4.java`, `Errors5.java` and `Errors6.java` files.
- Attached the filled in `BooleanOp.java` file.
- Attached the file named `Part4` containing explanation and results of the various expressions.
- Filled in your collaborator's name (if any) in the "Comments…" text-box at the submission page.

Also, remember to demonstrate your code to the TA or instructor before the end of the grace period.