Worksheet 3: Boolean Logic and Selection

Updated: 13th March, 2022

The objectives of this practical are to:

- practice the use of boolean expressions;
- practice the use of **IF-ELSE** selection statements;
- practice the use of **CASE** selection statements.

1. Crafting boolean expressions

Given the following variable declarations, with the specified data types:

```
final double TOL = 0.0001;
double sarah, tony;
int luke, scott, luca;
char jane, matthew;
```

Craft Java boolean expressions for the following statements. Write your answers in a file called **BooleanExpressions.txt** in your **P03** directory. For example, if the question was "Deciding if **scott** is equal to 7", the answer would be `scott == 7`.

> **Note:** You will need to make use of real number equivalence and logical operators. Consult the lecture notes if you are unsure or are in need of a refresher. Also, be mindful of mixed mode arithmetic!

Convert the following written statements to Java boolean expressions:

(a) Deciding if **luca** is an even number and **scott** is an odd number;

(b) Deciding if **jane** or **matthew** are equal to the letter `'S'`;

(c) Deciding if **jane** is the same as **matthew**;

(d) Deciding if **sarah** is the same as **tony**;

(e) Deciding if **luke** is positive or **sarah** is between 25.0 and 125.0 (inclusive).

Check your answers with your tutor. It is easy to craft a statement that looks correct, but does not function correctly under all circumstances. You could also write a test plan to see if your statements work.

2. Evaluating Java expressions

Given the following variable declarations and initialisations below:

```
String hello = "World";
double x = 314.219, y = 240.213;
int mill = 1999, eng = 201, cap = 9;
```

Evaluate the following Java expressions, ensuring you show all of your working out. Make sure you keep in mind the order of operations – BIMDAS/PEDMAS – per the lecture slides, noting that multiplication and division are completed at the same time left-to-right as are addition and subtraction. You will also need to refer to the table showing the order of operations for other operations.

Store your answers in a file named **JavaExpressions.txt** within your **P03** directory.

(a) `(int)x - mill / 100`

(b) `hello + eng`

(c) `(double)((eng - (int)y) / 2)`

(d) `(mill / (100 * cap)) * cap - eng / 10`

3. Modifying your code: adding the reverse conversion

Last week, you created a program that converted a temperature in degrees Celsius to degrees Fahrenheit.

Your first task this week is to design a program that will allow the user to select the direction of conversion that they want (i.e. from degrees Celsius to degrees Fahrenheit, or degrees Fahrenheit to degrees Celsius). This means we will have to refactor – or modify – our code.

Firstly, make a copy of your **CelsToFaren.txt** file from last week's practical (**P02**) in to your **P03** directory. Next, copy the corresponding **.java** file over from your **P02** directory to your **P03** directory. Rename the files to **CelsAndFaren** (keeping and not changing the **.txt** and **.java** extensions, respectively).

Edit your **CelsAndFaren** pseudocode (**.txt**) to allow the user to select either degrees Celsius (by entering the '**C**' or '**c**' characters) or degrees Fahrenheit (by entering the '**F**' or '**f**' characters) for their input data, before the conversion takes place.

An example of the output that would be presented to the user is shown below:

```
Which temperature scale are you converting from?
    > (C)elsius
    > (F)ahrenheit
```

Once the user has selected the appropriate input scale, they can then enter in the temperature, which can then be converted to the other scale/unit and then output to the user. Continue to adjust your pseudocode, such that the temperature is converted from degrees Fahrenheit to degrees Celsius, or degrees Celsius to degrees Fahrenheit, depending on the input of the user.

Through re-arranging the formula from last week, Celsius can be calculated from Fahrenheit in the following manner:

$$(f - 32) * \frac{5}{9} = c$$

where f is the value in degrees Fahrenheit and c is the value in degrees Celsius.

This week, we will also ensure that the user can enter a real (decimal) value, such as **32.5**, for their input. This will not affect your pseudocode, but it will affect your Java code.

> **Note:** When writing in Java, be careful about diving 5 by 9 (or vice-versa) even when you are assigning the result to a real data type. You will need to explicitly state **5.0** and **9.0** to ensure a integer division doesn't take place and that you get the right result. Also be aware when writing your **if** conditions; you may need brackets to ensure that your intentions are interpreted correctly by the computer.

Once you have finished modifying your pseudocode, edit your Java code (**.java**) to reflect these changes. From now on, you are required to comment your code. Here are examples of the two types of comments we will be using in this unit:

```
// This is an in-line comment describing something.

/* This is a block comment that will continue to
   comment until it receives the corresponding */
```

Compile and run your code, fixing any errors you encounter while trying to compile your work. Make a copy of (and update) your test plan from last week and then use this to test your program.

4. More character conversions

Make a copy in the **P03** directory of your of pseudocode and Java code from last week for the 'character converter' (that being **CharConverter.txt** and **CharConverter.java**). Modify your pseudocode, such that the program will ask the user if they wish to convert from uppercase to lowercase, or vice versa. Then, continue to modify your pseudocode, such that the conversion will be undertaken in the correct manner depending on the users' input.

> **Note:** Ensure you *only* convert **a-z** to **A-Z** and **A-Z** to **a-z**. If the user enters a different character, the program should output an error message instead.

Edit your Java code to reflect the above changes. Then, run your program to ensure that it meets these requirements above.

> **Note:** Make sure you change *both* the pseudocode and the Java code. Firstly, adjust the pseudocode to ensure that the logic solves the new problem at hand, then change your Java code such that it reflects the changes made to your pseudocode.

Once you have completed writing **both** the pseudocode and Java code, let your tutor know so they can have a look at it. If you are stuck, take another look at last week's worksheet or ask your tutor. Once your tutor is happy with your Java code, run it and fix any errors you encounter through testing the program with a variety of inputs.

5. Creating a menu

Another common use for **IF** statements is to create menus that allow users to run parts of your program, depending on what they input. In this case, we will be putting together some basic mathematical equations with some other new code to create a program that does just that.

In your **P03** directory, create a directory named **Menu** and move into the directory. Create a new file named **Menu.txt** and design the pseudocode to create a menu. The following is an example of the style of menu expected in PDI, with the options required for this task:

```
Welcome to Programming Design and Implementation, Workshop 3

What would you like to do?
    > 1. Sum of 2 Integers
    > 2. Product of 2 Integers
    > 3. Determine if 2 Integers are Divisible
    > 0. Exit
```

Some things to consider when designing a menu:

- How will the user enter their selection?
- Will you ask them to enter a number, a character, a string or something else?
- Why are you doing this now?
- How will it affect the design of your algorithm?
- When should you ask the user to input the two integers?

When considering the above questions, keep usability in mind. Requiring the user to enter a string for the selection means the user has to type the full string **exactly** each time, which is prone to errors (or 'typos'). A character or an integer input is far more user-friendly, as there is less scope for making a mistake.

When the program is run, the user should be prompted to input their selection from the available options. Once an option is selected, the appropriate calculations should be undertaking, gathering any required input.

If the user selects 'Exit', then the program should print a 'goodbye' message and exit with no further prompts. If another option is selected, then the program should perform only that option. This may require you to split some of the code that you have written into smaller parts.

Once you have modified your pseudocode (and had it checked by your tutor), you can adjust your Java code to reflect it. Remember to indent your code and update your documentation (commenting). Compile and run your code, correcting any errors you find while attempting to do so. Ensure you test your program with a variety of inputs, including those which you expect will not generate a correct output.

> **Note:** If you are using integers for your menu input, you may assume that the user will only input integers. If you are using characters, you *must* handle any characters that are input (but do not need to explicitly consider other data types).

6. What IF I don't want a CASE?

Rewrite the pseudocode below, changing the **CASE** statement to an **IF** statement.

```
PRINT "Please enter the direction you would like to go in
        (L) left, (R) right, (U) up, (D) down "
INPUT direction (Character)
CASE direction
    'L': 'l':
        PRINT "Going west"
    'R': 'r':
        PRINT "Going east"
    'U': 'u':
        PRINT "Going up"
    'D': 'd':
        PRINT "Going down"
    DEFAULT:
        PRINT "Going nowhere"
END CASE
```

Place the pseudocode in a file named **CaseToIf.txt** in your **Menu** directory. There is no need to convert this to Java code.

End of Worksheet