

CS 101: Introduction to Computer Science

Project Four - Branching

Student Name Mohammad El-Abid

Student ID

Professor Yasser Elleithy

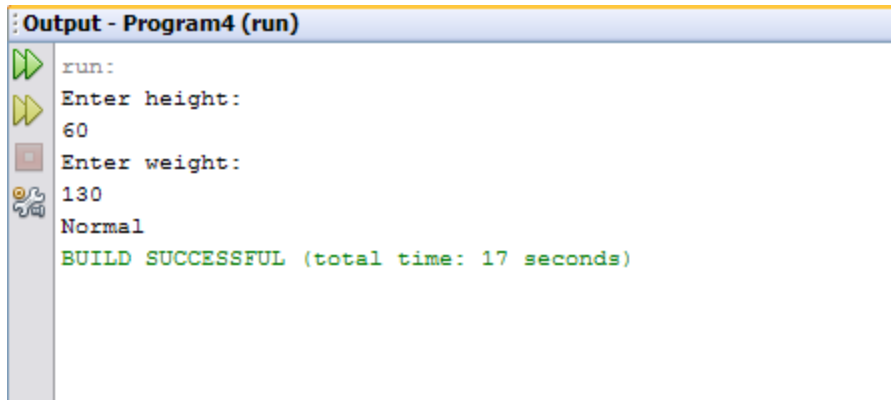
Date 10/31/11

Table of Contents

<u>Abstract</u>	page 3
<u>Introduction</u>	page 3
<u>Screenshots</u>	page 4
<u>Conclusion</u>	page 6
<u>Works Used</u>	page 6

Abstract

This application demonstrates the use of the branching control structure. I also added validations to ensure the user input was valid. The following photo was provided with the instructions for a sample use.



```
Output - Program4 (run)
run:
Enter height:
60
Enter weight:
130
Normal
BUILD SUCCESSFUL (total time: 17 seconds)
```

Introduction

This application was rather simple, take in the height in inches and weight in pounds then output their Body Mass Index (BMI) group. Considering that this was just an input and math application, I added some extra logic for handling the user input. I used a do-while loop to receive user input and ensure that it was a number and greater than zero.

Screenshots

Application output to stdout:

```
Please enter your height in inches
Sixty
Please enter a number.
-60
Please enter a number greater than zero.
60
Please enter your weight in pounds
120.0
Your BMI is 23.433333333333334, that is 'normal'.
```

Code

File: App.java

```
package edu.bridgeport.bmibranching;

import java.util.Scanner;

/**
 * The driver for a simple application that calculates Body Mass
 * Index(BMI) and lets
 * them know if they're overweight, etc.
 * @author Mohammad Typaldos [mohammad at reliablerabbit.com]
 */
public class App
{
    /**
     * Create only one instance since we use it twice.
     */
    static Scanner in = new Scanner(System.in);

    /**
     * A simple driver
     * @param args
     */
    public static void main( String[] args ){
        double height = getDouble("height in inches");
        double weight = getDouble("weight in pounds");
        double bmi = calculateBMI(height, weight);
```

```

        String group    = getBMIGroup(bmi);

        System.out.println("Your BMI is " +bmi + ", that is '" +
group + "'.");
    }

    /**
     * Uses regular expression to take only ints from the user.
     * @param varName Will be displayed to the user when asking
for input.
     * @return A parsed int from the user input.
     */
    private static double getDouble(String varName){
        double num = 0;
        boolean repeat = true;

        System.out.println("Please enter your " + varName);
        do {
            if(in.hasNextDouble()){
                num = in.nextDouble();
                if(0 > num){
                    System.out.println("Please enter a number
greater then zero.");
                    in.nextLine();
                }
            } else {
                System.out.println("Please enter a number.");
                in.nextLine(); // move the pointer to the end of
input so we don't get in an indef. loop
            }
        } while(0 >= num);

        return num;
    }

    /**
     * Calculates the Body Mass Index(BMI) of a person.
     * @param height The height of the user.
     * @param weight The weight of the user.
     * @return The BMI of the user.
     */
    private static double calculateBMI(double height, double
weight){
        if(height == 0){
            System.out.println("Invalid height");
            System.exit(3);
        }
        return weight * 703 / (height*height);
    }
}

```

```

/**
 * Returns a lower-case string of the "group" that the Body
Mass Index(BMI) is classified as.
 * @param bmi The calculated BMI
 * @return a lower-case string of the BMI "group"
 */
private static String getBMIGroup(double bmi){
    // Because of the sample input, it's safe to assume that
    // the groups are meant to be >= and not > because
    // otherwise the input would have been "overweight" and
    // not "normal" as displayed.

    // Mathematically:
    // [16, 19) - emaciated
    // (19, 25] - underweight
    // (25, 30] - overweight
    // (30, ∞) - obese

    if(16 >= bmi){
        return "emaciated";
    } else if(19 >= bmi){
        return "underweight";
    } else if(25 >= bmi){
        return "normal";
    } else if(30 >= bmi){
        return "overweight";
    } else {
        return "obese";
    }
}
}

```

Conclusion

As noted in the code, in order to achieve the same results as the sample picture, the groups must have been inclusive. I learned about some “dangers” of using `Scanner` since the pointer does not move when you use a method such as: `hasNextInt()`. I also had the chance to use a `do-while` loop, which was a new concept to me. This application was easy to code for me, but it was fun to add all of the validations on the user input.

Works Used

None