# COIT11222 Assessment 1 Part A (Programming Assignment)

# Report

|  |  |
| --- | --- |
| Student ID | S0259035 |
| Student Name | Shahriar Abrar Haque |

This coversheet **must** be completed with your submission.

Please check (X) to indicate that you have satisfied these requirements

This work and my programming submission are my original work and no part of it has been copied from any other student’s work or any other source.

No part of this work or my code has been written for me by any other person/student.

I have taken proper and reasonable care to prevent this work and my code from being copied by another student.

I acknowledge that it is my responsibility to check that the file submitted is readable and complete and that the code submitted can be uploaded in NetBeans and will compile and run correctly.

I understand that plagiarism also includes the act of assisting or allowing another person to plagiarise or to copy my own work.

# Introduction

This report on the development of the BMI Calculator, it is split into 3 Categories; Description of Phases – Includes Testing, Bugs & Limitations, Screenshot Index.

Description of Phases – Includes Testing, will show the purposes of each phase, what was added/ changed and any complications if any that occurred along the way.

Bugs & Limitations – Is split into its respective subcategories and show any bugs/ limitations that were discovered or tested for along the way.

Screenshot Index – Each phase will have its appropriate screenshots available in the screenshot index. This was chosen due to the nature of the screenshots and to enhance readability and navigation.

# Description of the Phases – Includes Testing

Note: Screenshots of each phase with their code/outputs will be in the screenshot index, in order of which phase.

## Phase 1 - Completed:

Created a scanner and prompting the user to put in values for weight/height, it then displays weight and height so the user can verify during testing that it is doing this correctly. Finally, there is a calculation at the bottom without formatting to verify that the math is being done properly too.   
“bmivalue = w/(h\*h)”

is what was used to calculate the BMI. The answer was verified by putting a print of “bmivalue” in the last line.

### What was tested:

Phase 1 was check to make sure the height and weight were inputted correctly and that the BMI was calculated correctly through printing it under it in the output (seen in screenshot index PHASE 1 – Outputs.  
Did it pass: Passed

## Phase 2 - Completed:

For developmental stages from 1-4, it was kept to a “Testing” project folder. With “Testing.java” and “Testing1.java” both representing “BMICalculator.java” and “Assessment1A.java” respectively.

The files were separated, the calculations for bmi were now in “Testing.java” and the prompts were in “Testing1.java”. “Testing” and was invoked in “Testing1” method using:

“double bmivalue = Testing.value(weight, height);”

And

“public Testing() {

}”

Was added to Testing.java

This allowed for the inputs of weight and height to get to the calculator and then

“return bmivalue;”

Was used to have it come back to the main.

### What was tested:

Using a similar test to phase 1, in phase 2 it was checked that the numbers were correctly sent to “Testing.java” , then calculated and then correctly invoked back to “Testing1.java”.  
Did it pass: Passed

## Phase 3 - Completed:

Implementing the eight-category classification, and the three-category classification was done. Both were done using if, else if & else to determine which category based on what bmivalue was. Started the if from highest to lowest bmivalue: Obese to underweight. Two variables were declared being: classif8 and classif3.These were to hold the data containing which category the bmivalue was in. This was then returned and invoked back to the main method using:

Return classif8;  
String classif8 = Testing.classification8(bmivalue);

And the same was done for the 3-category classification.

A println was added for both at the end to display the values of the two strings to verify if the data was being processed successfully.

### What was tested:

For phase 3, it was tested that if a “Control input” of 1.82m & 60kg were inputted that it would correctly display that the user was Underweight AND Low BMI from both the 8 and 3 category classification methods.

Did it pass: although testing wasn’t as thorough as could have been, it passed and selected the correct outputs for the BMI.

## Phase 4 - Completed:

During phase 4, it is worth noting that there were changes made to line 18 and line 22 in Testing1.Java between version 1 and 2. Ran into issues getting the loop to end if the user entered “0” but give 4 loops instead of 3 if the user entered 3. This was by changing current=1 in line 18 to current=0, also changing line 22 from current <= howmany to just current < howmany. From this it then was changed to Version 3, from where the “If” statement at line 22 was then shifted to line 21, so that it would before the “while(current < howmany)” loop. Testing by entering 0 to make sure that it would give the error and cancel the loop, and also testing using a number greater than but not equal to 0 to make sure it would continue without giving the error message.

### What was tested:

It was tested whether it properly looped for the correct amount of times, and outputs were still correct and also that the formatting was as specified for both layout and output.   
  
Did it pass: It passed and did what was necessary.

## Phase 5 – Completed:

Final phase, the calculations for the summary were added, with also sections to track min & max values, average, total, track the three category results too. A full example output is shown below:

### Screenshot of full test output

Graphical user interface, text, application

Description automatically generated

### Screenshot of output to match example given:

Graphical user interface, text, application

Description automatically generated

### What was tested:

Phase 5 as the final phase had the most thorough testing to make sure it was as accurate and free from flaw as can be, all categories were tested (examples to compare it too from the website were also supplied). Checking that the Lowest, Highest & Average BMI were all correct and the counters were functioning as intended in the Summary were done. This can be seen in both the above screenshots and the screenshots more extensively supplied in the “Screenshot Index” in section “Phase 5” and “Phase 5 8 Classification Testing” at the end of the document.  
  
Did it pass: It passed but overall and functions as intended. However the code still has improvements that could be made.

# Bugs & Limitations

## Bugs

Noteworthy bugs that have been recorded is that if you put in a negative height value in, it will still respond with a positive output as the math is simply height\*height, the two negatives will cross out. To counter this, a line could be added with an if/else statement following the prompt to enter the height is made, this statement could go along the lines of: “If height <= to 0, then input an error message, else – then continue to the rest of the code as usual”. If both height & weight are put in, the code will respond with “NaN which is severely underweight”, Not a Number. – Screenshots of both of these are available in the screenshot index under “Bugs”

## Limitations

Currently the known limitations that have been tested for are but not limited to, In the very unlikely situation that the lowest BMI is > 1000, the lowest will still default to 1000 since the variable was instanced with:   
“double lowest=1000;”

This could have been done with a much higher number, however, to reach a BMI of 1000, the user must have a ratio of 1000:1 of weight to height\*height.

Although specified “You are not required to cater for invalid input by the user in this assignment”, it is still noteworthy.

There is also no exit/ break command to exit the loop early/ midway during calculations.

# Screenshot Index

## PHASE 1

### Outputs:

Text

Description automatically generated

## PHASE 2

### Code:

Testing.java (to become BMICalculator.java):

Graphical user interface, text, application

Description automatically generated

Testing1.java (to become Assessment1A.java):  
  
Text

Description automatically generated

### Outputs:

Graphical user interface, text, application

Description automatically generated  
Graphical user interface, text, application

Description automatically generated  
Graphical user interface, text, application

Description automatically generated

## PHASE 3

### Code:

Testing.java (to become BMICalculator.java):  
Graphical user interface

Description automatically generatedTesting1.java (to become Assessment1A.java):  
Graphical user interface, text

Description automatically generated

### Output:

Graphical user interface, text, application

Description automatically generated

Graphical user interface, text, application

Description automatically generated  
Text

Description automatically generated

## PHASE 4

### Code:

Testing.java (to become BMICalculator.java):

Graphical user interface, text

Description automatically generated with medium confidence

Testing1.java (to become Assessment1A.java):

Version 1:

Graphical user interface, text, application

Description automatically generated

Version 2: Graphical user interface, text

Description automatically generated

Version 3:  
Graphical user interface, text, application

Description automatically generated

### Output:

Graphical user interface, text, application

Description automatically generated

Graphical user interface, text

Description automatically generated with medium confidence

## PHASE 5

### Code:

BMICalculator.java:

Graphical user interface

Description automatically generatedText

Description automatically generated with low confidence

Assessment1A.java Screenshot of lines 1-53:

Graphical user interface, text, application

Description automatically generated

Assessment1A.java Screenshot of lines 54-89:

Graphical user interface, text

Description automatically generated

### Output:

Graphical user interface, text, application

Description automatically generated

## Phase 5 8 Classification Testing:

### Expectations:

#### Very severely obese:

Graphical user interface, application

Description automatically generated

#### Severely obese:

Graphical user interface, application

Description automatically generated

#### Moderately obese:

Graphical user interface, application

Description automatically generated

#### Overweight:

Graphical user interface, application

Description automatically generated

#### Normal (healthy weight):

Graphical user interface, application

Description automatically generated

#### Underweight:

Graphical user interface, application

Description automatically generated

#### Severely Underweight: Graphical user interface, application Description automatically generated

#### Very Severely Underweight:

Graphical user interface, application

Description automatically generated

### Output:

Graphical user interface, text, application

Description automatically generated

## BUGS & LIMITATIONS

### BUGS

If negative input for height, still has a positive result:

Graphical user interface, text, application, email

Description automatically generated

If 0 is put in for both weight and height:

Graphical user interface, text, application, email

Description automatically generated

### LIMITATIONS

The lowest is 1000 in both screenshots, even if there is only one input as long as it is over 1000:

Graphical user interface, text, application, email

Description automatically generated

Graphical user interface, text, application, email

Description automatically generated