Assignment Report for

Course and Section	CSC.215
Assignment Name	Assignment 03
Due Date and Time	[Due Date] @ [Due Time]
First Name and Last Name	AJ Arce
SFSU Email Account	Aarce3@sfsu.edu
First Name and Last Name of Teammate	[First Name] [Last Name]
SFSU Email Account of Teammate	[SFSU Email Account]

PART A

Question Description and Analysis:

This part of the assignment asks that I create the English version of the BMI calculator. It asks that I write a program that has at least 5 methods. It also requires me to write a problem analysis/problem-solving in half a page. Then name the program properly and list the 5 most important methods in this program and answer some questions in detail about all of them. Then write another half page about the results and a few questions.

Answer:

1. The problem requires making a interactive Java program thats able to calculate the BMI for the user of the program based on their height (in feet and inches) and weight (in pounds). You also need to generate a table of BMI values for a range that is specified by the user. The client expects the program to be well-structured, using five methods at minimum to handle different tasks in the program, such as displaying the welcome message, gathering the inputs, performing the calculations, and presenting the results. It's important that the program is able to follow the formatting guidelines, with special

attention to spacing, number formatting, text alignment, and ensuring that the date and time are live. In order to tackle this problem, we need to divide the program into clear methods. With each method performing a specific task. One method will handle the user inputs, another will calculate the BMI of the user, and the others will handle other tasks like the results and generating the BMI table. The solution will involve loops and a lot of tweaking of the format in order to ensure the output is able to match the client's expectations. Testing each part of the program, making sure that methods are able to pass parameters properly, and output formatting will be able to fulfill the requirements of the problem. The big focus on this issue will be on accuracy and precision in matching the output the client wants, with attention to every detail of the user interface.

3. Method #1: welcomeMessage()

a. **Method Header**: public void welcomeMessage()

b. **Modifiers**: public, this makes the method accessible from any other class; void means it does not return any specific value.

c. Method Name: welcomeMessage - named this way because it displays the welcome message to the user.

d. Method Signature: public void welcomeMessage() - The method has no parameters, making the method signature simple.

e. Parameter List: No parameters for this method.

f. Formal Parameters: No formal parameters for this method.

g. Actual Parameters: No actual parameters in this method.

h. **Method Body:** The method prints a welcome message and the version of the program using System.out.println().
i. **Return Value:** No return value because it's void
j. **Return Value Type:** Since no return value, no return value type.

Explanation: This method is important because it helps identify what exactly the program is and it sets the tone of the program by welcoming the user and letting them know which version they are using. Making the program feel much more friendly.

Method #2: getUserInfo()

a. **Method Header:** public String[] getUserInfo()
b. **Modifiers:** public, making it accessible to any class; String[], meaning it returns an array of strings that contains the user info.
c. **Method Name:** getUserInfo - it's accurate because it gathers the user's name, height, and weight.
d. **Method Signature:** public String[] getUserInfo()
e. **Parameter List:** None
f. **Formal Parameters:** None
g. **Actual Parameters:** None
h. **Method Body:** The method prompts the user to enter their full name, height (in both feet and inches), and weight (in pounds), and returns this information in a String[] array.
i. **Return Value:** The method returns an array of strings that contains the users name, height in both feet and inches, and weight
j. **Return Value Type:** String[] array

Explanation: This method is important because it gathers all the necessary inputs and info from the user to make sure that the BMI calculations could be done with accuracy.

Method #3: calculcateBMI()

a. **Method Header:** public double calculateBMI(int heightFeet, int heightInches, double weight)
b. **Modifiers:** public, this makes it accessible to any class. double, makes it return a double value that represents the user's BMI.
c. **Method Name:** calculateBMI - calculates the BMI so its the perfect name.
d. **Method Signature:** public double calculateBMI(int heightFeet, int heightInches, double weight)
e. **Parameter List:** This method takes three parameters, heightFeet, heightInches, and weight.
f. **Formal Parameters:** int heightFeet, int heightInches, double weight
g. **Actual Parameters:** These parameters are provided when the method is called in the program. calculateBMI(**heightFeet, heightInches, weight).**
h. **Method Body:** This method is able to convert the height from feet and inches to total inches, and the uses the formula to calculate the BMI and it returns the calculated value.
i. **Return Value:** The method returns the BMI that was calculator as a double
j. **Return Value Type:** double.

Explanation: One of the most important methods because the core of this program to the function to calculate BMI, and this method does exactly that,

Method #4: getWeightStatus()

a. **Method Header:** public String getWeightStatus(double bmi)
b. **Modifiers:** public makes it accessible from any other class. String means that the meehod will return a string.
c. **Method Name:** getWeightStatus - pretty accurately named because it detemrs the user's weight status based on what BMI they have.
d. **Method Signature:** public String getWeightStatus(double bmi)
e. **Parameter List:** This method takes one parameter, bmi.
f. **Formal Parameters:** double bmi
g. **Actual Parameters:** The calculated BMI passed when calling this method
h. **Method Body:** Uses conditional statements in order to categorize the BMI into different weight statuses (Underweight, Healthy Weight, Overwight, Obesity)
i. **Return Value:** The method returns a String that presents the user's weight status.
j. **Return Value Type:** String

Explanation: This method is important because it helps the user understand their BMI by categorizing it into a weight status.

Method #5: displayBMITable()

a. **Method Header:** public void displayBMITable (String fullName, int heightFeet, int heightInches, double userWeight)
b. **Modifiers:** public makes it accessible to other classes. void means that it won't return a specific value.
c. **Method Name:** displayBMITable - appropriately named because it generates and displays a table of BMI values.

d. **Method Signature:** public void displayBMITable(String fullName, int heightFeet, int heightInches, double userWeight)
e. **Parameter List:** It takes four parameters, String fullName, int heightFeet, int heightInches, and double userWeight.
f. **Formal Parameters:** String fullName, int heightFeet, int heightInches, double userWeight.
g. **Actual Parameters:** Provided when calling method: displayBMITable(fullName, heightFeet, heightInches, weight)
h. **Method Body:** The method generates a table that displays BMI values and corresponding weight statuses for a range of weights, It also highlights the low & high weights as well as the users.
i. **Return Value:** No return value because it's void.
j. **Return Value Type:** void

Explanation: This method is important because it enhances the program by providing the user with a useful table of BMI information, it includes their own MI and helps them see how their weight fits into a broader range.

4. The program is able to compile and run without any errors, which should fulfill all of the client's requirements. It prompts the user for their name, height and weight, and accurately calculates and displays the BMI. The program also creates a BMI table for a specified range of weights, ensuring that hte user's BMI is clearly highlighted. Being able to organize my code in multiple methods helped me organize the code and I was able to follow and maintain it more properly. The outputs match the desired format, including the formatting of the BMI table, which

displays the weight, BMI, and weight status neatly, with proper alignment and spacing. For future development, one area that I could work on is being able to allow for more flexibility in input handling. I need to start incorporating error checks for invalid inputs. The program could also be extended to handle a wider range of BMI classifications or customized ranges for specific metrics. The user interface could be enhanced slightly with more dynamic interaction. Letting the user decide whether they want to calculate a differentBMI or just edit the weight range. Adding unit tests for the key methods would help improve the program and ensure consistent results across different use cases.

PART B

Question Description and Analysis:

This part of the assignment asks that I do the same as part A but as a Metric version. The methods are more or so the same but it wants us to explain how different it is from the english version. It wants us to list 5 methods again and answer questions about them. Then answer in another half page about the same as part A.

Answer:

1. In this version of the BMI program, the goal is to calculate the BMI using metric units such as centimeters for height and kilograms for weight. The client expects an interactive program that prompts the user for their name, height, and weight, and then calculates the BMI based on the metric formula instead of the English formula used in part A. The program also generates a table of BMI values for a specified range, displaying the BMI and weight status for each weight in that range. The big difference between this version and the English version is the units of measurement. Height is gathered in centimeters

instead of feet and inches, and weight in kilograms instead of pounds. The program needs to convert the centimeters into meters in order to calculate the BMI accurately. The program needs to generate a table with BMI values and weight statutes for a range of weights, just like the English version, ensuring that the outputs match the specified format. The design of the program involves using at least five methods, each handling different tasks such as performing the BMI calculations, generating the table displaying results, and gathering user info..Paying attention to the output format is critical to match the provided desired input.

3. Method#1: welcomeMessage()

a. Method header: public void welcomeMessage()

b. Modifiers: public, making the method accessible from any class; void, meaning it does not return anything.

c. Method name: welcomeMessage, appropriately named to indicate that it displays a welcome message.

d. Method signature: public void welcomeMessage()

e. Parameter list: None.

f. Formal parameters: None.

g. Actual parameters: None.

h. Method body: It prints out a formatted welcome message, informing the user about the BMI program.

i. Return value: No return value.

j. Return value type: Not applicable.

Explanation: This method is essential because it serves as the entry point to the program, welcoming the user and giving context about the BMI calculation they are about to perform.

Method #2: getUserInfo()

a. Method header: public String[] getUserInfo()

b. Modifiers: public, making it accessible from other classes; String[], meaning it returns an array of strings.

c. Method name: getUserInfo, named appropriately since it retrieves user information.

d. Method signature: public String[] getUserInfo()

e. Parameter list: None.

f. Formal parameters: None.

g. Actual parameters: None.

h. Method body: This method prompts the user for their full name, height in centimeters, and weight in kilograms, then returns these values as a string array.

i. Return value: Returns a String[] containing the full name, height, and weight.

j. Return value type: String[]

Explanation: This method is crucial for gathering user input, which is required for calculating BMI and generating reports. By returning the data as an array, the method makes it easy for the rest of the program to access the user's information in a structured way.

Method #3: calculateBMI()

a. Method header: public double calculateBMI(double heightInCentimeters, double weight)

b. Modifiers: public, making it accessible from other classes; double, meaning it returns a double representing the BMI value.

c. Method name: calculateBMI, clearly named because it calculates the user's BMI.

d. Method signature: public double calculateBMI(double heightInCentimeters, double weight)

e. Parameter list: It accepts two parameters: height in centimeters and weight in kilograms.

f. Formal parameters: double heightInCentimeters, double weight.

g. Actual parameters: These are provided when the method is called, such as calculateBMI(height, weight).

h. Method body: Converts height from centimeters to meters and uses the formula BMI = weight / (height²) to calculate the BMI, returning the result.

i. Return value: Returns the BMI as a double.

j. Return value type: double

Explanation: This method performs the essential task of calculating BMI. It's a straightforward calculation but critical to the functionality of the program, as the user's BMI is the primary output. Converting height from centimeters to meters ensures the calculation is done correctly, as BMI is calculated using the metric system.

Method #4: displayBMITable()

a. Method header: public void displayBMITable(String fullName, double height, double weight)

b. Modifiers: public, meaning it's accessible from other classes; void, meaning it returns no value.

c. Method name: displayBMITable, appropriately named as it generates and displays the BMI table.

d. Method signature: public void displayBMITable(String fullName, double height, double weight)

e. Parameter list: It takes the user's full name, height, and weight as parameters.

f. Formal parameters: String fullName, double height, double weight.

g. Actual parameters: These are passed when the method is called, such as displayBMITable(fullName, height, weight).

h. Method body: This method generates a table of BMI values for a range of weights and highlights the user's own weight.

i. Return value: No return value as it is a void method.

j. Return value type: Not applicable.

Explanation: This method enhances the functionality of the program by giving the user a clear view of BMI values over a range of weights. It helps the user see where their BMI stands in comparison to other weights, making the output more informative and useful. Highlighting the user's own weight and BMI adds a personalized touch to the table.

Method #5: displaySummary()

a. Method header: public void displaySummary(String fullName, double bmi, String weightStatus)

b. Modifiers: public, meaning it is accessible from other classes; void, meaning it does not return anything.

c. Method name: displaySummary, appropriately named because it displays a summary report of the user's BMI.

d. Method signature: public void displaySummary(String fullName, double bmi, String weightStatus)

e. Parameter list: It takes three parameters: the user's name, BMI, and weight status.

f. Formal parameters: String fullName, double bmi, String weightStatus.

g. Actual parameters: These are provided when the method is called, such as displaySummary(fullName, bmi, weightStatus).

h. Method body: This method prints out a formatted summary of the user's BMI, including the calculated BMI and their weight status.

i. Return value: No return value as it is a void method.

j. Return value type: Not applicable.

Explanation: This method is important for providing the user with a clear and concise summary of their BMI calculation. It organizes the user's information and results into a well-formatted report, making it easy for the user to understand their BMI and weight status at a glance.

4. The metric version is able to compile and run properly, accurately calculating the user's BMI based on the metric formula. It gathers the necessary inputs from the user, computes the BMI, and displays a summary report that includes the user's BMI and weight status. It also generates a table of BMI values for a specified weight range, with the user's BMI clearly highlighted in the table. The table displays the weights, corresponding BMI values, and their respective weight status while having the proper format and alignment. For future development, there are many

ways I could improve as a programmer. One of the ways I was looking at was to include a more interactive program, such as allowing the user to choose whether they want to calculate another BMI or explore different weight ranges without terminating and restarting the whole program. I also need to start testing the program under different scenarios with edge cases in order to make sure the program is in good quality when it comes to calculations and inputs.

Screenshots of Outputs and Explanation:

These screenshots show what I accomplished…

PART C

Question Description and Analysis:

This part of the assignment asks that I combine part A and part B to create a master program that allows the user to choose which version of the program they would like to use. It also asks to answer the same questions as part A and B as well.

Answer:

1. For this problem, I'm tasked with creating a master version of the BMI program, which allows the user to choose between two versions: the English version, which uses feet, inches, and pounds, and the Metric version, which uses centimeters and kilograms. The user can select the desired version at runtime, and the program will calculate the BMI

accordingly, display the BMI and weight status, and generate a table of BMI values for a specified range of weights. The client expects the program to handle both versions seamlessly while also maintaining the same functionality as the individual English and Metric versions. This requires conditional logic to choose the correct version and switch between the two sets of calculations and formatting based on the user's input. The program needs to follow the same structure as the previous versions, with clear outputs, proper formatting, and the ability to handle various user inputs for both units of measurement. Additionally, there's an option to exit the program using an exclamation mark. The solution to this involves combining the two BMI programs into a single program that could switch between Engish and Metric systems at runtime. By organizing the logic into distinct methods for each version and using a user manual to allow the user to choose their desired version, the program becomes very maintainable.

3. Method #1: runProgram()

a. Method header: public void runProgram()

b. Modifiers: public, making it accessible from any other class; void, meaning it does not return anything.

c. Method name: runProgram, aptly named because it is responsible for executing the main program flow.

d. Method signature: public void runProgram()

e. Parameter list: None.

f. Formal parameters: None.

g. Actual parameters: None.

h. Method body: This method presents the user with a menu to choose between the Metric and English versions of the BMI calculator, or exit the program. It captures the user's choice, calls the appropriate methods for input, calculation, and display based on the version selected, and handles invalid inputs or the exit command.

i. Return value: No return value.

j. Return value type: Not applicable.

Explanation: This method is critical because it orchestrates the entire program by managing the user's choice and directing the flow to the appropriate version of the BMI calculator. It ensures the user can choose between the Metric and English versions or exit the program.

Method #2: getUserInfoMetric()

a. Method header: public String[] getUserInfoMetric()

b. Modifiers: public, making it accessible from any other class; String[], meaning it returns an array of strings.

c. Method name: getUserInfoMetric, appropriately named as it gathers user information for the Metric version.

d. Method signature: public String[] getUserInfoMetric()

e. Parameter list: None.

f. Formal parameters: None.

g. Actual parameters: None.

h. Method body: This method prompts the user for their full name, height in centimeters, and weight in kilograms, and returns this information in a string array.

i. Return value: Returns a String[] containing the full name, height, and weight in metric units.

j. Return value type: String[]

Explanation: This method is crucial because it handles the input collection for the Metric version of the program, allowing the user to enter the necessary data for BMI calculation in the metric system.

Method#3: displayEndingMessage()

a. Method header: public void displayEndingMessage(String fullName)

b. Modifiers: public, making it accessible from other classes; void, meaning it does not return anything.

c. Method name: displayEndingMessage, aptly named as it displays a customized ending message based on the user's name.

d. Method signature: public void displayEndingMessage(String fullName)

e. Parameter list: It takes one parameter, the user's full name.

f. Formal parameters: String fullName

g. Actual parameters: The user's full name, passed when the method is called.

h. Method body: This method prints a customized thank-you message depending on the user's name, making the program more interactive and personalized.

i. Return value: No return value.

j. Return value type: Not applicable.

Explanation: This method adds a fun, interactive element to the program by personalizing the exit message based on the user's name. It enhances user engagement and makes the program more memorable.

Method #4: calculateBMIEnglish()

a. Method header: public double calculateBMIEnglish(int heightFeet, int heightInches, double weight)

b. Modifiers: public, making it accessible from other classes; double, meaning it returns a double value representing the user's BMI.

c. Method name: calculateBMIEnglish, appropriately named because it calculates the BMI using English units.

d. Method signature: public double calculateBMIEnglish(int heightFeet, int heightInches, double weight)

e. Parameter list: It accepts three parameters: height in feet, height in inches, and weight in pounds.

f. Formal parameters: int heightFeet, int heightInches, double weight.

g. Actual parameters: These are passed when the method is called, for example, when using the English version of the BMI calculator.

h. Method body: This method converts the height from feet and inches into total inches, and then uses the BMI formula for English units: (weight / (height²)) * 703. It returns the calculated BMI as a double.

i. Return value: The method returns the calculated BMI as a double.

j. Return value type: double

Explanation: This method is critical for calculating the BMI in the English version of the program. By encapsulating the BMI calculation logic specific to the English system, the program ensures that the correct formula is used when users input their height in feet and inches and weight in pounds.

Method #5: displayBMITableMetric()

a. Method header: public void displayBMITableMetric(String fullName, double height, double weight)

b. Modifiers: public, making it accessible from other classes; void, meaning it does not return a value.

c. Method name: displayBMITableMetric, clearly named because it displays a BMI table using metric units.

d. Method signature: public void displayBMITableMetric(String fullName, double height, double weight)

e. Parameter list: It accepts three parameters: the user's full name, height in centimeters, and weight in kilograms.

f. Formal parameters: String fullName, double height, double weight.

g. Actual parameters: These are passed when the method is called to generate a BMI table for a specified weight range in the Metric version.

h. Method body: This method generates and displays a table of BMI values based on a weight range input by the user. It calculates the BMI for each weight using metric units and prints the corresponding weight status for each value.

i. Return value: No return value as it is a void method.

j. Return value type: Not applicable.

Explanation: This method is essential for displaying the BMI table in the Metric version of the program. It calculates BMI values for a range of weights and formats the output in a well-organized table, ensuring the user can see their own BMI along with the BMI values for

other weights in the range. This method enhances the program's interactivity by providing a detailed breakdown of BMI calculations.

4. The master version of the BMI program successfully compiles and runs, it allows the user to choose between the Metric and English versions of the BMI calculator. The program correctly calculates the BMI based on the selected system of measurement and displays a summary report with the user's BMI and weight status. The program also generates a table of BMI values for a specified range of weights, displaying the results in the correct format. The ability to switch between versions or exit the program using a single menu ensures a smooth experience for the user. For future development, one potential improvement to the program would be to add a feature that allows the user to switch between Metric and English versions without restarting the program. Currently, once a version is chosen, the user has to terminate and rerun the program to try the other version. Letting the user easily toggle between versions or recalculate their BMI for different inputs without restarting would improve the overall user experience. Another improvement could be to allow the user to save their results in a file. Being able to review their BMI calculations and weight status over time could be helpful for them. Finally, you could improve user experience by incorporating color-coded outputs for different weight statuses, which could make the program more interactive and fun.