**HW05 Methods CS 1150 – Spring 2023**

Due Date: **See Canvas**

Purpose: Learn how to design, implement and test a quality maintainable solution following the software development life cycle using Strings, characters and Math class.

Effort: **Individual**: Read [CS Academic Integrity .pdf](https://drive.google.com/file/d/1vHXGQPPRUXZQrUxfnM6hJQid9jio7xlt/view?usp=sharing)

Points: **100 (see rubric in canvas)**

Deliverables: **Upload your design document with your reflection/learning question and your .java file as separate files. Do not upload as a zip file.**

**Assignment Description**

Write a program that allows the user to perform 3 health checks: body mass index (BMI), cholesterol, and blood pressure. The ranges and the results are presented in the table for each health check. This range information is presented to the user based on the selected health check.

| **Health Check** | **Range** | **Result** |
| --- | --- | --- |
| BMI | Less than 18.5  Between 18.5 and 25  Greater than 25 | Underweight  Normal  Overweight |
| Cholesterol | Less than 200  Between 200 and 239  Greater than 239 | Good  Borderline  High |
| Blood Pressure | Top number is less than or equal to 120 and  Bottom number is less than or equal to 80  Top number is between 121 and 129 and  Bottom number is less than or equal to 80  Top number is between 130 and 139 and  Bottom number is between 80 and 89  Anything else | Normal  Elevated  Stage 1 high blood pressure  Stage 2 high blood pressure |

**Output**

Your output may look similar to the following:

**Example #1 – Valid Input**

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Welcome to the Health Checker

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Option Health Check

------------------------

1 BMI

2 Blood Pressure

3 Cholesterol

4 Exit

**The bolded values are the**

Which health check? Select option 1, 2, 3, or 4: **1 values entered during run**

Enter weight in pounds: **160**

Enter height in feet: **5**

Enter height in inches: **10**

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

BMI Result = Normal

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Option Health Check

------------------------

1 BMI

2 Blood Pressure

3 Cholesterol

4 Exit Running program again for

a different health check

Which health check? Select option 1, 2, 3, or 4: **3**

Enter cholesterol: **230**

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Cholesterol Result = Borderline

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Option Health Check

------------------------

1 BMI

2 Blood Pressure

3 Cholesterol

4 Exit User ending the program



Which health check? Select option 1, 2, 3, or 4: **4**

Have a healthy day! Goodbye

**Example #2 – User performs no health checks**

Your code must allow the user to quit without performing any health checks*.*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Welcome to the Health Checker

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Option Health Check

------------------------

1 BMI

2 Blood Pressure

3 Cholesterol

4 Exit

Which health check? Select option 1, 2, 3, or 4: **4**

Have a heathy day! Goodbye

**Example #3 – Handle invalid input**

Your code **must check** for valid input when *prompting user for which health check.*

When input is invalid, keep prompting until a valid shape is entered.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Welcome to the Health Checker

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Option Health Check

------------------------

1 BMI

2 Blood Pressure

3 Cholesterol

4 Exit **When invalid menu option, tell user**

**to reenter until valid option entered**



Which health check? Select option 1, 2, 3, or 4: **0**

0 is not a valid entry - try again

Which health check? Select option 1, 2, 3, or 4: **5**

5 is not a valid entry - try again

Which health check? Select option 1, 2, 3, or 4: **2**

Enter top blood pressure number (systolic): **120**

Enter bottom blood pressure number (diastolic): **90**

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Blood Pressure Result = Stage 2 high blood pressure (hypertension)

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Option Health Check

------------------------

1 BMI

2 Blood Pressure

3 Cholesterol

4 Exit

Which health check? Select option 1, 2, 3, or 4: **4**

Have a heathy day! Goodbye

## Requirements Analysis and Design

Read slides 1-11 on  [Problem Solving and the Software Development Life Cycle](https://docs.google.com/presentation/d/e/2PACX-1vSC4tM0BsOaYjRmd_emNeHtKrWaExKXfvvLxQC10rCt77CPT_WvF_s3W--o_yuj8XczYGo7qHxgjMIq/pub?start=false&loop=false&delayms=3000&slide=id.p1) and review the last homework examples of breaking into tasks, creating test cases and writing pseudocode.

Think about where you will use if/else or switch statements. Do not write a code solution until you complete this first. This is to get ideas on paper and practice problem solving. Create a document and include the following in your design.

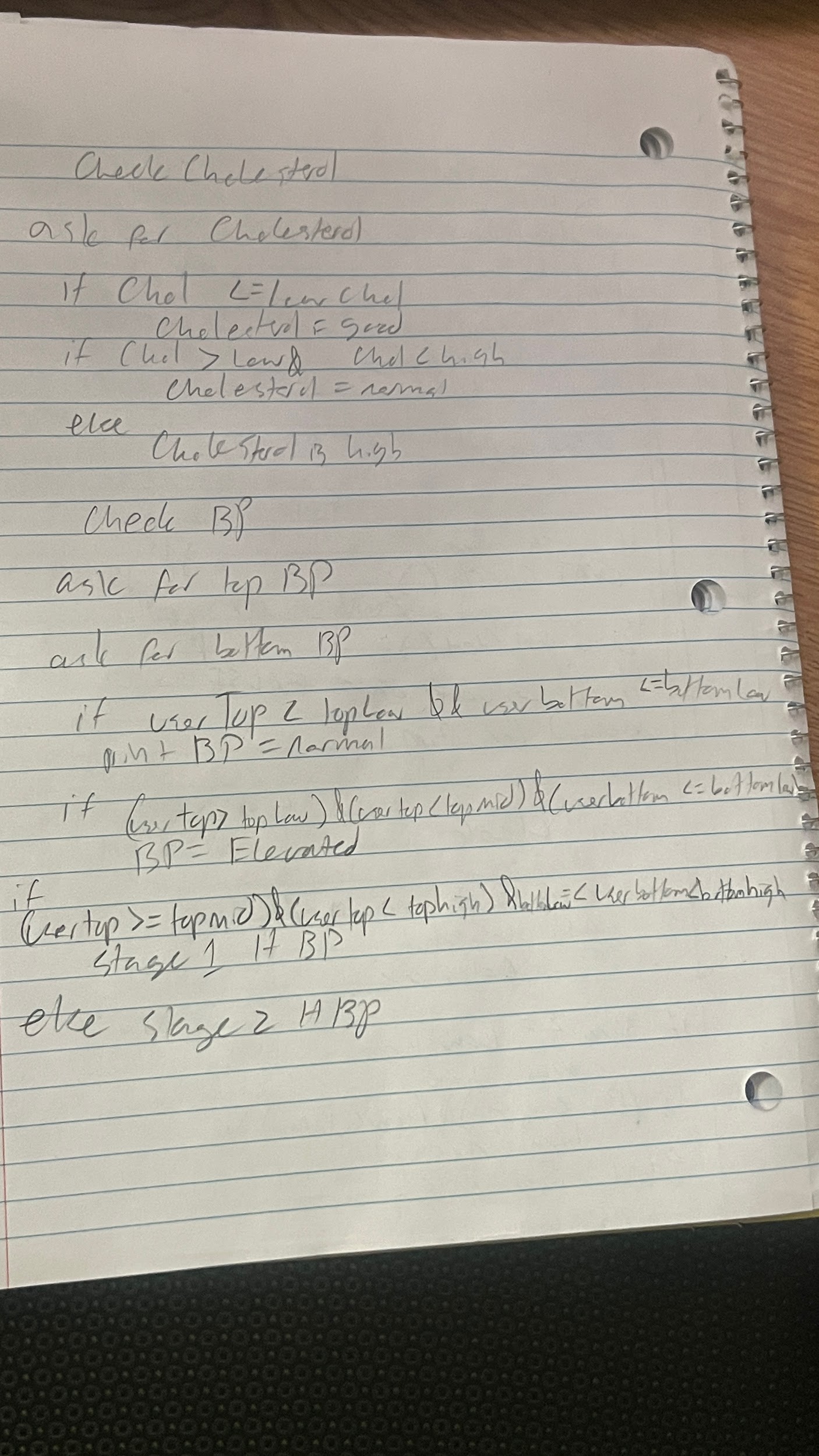
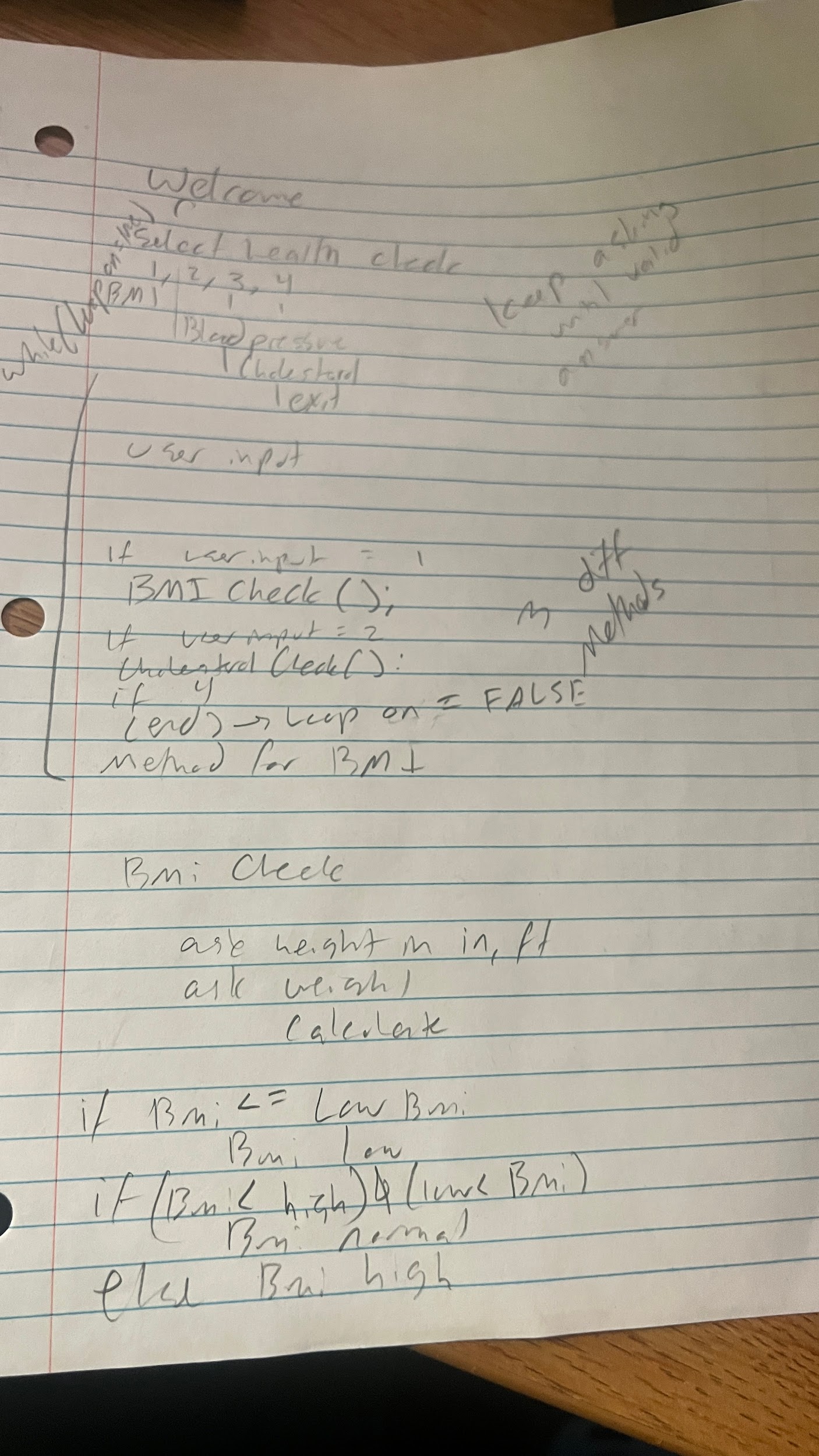
**Requirements Analysis - Tasks**

* Break the problem into smaller pieces by listing all tasks needed to implement a solution.
* Put in the design document or on paper.

**Requirements Analysis - Test Cases**

* A table for each task needed to solve the problem with a wide range of possible inputs and outputs to help ensure functional correctness.
* This can be on paper or in the design document.

**Resources**

* List at least 4 resources that are or will be useful for the assignment. This should be in the document so it can include links.
* Resource Location -can be any of the following -
  + Lecture Number and slide(s) -
  + Section from the book -
  + JavaTPoint : <https://www.javatpoint.com/java-tutorial>

**Design Possible Solution**

* Hand written - paper, whiteboard or tablet. This might include a sketch of ideas to understand the problem.
* For each task include a pseudocode of steps (algorithm). Put images from your writing in the final design document.

## Specification for Implementing Code

[Review Problem Solving Using Software Development Life Cycle](https://docs.google.com/presentation/d/e/2PACX-1vSC4tM0BsOaYjRmd_emNeHtKrWaExKXfvvLxQC10rCt77CPT_WvF_s3W--o_yuj8XczYGo7qHxgjMIq/pub?start=false&loop=false&delayms=3000&slide=id.ge6997ea817_0_125) slides 12 - 19

Method reminders

* Use meaningful names: start with lowercase and use camel case
* Comments above methods with description, inputs and returns
* **Be sure to properly layout your code when including methods**

**public** **class** ProperCodeLayoutWithMethods {

**public** **static** **void** main(String[] args) {

// Code inside main to prompt user for which health check

**int** healthCheck = *getHealthCheck*(userInput);

} // main

// Methods go after the closing curly brace for main

**public** **static** **int** getHealthCheck (Scanner userInput) {

// Code to ask user which health check to perform

} // getHealthCheck

} // ProperCodeLayoutWithMethods End of main class is after all methods

1. Create a Java class **LastNameFirstNameHW05** within that project calledwithin the **CS1150HW** project
2. In Main create a loop to perform health checks as many times as the user desires. Think about the condition needed to exit the loop.

* Call ***getHealthCheck*** method to get valid health check choice
* The code must allow the user to quit the program where no health checks are performed, that is, if the user enters 4 on the first menu.
* Call correct health method based on value returned by ***getHealthCheck***
* Display results for the selected health check by calling the method ***displayResults*** inside.

1. Design your program to use the following methods with the **given return type and parameters**

* You can add more methods, but you must write all the methods as shown below
* Do not change the return types or parameters of these methods.
* To help develop a strong understanding of methods, you must store the value returned by any value-returning method in a variable.
  + This means you are not allowed to use a method call as an argument of another method call. For example, you cannot use the checkBMI method call as an argument of the displayResults call

*displayResults*(BMI, *checkBMI*(bmi));

* + Instead, you must store the value returned from a method in a variable.
    1. String bmiResult = *checkBMI*(bmi);
    2. *displayResults*(BMI, bmiResult);
* Remember SRP: the health check methods should perform only 1 task, that is, a specific health check and return a String.

// Prompt user for which health check to perform and validates user input.

// Returns a valid menu option.

**public** **static** **int** getHealthCheck (Scanner userInput)

* This method displays the health check menu and prompts user to select an option

Option Health Check

------------------------

1 BMI

2 Blood Pressure

3 Cholesterol

4 Exit

Which health check? Select option 1, 2, 3, or 4:

* The method reads health check option and ensures user entry is a valid option (a number between 1 & 4)
* When an invalid health check option is entered, repeat the *select health check to perform* prompt until a valid option is entered.
* Method returns a valid health check option - a number between 1 and 4.

// Prompt user for values needed to compute body mass index (BMI) & computes BMI.

// Returns the computed BMI value.

**public** **static** **double** computeBMI(Scanner userInput)

* **When computing BMI enter user’s height in feet and inches**
  + The formula for calculating BMI is:

BMI = 703 X

* + Allow user to enterheight in feet and inches for BMI and perform conversion to inches inside the ***computeBMI*** method.

Enter weight in pounds: **160**

Enter height in feet: **5**

Enter height in inches: **10**

// Checks the range for the body mass index and returns the result as a string:

// underweight, normal, or overweight.

**public** **static** String checkBMI(**double** bodyMassIndex)

// Checks the range for the cholesterol and returns the result as a string:

// good, borderline, high

**public** **static** String checkCholesterol(**int** cholesterol )

// Checks the range for the blood pressure and returns the result as a string:

// normal blood pressure, elevated blood pressure, stage 1 high blood pressure,

// stage 2 high blood pressure

**public** **static** String checkBloodPressure(**int** topNumber, **int** bottomNumber )

// Displays the results for a specific health check.

// Requires using a switch statement.

**public** **static** **void** displayResults(**int** healthCheck, String healthCheckResult )

* The ***displayResults*** method is the only method where results will be displayed.
* A **switch statement** must be used in the method **displayResults**

## Learnings and Reflection

* Explanations provide elaboration of the concepts using supporting evidence and vocabulary.
* Thoroughly reflects making clear the connection(s) with the experience and assignment.

| 1. Explain what it means to write modular code that follows SRP and DRY.  DRY Means dont repeat yourself, and if you find yourself repeating code you should find a way to write it where you dont have to repeat yourself. SRP means Single Responsibility Principle, Where a programmer’s goal should be to have a single responsibility for each function or method. |
| --- |
| 2. Explain method concepts using examples from your code. Include the following in your explanation: pass by value, copy, parameters, local variables, scope and return void and return a primitive data type value. |
| 3. How does the design help you implement modular code that is maintainable and testable? Include screenshots of an example from your requirements analysis, design and code |
| 4. What does it mean to overload a method? Give two examples from the java Math class.  What must be true for a method to be considered overloaded? |
| 5. How do you feel about creating modular solutions for a problem?  It makes your code easier to write, and easier to follow and segments it problem by problem.  What are you proud of from your design and implementation?  I am proud of my code being easy to understand  What are at least two strengths of yours as you follow the [Software Development Life Cycle](https://docs.google.com/presentation/d/e/2PACX-1vSC4tM0BsOaYjRmd_emNeHtKrWaExKXfvvLxQC10rCt77CPT_WvF_s3W--o_yuj8XczYGo7qHxgjMIq/pub?start=false&loop=false&delayms=3000&slide=id.p1)?  Two strengths I have are problem solving and Testing code, and something I can improve on is following design requirements better.  What is one area you could improve? |