Assignment 1 Instructions

1. Assignment 01: **50 points total with 2 E.C. points (For class participation, for extra work helping others in class, for not being late on submitting your assignment.)**
2. Due Date & Time: **09-07-2020 at 11:59 PM**

WHAT TO SUBMIT

Submit 4 files to iLearn by the deadline 09/07 11:59PM PST. [48pts + 2 E.C. pts = 50 points]

- 3 Files: Please submit 3 files to iLearn: BMI.java, KnockKnock.java, <YourOwnIdea>.java. **[30 points]**
- 1 File: Submit 1 Word/PDF file which is a filled-out, downloaded local copy of this Google page on your local computer, named "firstname-lastname-assignment-1-report.pdf". Fill this out with screenshots and your reflection, then save it as Word or PDF **[18 points]**

HOW TO SUBMIT

Please upload all 4 files separately via iLearn Assignments Submission

---

GUIDELINES FOR ALL ASSIGNMENTS:

1. Each assignment includes a code portion and a non-code portion. Please submit both 2 portions.
    a. Code portion: Your source code files, only the files which you create and edit.
    b. Non-code portion: Your assignment report, only 1 **Word** or **PDF** file.

2. Please submit all required files separately, un-zipped, via iLearn Assignments Submission

3. Always **read through the entire assignment before starting and submitting any of it. Missing files or missing requirements will result in deducted points**

Assignment 1
Simple Java Programs

ABOUT THIS PROJECT

You will be creating simple Java programs. By the end of this assignment, you should understand how to create simple Java programs

❒Part 1: BODY MASS INDEX (BMI) Calculator[9 points]

INSTRUCTIONS

Body mass index (BMI) is an estimate of whether a person is overweight. (See for example http://www.halls.md/body-mass-index/bmi.htm .) BMI is defined as 703 * (weight in pounds) / (height in inches)^2. A BMI of 25 is considered borderline overweight. Write a Java program that performs these actions:

- Prompt the user to enter height in feet and inches (both integers)
- Read the height in two separate prompts.
  For example if someone is 6 feet 7 inches, then
  1. Ask the user to enter their feet part of their height as an **integer**, first, so : 6
  2. Then, ask the user to enter their inches part of their height as an **integer**, second, so for example: 7
- Prompt the user to enter their weight in pounds (a **double**)
- Calculate the BMI of the user (a **double**)
- Display the BMI

NOTE : Please do not use “Integer” class, use “int”

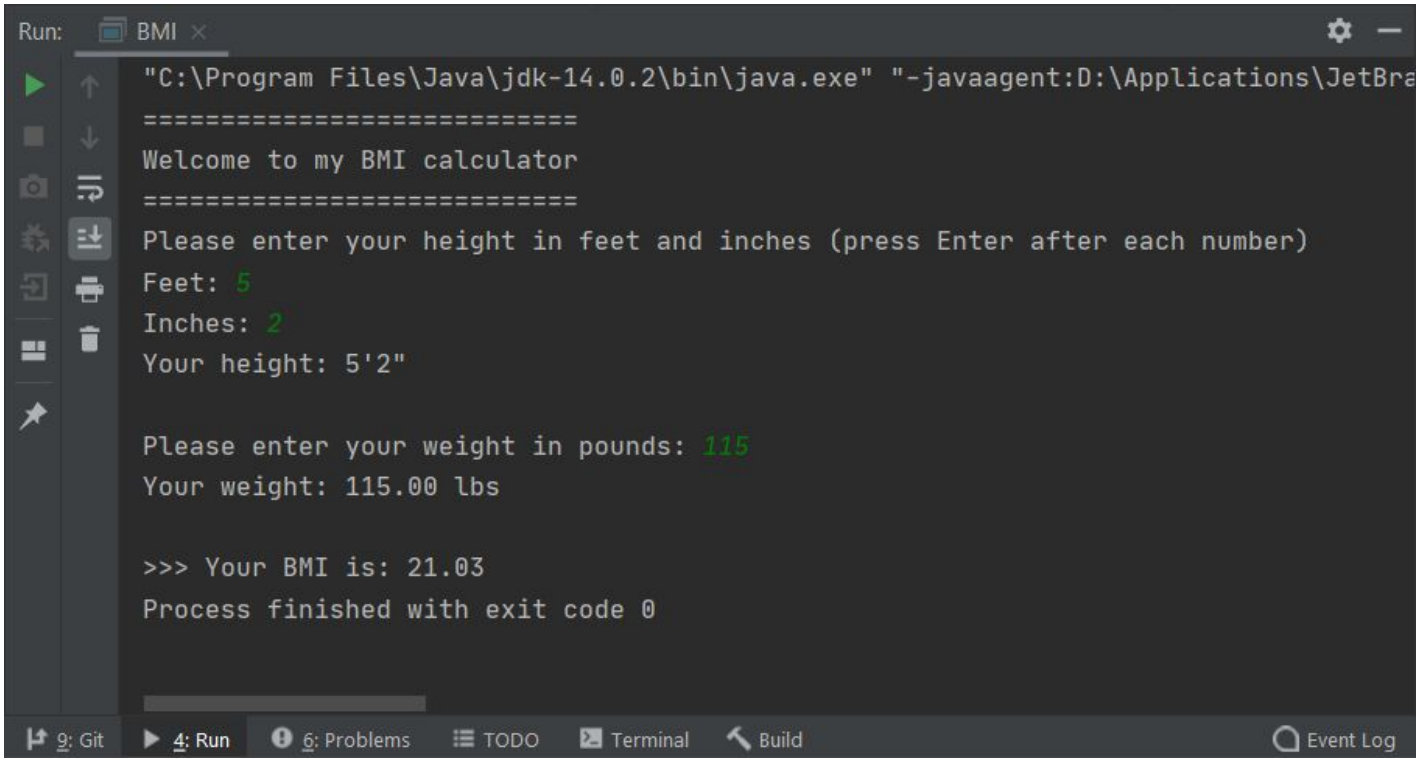
FILE NAME REQUIREMENTS

BMI.Java

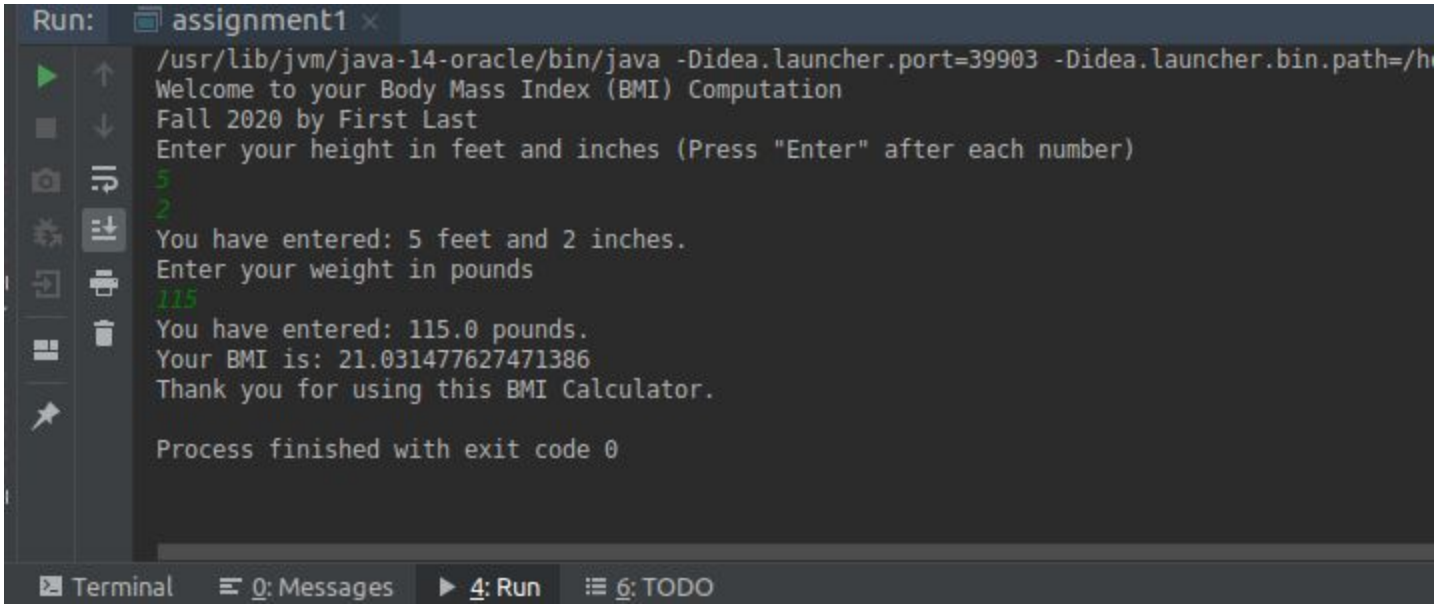
SUBMISSION INSTRUCTIONS

1. Submit the 1 BMI.Java file directly on iLearn
2. Take a screenshot of the output of your program and paste it here

Note: BMI and weight outputs are rounded to 2 decimal points (.2f)



Example



TIPS FOR PART 2 AND PART 3:

1. Save your user's input in a variable in order to use it in the next printout

When you use the Scanner object,
don't forget to save it in a variable like
this

To save an integer/a number:

int **myInt** = myScanObj.nextInt(); //save user's number input into an integer primitive variable named myInt

To save a string:

or String **myString** = myScanObj.nextLine();/save user's string input into a String reference variable named myString. And yes, that is a "String" data type, with a capital "S"

Why do you want to do this?

So that you can print out what the user had entered.

For example:

System.out.println("You entered this" + **myInt**); or System.out.println("You entered this" + **myString**);

2. If you **will not** be using the user's input for an output later on, then you don't need to scan it at all.

"scan" means this myScanObj.nextInt(); or myScanObj.nextLine();

3. The KnockKnock joke is a very simplistic program

It is a very simplistic program, so if the user answers anything at all, it will still proceed as if the user entered a correct answer.

Meaning you could expect a conversation like this:

Knock Knock? ←-your program's output

XXXBLBBLABs;isdfa98f0s <---- user input, no need to save or scan this

Annie <-- ignores user input, and goes on with your program output joke

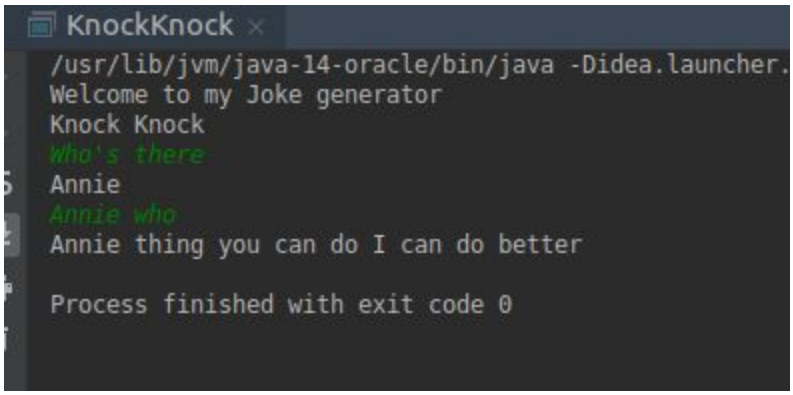
XXX...!!ososd9 ← user's input, no need to save or scan this

XXX...!!ososd9 thing you can do I can do better <--- you take this last user's input and add to your response output " you can do I can do better"

❒Part 2: Knock Knock Joke [8 points]

Knock Knock Joke: In the future assignment, you can make a more fun knock knock joke program. For now, it would work for only 1 joke and 1 exact response you may like the best. 😊

Ideally, users are expected to enter the proper response as shown in the green font. And your program responses with 1 joke.



SUCCESSFUL RUN REQUIREMENTS

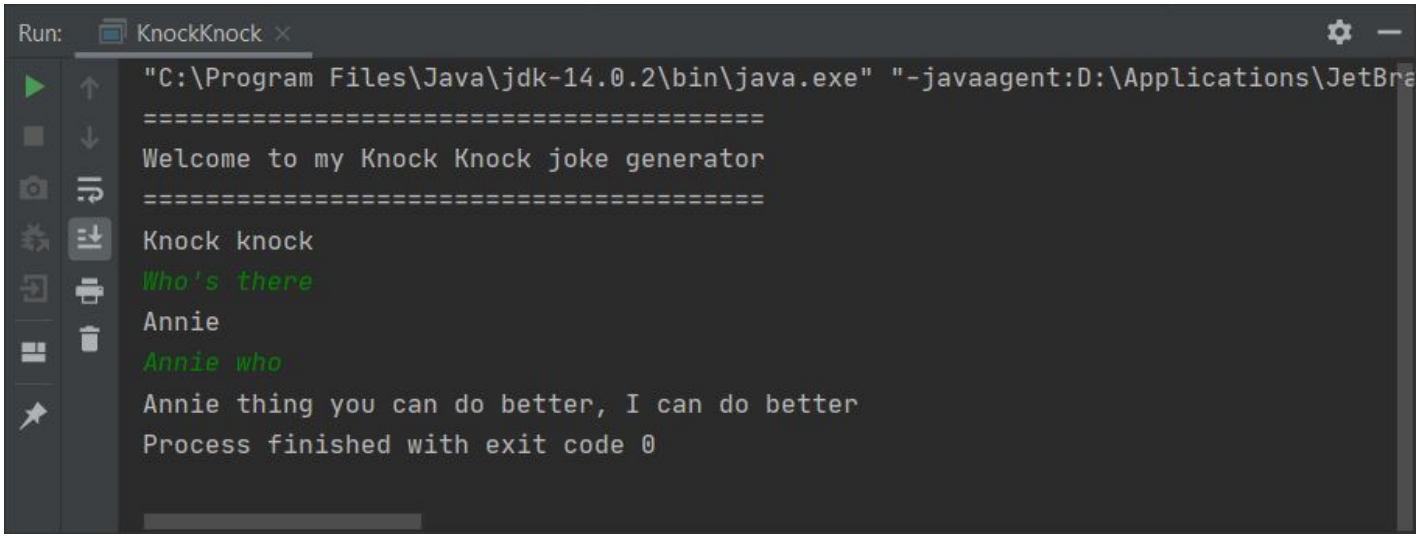
1. Running the program should output a statement like the above

FILE NAME REQUIREMENTS

KnockKnock.Java

SUBMISSION INSTRUCTIONS

1. Submit the 1 KnockKnock.Java file directly on iLearn
2. Take a screenshot of your Knock Knock Joke and paste it here. It can be any Knock Knock joke that takes in a user input and outputs a response


Run:
KnockKnock
"C:\Program Files\Java\jdk-14.0.2\bin\java.exe" "-javaagent:D:\Applications\JetBra
==========
Welcome to my Knock Knock joke generator
==========
Knock knock
Who's there
Annie
Annie who
Annie thing you can do better, I can do better
Process finished with exit code 0

❒Part 3: Your own idea [11 points]

It can be similar to part 1 or 2. Please use what you learned from problem 1 and 2 and write a simple code to make it work.

a. It must provide a welcome printout “Welcome to my program called: ... “ (2 points)

b. Must include prompting reading user input ex. “Please enter this value… “ (2 points)

c. And, **must use this input for the next step printout.** It can be completely funny or serious, or maybe a project you have been thinking on. (2 points)

d. Must provide an explanation of your idea, and what it is trying to do as I showed in 1 and 2. ex. “This program is to calculate blablabla…” (2 points)

FILE NAME REQUIREMENTS

<YourFileName>.Java

SUBMISSION INSTRUCTIONS

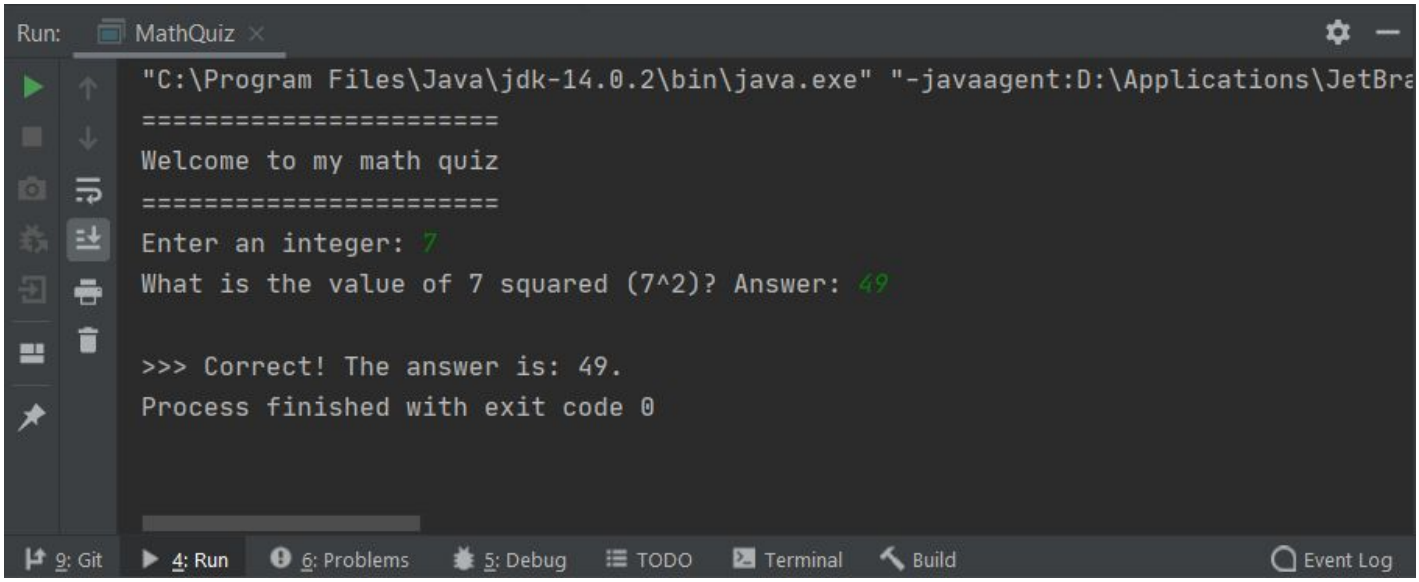
1. Submit the 1 <YourFileName>.Java file directly on iLearn
2. Take a screenshot of the output of your program and paste it here

Explanation:
This program is a math mini-game. The pseudocode is as follows:

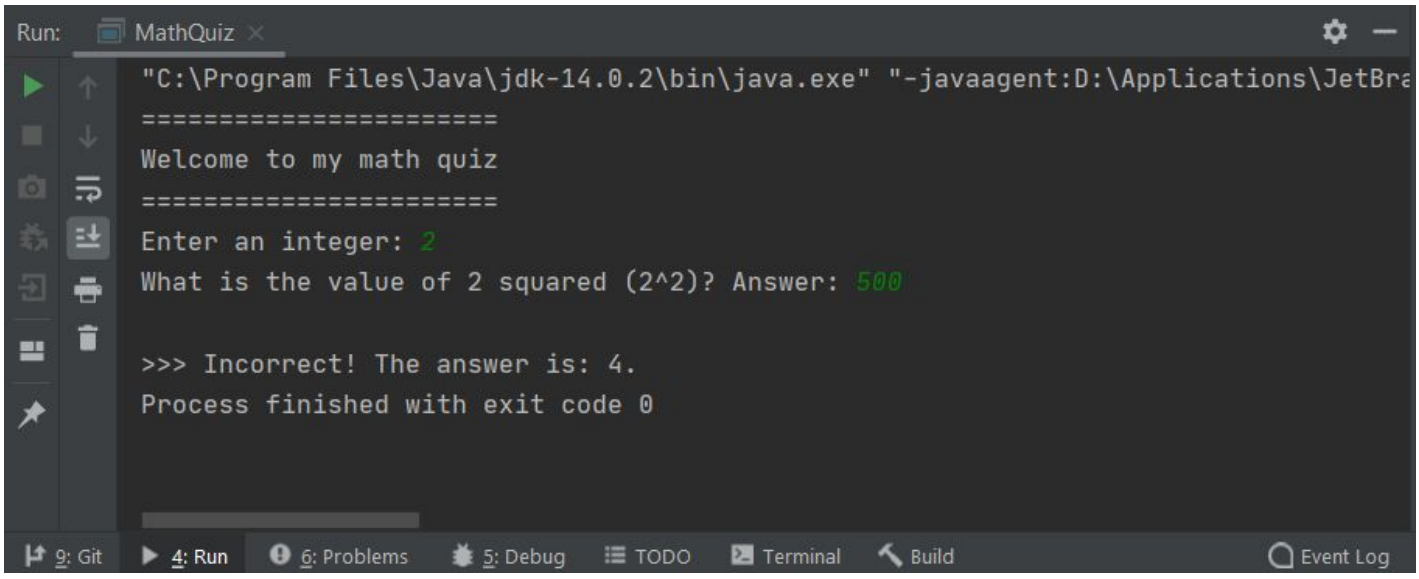
- ask user for an integer (e.g., 5),
- ask user for the squared value of the integer provided in part 1 (e.g., 25),
- evaluate the answer,
- print a message to tell the user if that was correct.

Note: the program does not perform any checks on the input and will crash if an integer is not provided in either parts.

Example 1:
If the user provides a correct answer to their question



Example 2:
If the user provides an incorrect answer to their question



❒Part 4: Comment your code [2 points]

Every Java file you write in this assignment will require you to include descriptive comments.

In this assignment, you are tasked with writing a descriptive

1. Headers

2. Comments

You can write comments in two ways:

- Single-line comments using the // notation.
- Multi-line comments using the /* and */ notation.

a. Include a proper header at the top of every Java file. Figure 1

Header Format
/* * Assignment **<assignment number>** * Description: **<program description>** * Name: **<your name>** * ID: **<your SFSU ID number>** * Class: CSC 210-**<section number>** * Semester: **<current semester>** */

Replace each tag (such as **<assignment number>**) with the appropriate text.
You should adhere to this format as closely as possible. You do not need to include the **<>** symbols in your header fields.

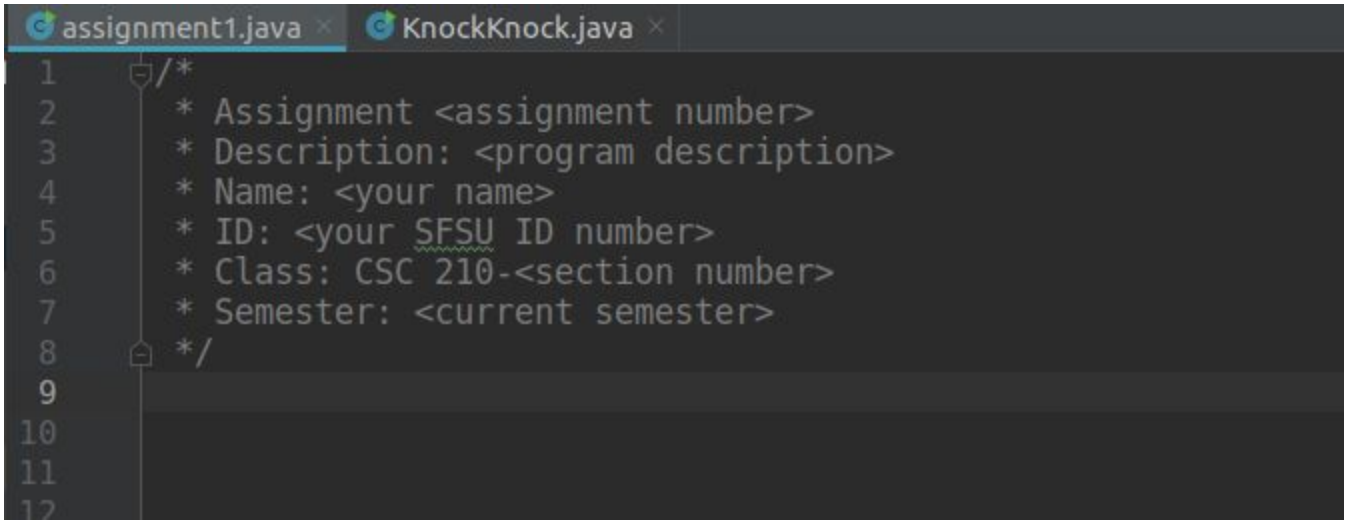
b. Only if you work with a Study Buddy, include your Buddy's name in your header at the top of every Java file. Figure 1

Header Format
/* * Assignment **<assignment number>** * Description: **<program description>** * Name: **<your name>** * Teammate: **<Study Buddy name>** * ID: **<your SFSU ID number>** * Class: CSC 210-**<section number>**

```
 * Semester: <current semester>
 */
```

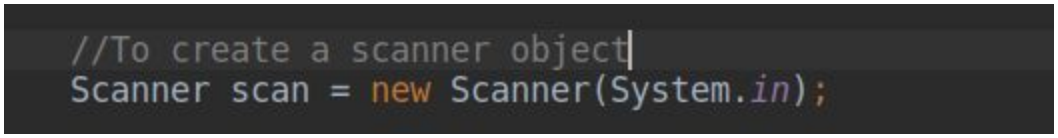
Replace each tag (such as **<assignment number>**) with the appropriate text.
You should adhere to this format as closely as possible. You do not need to include the **<>** symbols in your header fields.

Figure 1



b. Place your comments at the top of **each** Statement, however you don't need to comment print (i.e. anything that starts with System.out.print…) statements. An example of commenting codes is included below in Figure 2:

Figure 2



❒Part 5: Reflection 200 words [18 points]

*Points will be deducted for less than 200 words.

Please put what was helpful and what was not helpful in working on this program.
And also please tell me how you would improve your own program from part 3, if you were given more time.

Only if you work with a Study Buddy, write down how your buddy helped you, i.e. Don helped me learn a new technique for understanding OOP, by thinking about containers.

Place your reflection in this Google Form

The reflection below is identical to the one submitted via Google Forms.

The instructions given in the assignment were quite clear. The tips on how to use the Scanner class and storing the values read from stdin were very detailed. I have written code before so it was easy for the most part, although it has been a while since I've used Java so it took me a little bit to get used to Java syntaxes again after being spoiled by Python for so long. So this assignment was an opportunity for me to get used to some of the syntaxes and expressions that aren't available in Java and those that are specific to it (oh, how I've missed you Python f-strings).

Since this assignment was supposed to be simple, I have not implemented any methods to sanitize the user's input in the programs. For the most part, the programs would work fine if the inputs received are as expected. If I had more time, I would check the user's input to make sure that the program runs smoothly and doesn't crash when it receives an unexpected input. For example, in the "Knock Knock" generator, you would check that the user has entered a specific prompt (e.g., "Annie who?") before proceeding. And for the BMI calculator and the math game, I would make sure that the input that was being given was of an expected type (e.g., int, double, etc.) before reading them in in order to prevent them from crashing.

Unfortunately, Amber's schedule hasn't quite lined up with mine. And it definitely didn't help that I live in Bangkok so our time zones didn't match at all. So we didn't get an opportunity to help each other during this assessment. Hopefully, we are able to make it work next time…