

Assignment 2 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: **09-14-2020 at 11:59 PM**

WHAT TO SUBMIT

Submit 4 files to iLearn by the deadline 09/14 11:59PM PST. [48pts + 2 E.C. pts = 50 points]

- 2 Files: Please submit 2 files to iLearn: **DrivingExamEvaluator.java**, <YourOwnIdea>.java. **[40 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-2-report.pdf". Fill this out with screenshots and your reflection, then save it as Word or PDF **[8 points]**

How TO SUBMIT

Please upload all 3 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 2

Selections

ABOUT THIS PROJECT

You will be creating a program that will process the scores that a student receives during a driving exam.

You will not be creating the driving exam itself.

The program is intended to process the scores after the exam has been completed.

Program Input:

- The program will ask for the student's ID number, the written exam score, and the practical exam score.

Program Output:

- Then, the overall score will be calculated using a specific formula designed below.
- Once the overall score is computed, the program will let the user know whether or not they passed.

Part 1: Driving Exam Evaluator [28 points]

In this project, you will be creating a program that will process the scores that a student receives during a driving exam. You will not be creating the driving exam itself. The program is intended to process the scores after the exam has been completed. The program will ask for the student's ID number, the written exam score, and the practical exam score. Then, the overall score will be calculated using a specific formula designed below. Once the overall score is computed, the program will let the user know whether or not they passed.

FILE NAME REQUIREMENTS

Java file name: `DrivingExamEvaluator.java` Main class name: `DrivingExamEvaluator`

INSTRUCTIONS

1. Collect the information

Your first task is to collect the following information from the user:

- The student's ID number, as a String. The program should validate the input to ensure that it does not exceed 9 characters. If it does, then the program should ignore the user input and automatically set the ID number as a “default” variable integer number, which you declared and assign the value “999999999”;
- The written exam score. See the paragraph below for required validation.
- The practical exam score. See the paragraph below for required validation.

For both the written exam score and practical exam score, the minimum possible value is 0 and the maximum possible value is 500. For both of these scores, if the user attempts to enter a value that is less than 0, then the program should ignore user input and set the score as 0. If the user attempts to enter a value that is greater than 500, then the program should ignore user input and set the score as 500.

NOTE: This means, that you will have to re-assign the scores to either 0 or 500.

Also please print out what the user’s final scores are for both written and practice, AND when you have re-written it so the user know that they provide an invalid score and you had to modify it to stay within the valid values of 0 and 500.

2. Compute the overall score

The next step is to calculate the overall score. You do this by taking into account the following weights:

- The written exam score makes up 74 percent of the overall score.
- The practical exam score makes up 26 percent of the overall score.

You will need to calculate the overall score based on the two scores and their respective weights.

3.Display the summary

Finally, you will be displaying the results of the exam to the user. The following results should be displayed:

- The student's ID number.
- The written exam score (in terms of percentage).
- The practical exam score (in terms of percentage).
- The overall score (in terms of percentage).
- A message telling the student if they passed or failed the exam. In order to pass the exam, the student's overall score must be at least 52 percent of the raw maximum score of 500. If the student does not satisfy this requirement, then that means they fail the exam.

The program should be finished once this information is displayed. It is up to you to decide how you would like the information to be displayed. The only requirement is that you present the exact values that are requested above, and the summary must be easy for the user to understand. Aside from that, you can be creative with the design and arrangement of the summary.

Output Passing

```
/usr/lib/jvm/java-14-oracle/bin/java -Didea.launcher.port=36459 -Didea.launcher.bin.path=/home/in
Enter your student id number
66666666666666666666
Your id number is 999999999
Enter your written exam score
502
Your written score is rewritten to: 500
Enter your practical exam score
200
Your practice score is 200
Final score is for writing: 370.0 and practice: 52.0
Congratulations for finishing your test!
For Student ID: 66666666666666666666
Your written score is, weighted 74% of total score: 74.0%
Your practice score is, weighted at 26% of total score : 10.4%
Your total score is: 84.4%
Congratulations, you passed! A passing grade requires your total score to be over 52%

Process finished with exit code 0
```

Output NON Passing

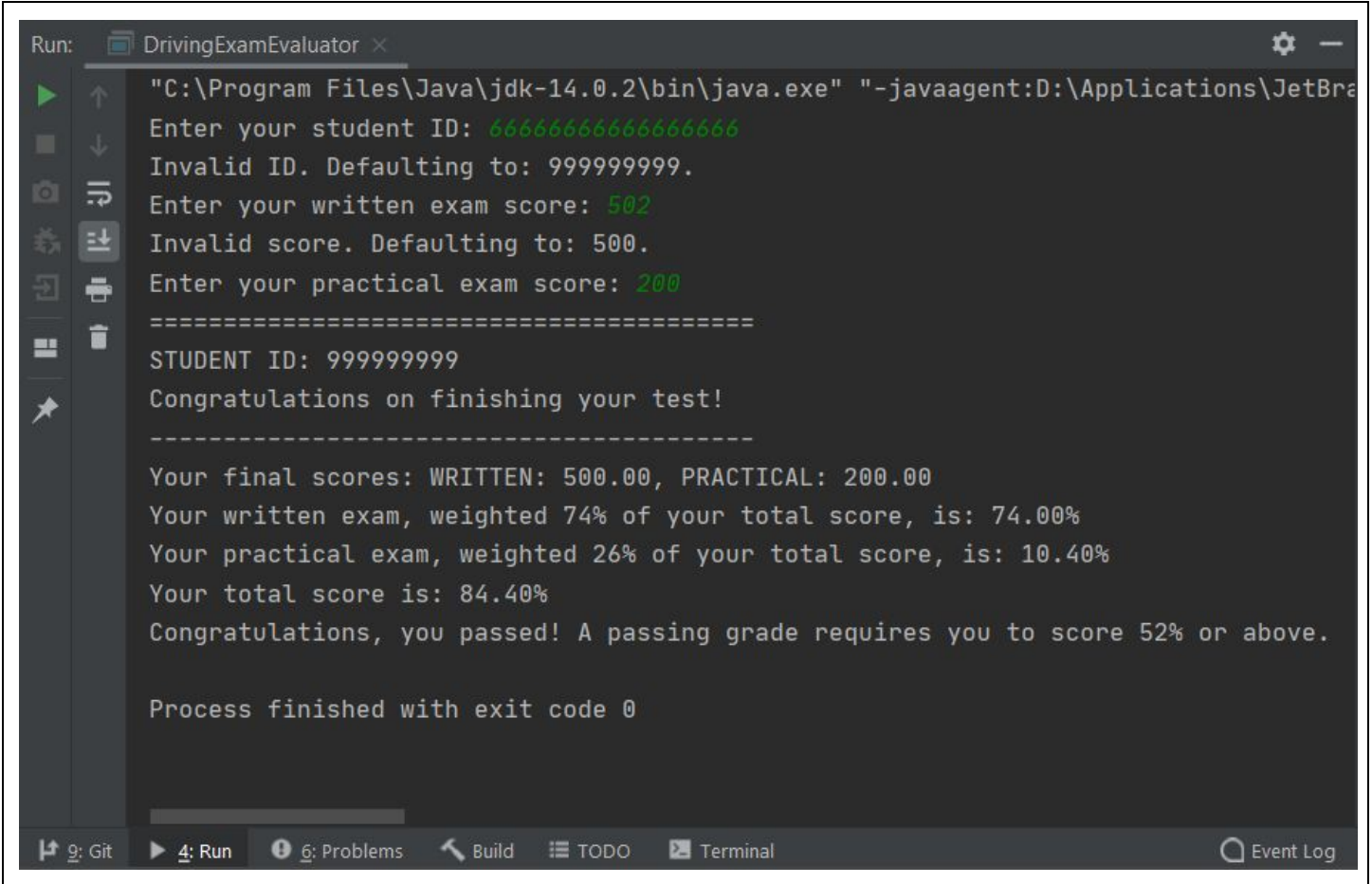
```
DrivingExam x
/usr/lib/jvm/java-14-oracle/bin/java -Didea.launcher.port=40801 -Didea.launcher.bin.path=/ho
Enter your student id number
66666666666666666666
Your id number is 999999999
Enter your written exam score
100
Your written score is 100
Enter your practical exam score
200
Your practice score is 200
Final score is for writing: 74.0 and practice: 52.0
Congratulations for finishing your test!
For Student ID: 66666666666666666666
Your written score is, weighted 74% of total score: 14.799999999999999%
Your practice score is, weighted at 26% of total score : 10.4%
Your total score is: 25.2%
I'm sorry, you did not pass! A passing grade requires your total score to be over 52%

Process finished with exit code 0
```

SUBMISSION INSTRUCTIONS

1. Submit the 1 **DrivingExamEvaluator.java** file directly on iLearn
2. Take a screenshot of the output of your program for passing and non passing and paste it here

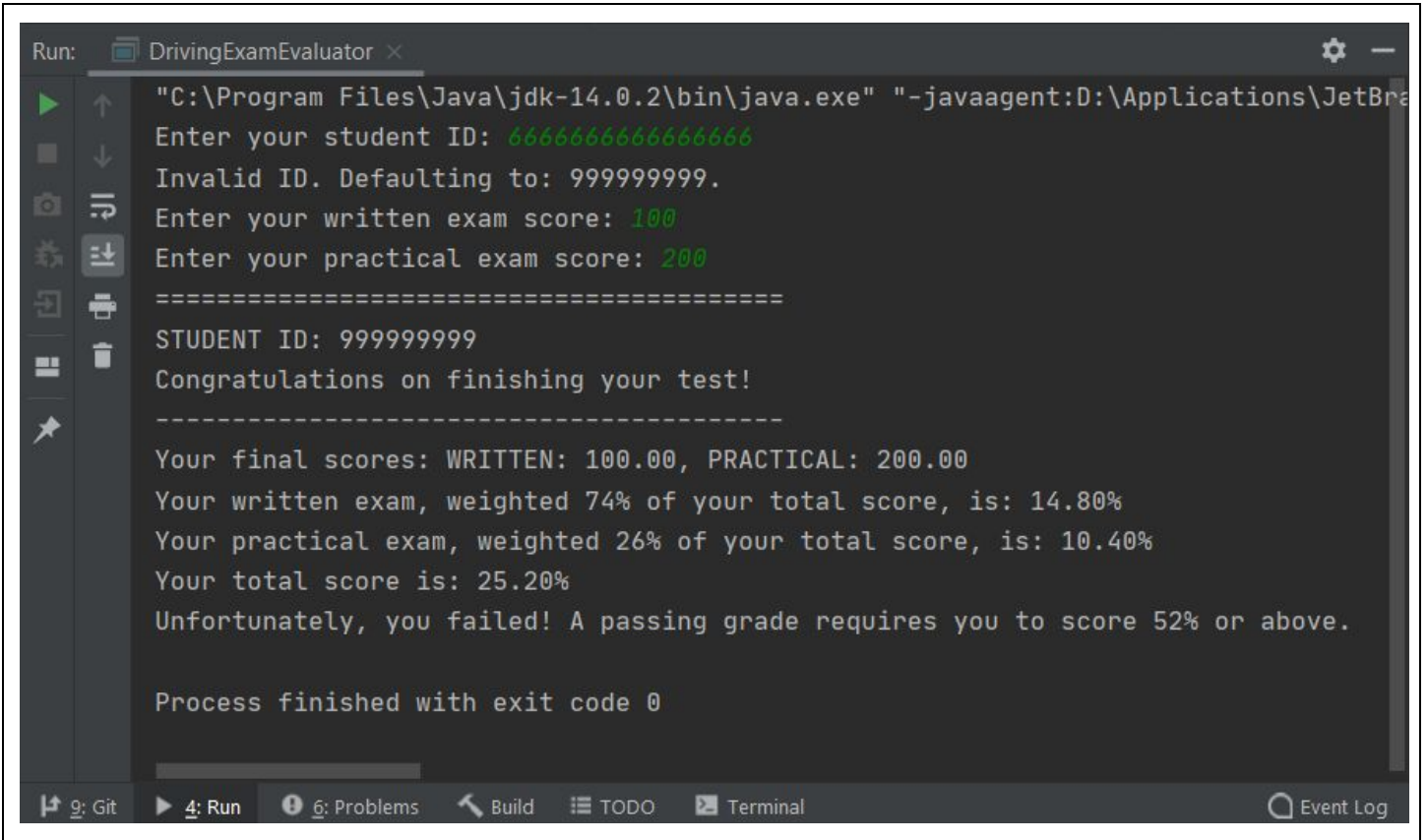
Output for passing :



```
Run: DrivingExamEvaluator x
"C:\Program Files\Java\jdk-14.0.2\bin\java.exe" "-javaagent:D:\Applications\JetBra
Enter your student ID: 6666666666666666
Invalid ID. Defaulting to: 999999999.
Enter your written exam score: 502
Invalid score. Defaulting to: 500.
Enter your practical exam score: 200
=====
STUDENT ID: 999999999
Congratulations on finishing your test!
-----
Your final scores: WRITTEN: 500.00, PRACTICAL: 200.00
Your written exam, weighted 74% of your total score, is: 74.00%
Your practical exam, weighted 26% of your total score, is: 10.40%
Your total score is: 84.40%
Congratulations, you passed! A passing grade requires you to score 52% or above.

Process finished with exit code 0
```

Output for not passing :



```
Run: DrivingExamEvaluator x
"C:\Program Files\Java\jdk-14.0.2\bin\java.exe" "-javaagent:D:\Applications\JetBra
Enter your student ID: 6666666666666666
Invalid ID. Defaulting to: 999999999.
Enter your written exam score: 100
Enter your practical exam score: 200
=====
STUDENT ID: 999999999
Congratulations on finishing your test!
-----
Your final scores: WRITTEN: 100.00, PRACTICAL: 200.00
Your written exam, weighted 74% of your total score, is: 14.80%
Your practical exam, weighted 26% of your total score, is: 10.40%
Your total score is: 25.20%
Unfortunately, you failed! A passing grade requires you to score 52% or above.

Process finished with exit code 0
```

9: Git 4: Run 6: Problems Build TODO Terminal Event Log

☐ Part 2: Your own idea [10 points]

It can be similar to part 1.

Please use what you learned from Selections.

- a. In Assignment 1, Part 3: Your own idea, you created your own idea of a program. For this assignment, you must improve upon your first program by adding what you learned from Selections
- b. Must include Selections of 2 or more different conditions, using “if”, “else if”, or “if” and “else”
- c. (Optional) you can use “switch”
- c. And, must output different results based on the selection to the user
- 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 run for **all conditions** and paste it here

Explanation:

This is a number guessing game. The execution loop is as follows:

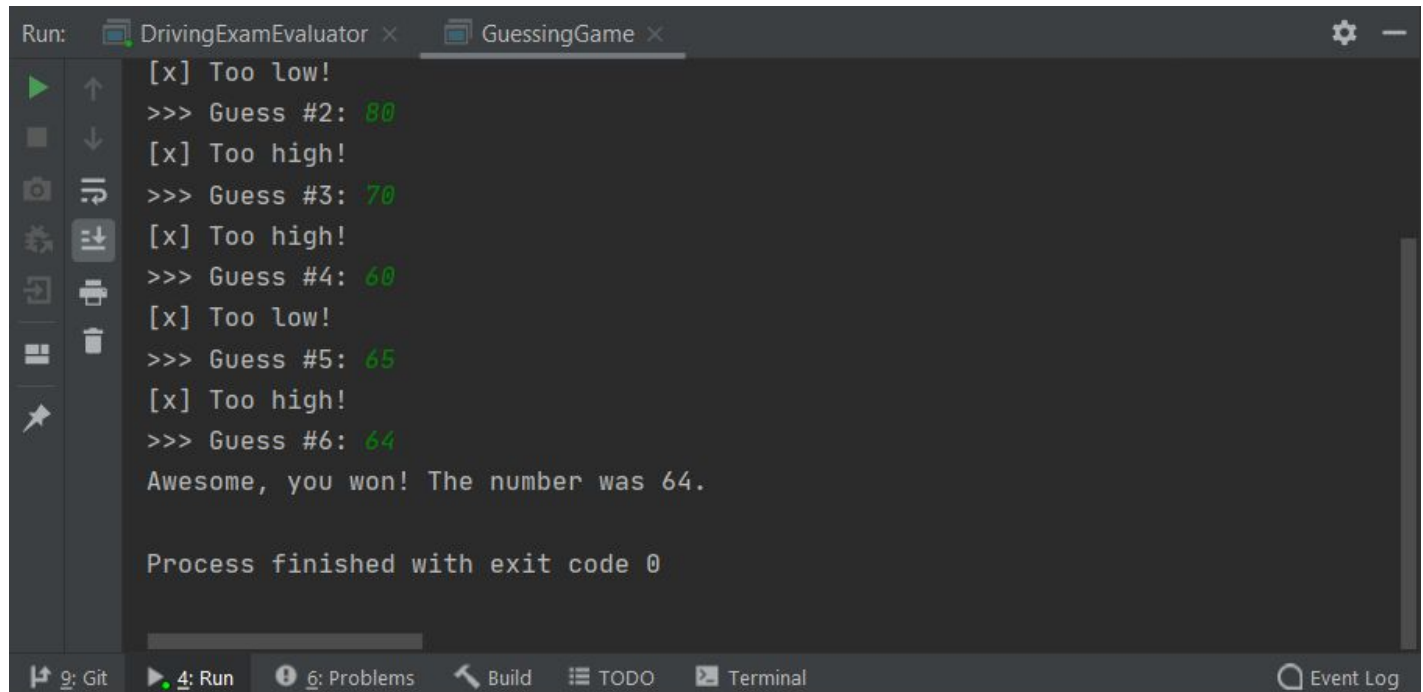
1. The program generates a pseudorandom integer between a certain range (1–100 inclusive, by default).
2. Asks the user for a guess between (1–100 by default).
3. The program checks if the user has selected the correct number by checking the difference between the target number and the user’s guess. The user is given a hint, if:
 - $\text{delta} > 0$, then the number is too high
 - $\text{delta} < 0$, then the number is too low
4. The loop (step 2–3) continues until the user has correctly guessed the number ($\text{delta} = 0$) or

when they have run out of attempts (10, by default).

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

Example 1:

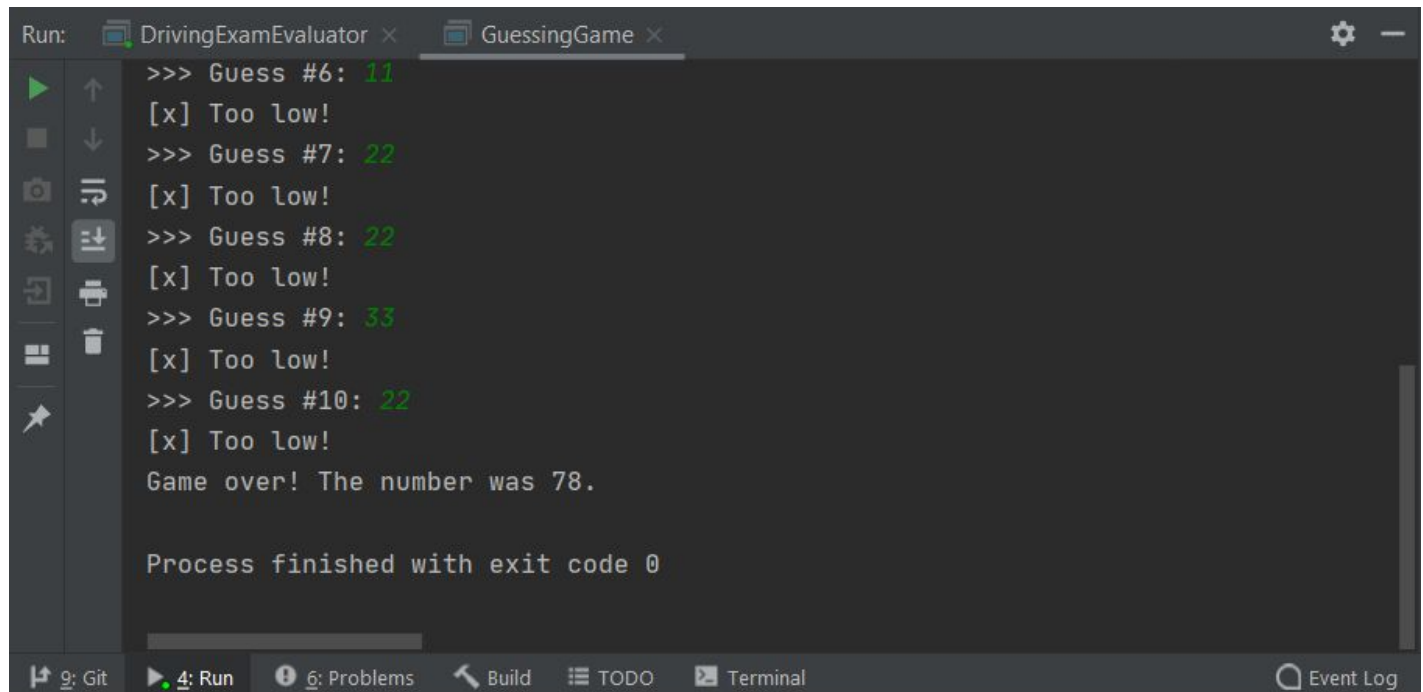
When the user guesses the correct number before they run out of guesses (win)



```
Run: DrivingExamEvaluator x GuessingGame x
[x] Too low!
>>> Guess #2: 80
[x] Too high!
>>> Guess #3: 70
[x] Too high!
>>> Guess #4: 60
[x] Too low!
>>> Guess #5: 65
[x] Too high!
>>> Guess #6: 64
Awesome, you won! The number was 64.

Process finished with exit code 0
```

Example 2 and 3 are on the following page.

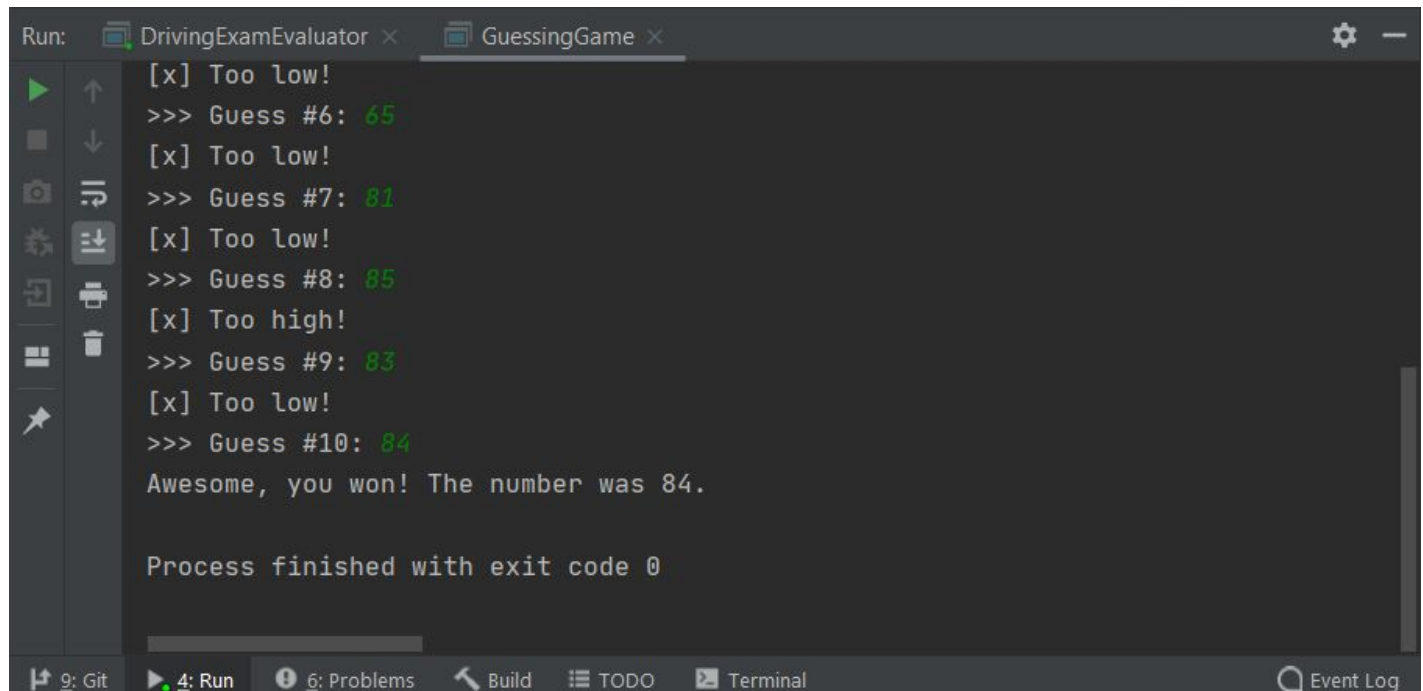
Example 2:**When the user runs out of guesses; fails to guess the right number (loss)**

The screenshot shows a terminal window with two tabs: 'DrivingExamEvaluator' and 'GuessingGame'. The 'GuessingGame' tab is active. The terminal displays the following text:

```
>>> Guess #6: 11
[x] Too low!
>>> Guess #7: 22
[x] Too low!
>>> Guess #8: 22
[x] Too low!
>>> Guess #9: 33
[x] Too low!
>>> Guess #10: 22
[x] Too low!
Game over! The number was 78.

Process finished with exit code 0
```

The terminal window has a sidebar on the left with various icons (run, debug, etc.) and a bottom status bar showing '9: Git', '4: Run', '6: Problems', 'Build', 'TODO', 'Terminal', and 'Event Log'.

Example 3:**When the user guesses the correct answer on the last guess (win)**

The screenshot shows a terminal window with two tabs: 'DrivingExamEvaluator' and 'GuessingGame'. The 'GuessingGame' tab is active. The terminal displays the following text:

```
[x] Too low!
>>> Guess #6: 65
[x] Too low!
>>> Guess #7: 81
[x] Too low!
>>> Guess #8: 85
[x] Too high!
>>> Guess #9: 83
[x] Too low!
>>> Guess #10: 84
Awesome, you won! The number was 84.

Process finished with exit code 0
```

The terminal window has a sidebar on the left with various icons (run, debug, etc.) and a bottom status bar showing '9: Git', '4: Run', '6: Problems', 'Build', 'TODO', 'Terminal', and 'Event Log'.

❑ Part 3: 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
<pre>/* * Assignment <assignment number> * Description: <program description> * Name: <your name> * ID: <your SFSU ID number> * Class: CSC 210-<section number> * Semester: <current semester> */</pre>

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.

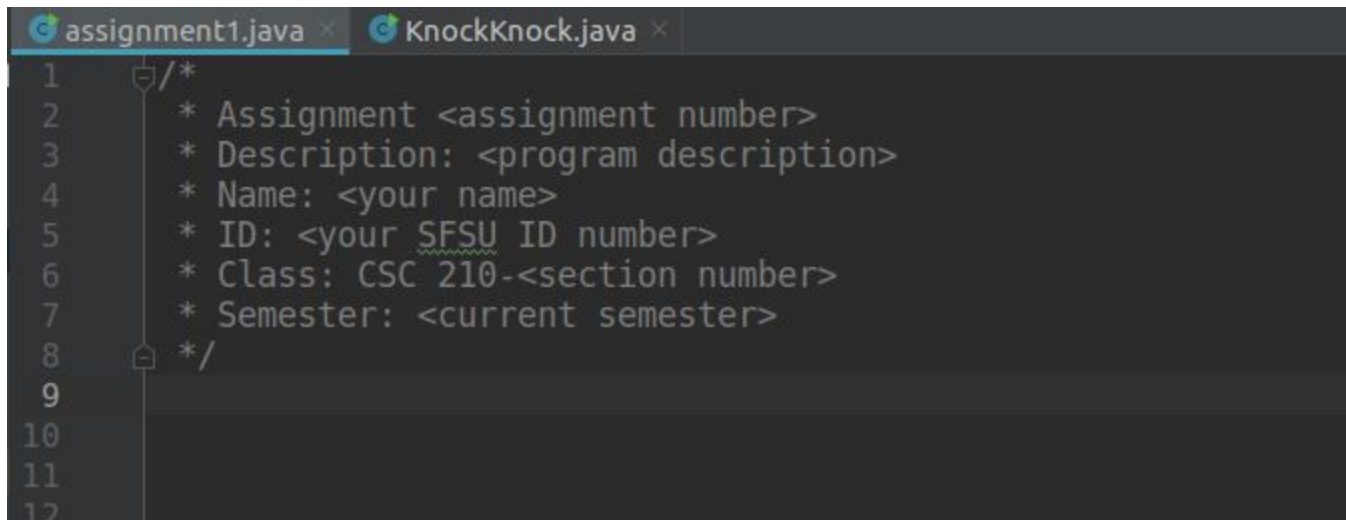
b. Only if you work with a Study Buddy, include your Buddy's name in your header at the top of every Java file. Figure 1

Header Format
<pre>/* * 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> */</pre>

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.

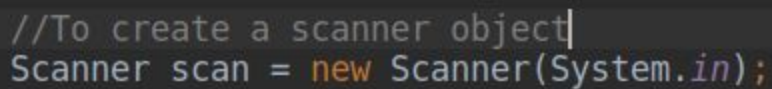
Figure 1



```
1  /*
2      * Assignment <assignment number>
3      * Description: <program description>
4      * Name: <your name>
5      * ID: <your SFSU ID number>
6      * Class: CSC 210-<section number>
7      * Semester: <current semester>
8      */
9
10
11
12
```

- 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



```
//To create a scanner object|
Scanner scan = new Scanner(System.in);
```

☐ Part 4: Reflection 50 words [8 points]

*Points will be deducted for less than 50 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 on Google Forms.

The instructions were clear and quite straightforward. There was a slight ambiguity on what constituted as a “pass” for the driving exam program, but that was quickly resolved via Slack. Again, this assignment was kind of a refresher for me on Java syntaxes and features. I’ve also learned to write a more consistent Javadoc and other documentation methods.

I was helping Amber with her assignment and I found out that she was completely new to coding. I worked with her and found a way to better explain things.