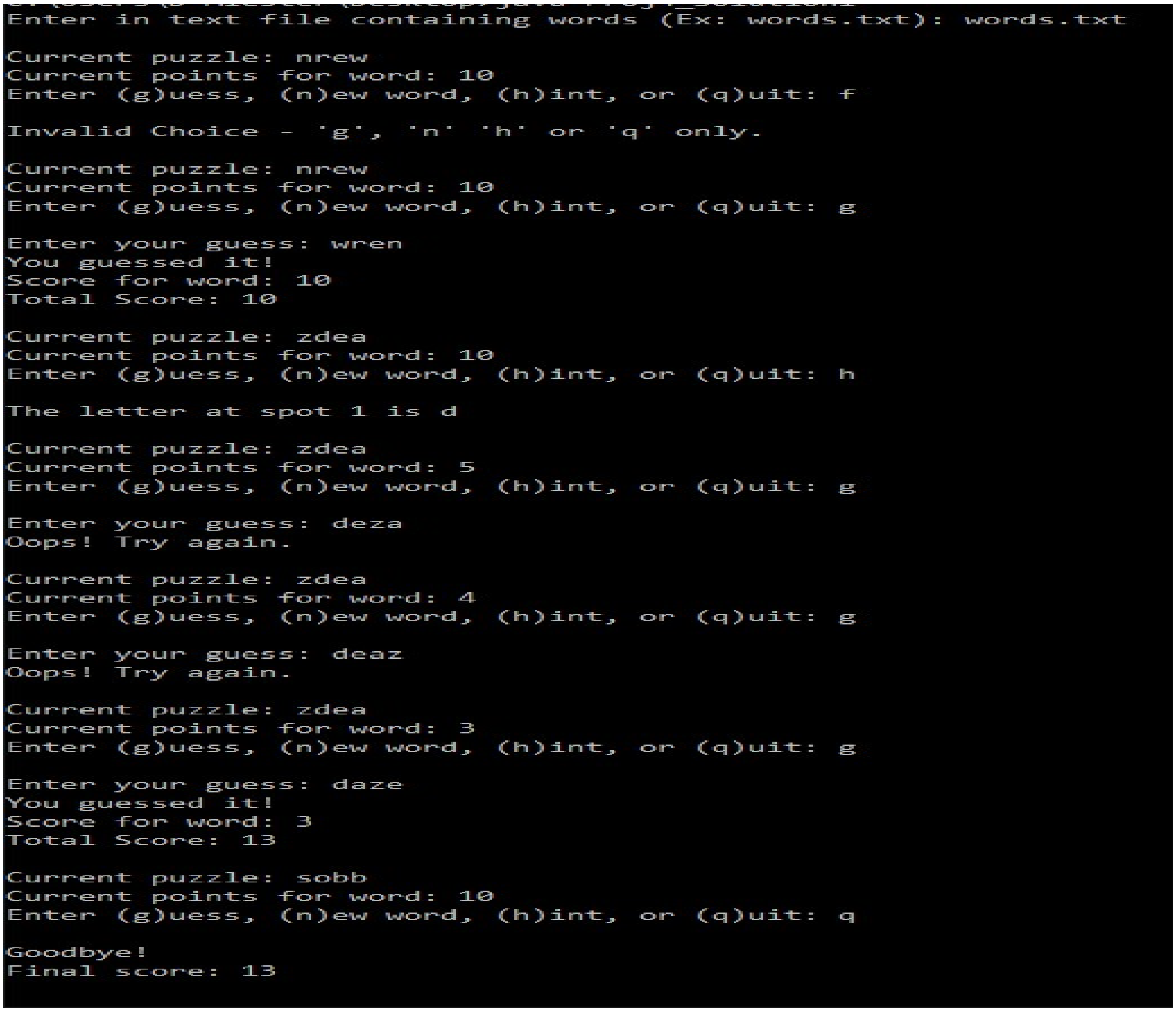
CIS 200: Project 4 (50 points)
Due Friday, February 26th by midnight

Reminder: Programs submitted after the due date/time will be penalized 10% for each day the project is late (not accepted after 3 days, i.e. Midnight, Mon, Feb 29th).

Reminder: **ALL projects are intended to be done *individually*. Do NOT share your code with anyone.** If it is not your work, don't submit it as yours! Refer to the policy within the course syllabus which states, ***"If two (or more) students are involved in ANY violation of this policy,*** **at a minimum, *ALL* students involved receive a *zero*** for the assignment and the offense is officially reported to the *KSU Honor Council*. The second offense results in a failing grade for the course and possible suspension from the university (this decision is made by the *K-State Honor Council*)."

It is strongly recommended that you do some '*pre-planning*' before you code (i.e. I-P-O, **algorithm**, etc.) The difficulty lies in the logic and algorithm more than the coding.

Assignment Description:
In this assignment, you will write a *jumble* game. You will repeatedly pick a random word, randomly jumble it, and let the user guess what it is. The user will also be able to get hints in case they have trouble figuring out the word. ***Here is an example run of the program:***


```
Enter in text file containing words (Ex: words.txt): words.txt

Current puzzle: nrew
Current points for word: 10
Enter (g)uess, (n)ew word, (h)int, or (q)uit: f

Invalid Choice - 'g', 'n' 'h' or 'q' only.

Current puzzle: nrew
Current points for word: 10
Enter (g)uess, (n)ew word, (h)int, or (q)uit: g

Enter your guess: wren
You guessed it!
Score for word: 10
Total Score: 10

Current puzzle: zdea
Current points for word: 10
Enter (g)uess, (n)ew word, (h)int, or (q)uit: h

The letter at spot 1 is d

Current puzzle: zdea
Current points for word: 5
Enter (g)uess, (n)ew word, (h)int, or (q)uit: g

Enter your guess: deza
Oops! Try again.

Current puzzle: zdea
Current points for word: 4
Enter (g)uess, (n)ew word, (h)int, or (q)uit: g

Enter your guess: deaz
Oops! Try again.

Current puzzle: zdea
Current points for word: 3
Enter (g)uess, (n)ew word, (h)int, or (q)uit: g

Enter your guess: daze
You guessed it!
Score for word: 3
Total Score: 13

Current puzzle: sobb
Current points for word: 10
Enter (g)uess, (n)ew word, (h)int, or (q)uit: q

Goodbye!
Final score: 13
```

For initial development and testing, create an array of ***ten*** 4 and 5-letter words. Get your program working for this short list of words first. You will be able to easily add the File I/O portion described below later when it is discussed in class Week 6.

The program will initially ask the user for an input file. An input file has been provided for you within Canvas (**`words.txt`**) within the Projects folder, but **your program should work with ANY correctly formatted text file**. Download the file and open it. Make sure you understand its format. The first line contains the number of words in the file (use this to initialize your array size). This is followed by a long list of *4- and 5-letter words*. Your program will read in all the words in the text file into a String array. You can assume the file is located in the same working directory as your code/*class* file.

Requirements:
This program should contain a single class (called `Proj4`) with a `main` method. Your program must compile (by command-line) with the statement: **`javac Proj4.java`**

It must then run with the command: **`java Proj4`**

Picking/Jumbling a Word:

You need to *develop your own algorithm* and *write your own code* for this portion of the program. You can seek assistance from other students, the internet, GTAs, etc. but **do not copy code**. No "pre-built" shuffle method will be accepted for credit. Document in your code any sources used from the internet.

You should randomly pick a new word for your game at the following times:

- When the program starts
- After the user has correctly guessed the previous word
- If the user indicates that they want a new word

I recommend using a random number generator (see Project 3) to randomly pick an index in your array of words (which you read from the input file).

Once you have picked the current word, you will need to randomly jumble its letters. We will discuss one technique for jumbling a word during class, but any process that randomly mixes up the letters is fine. You will want to use a random number generator for this step as well.

Keep in mind that it is possible to randomly jumble the word and end up with the original word. If the jumbled word is the same as the original word, jumble it again. Also, you may find that several words in the input file contain the same letters. (For example, *meat* and *team*.) If the original word that your program picked was *meat*, and the user guesses *team*, then it is fine if your program says that the guess was incorrect. (It is also ok if you detect this occurrence and give credit to *team* since it uses all the letters and is also a word in the file, but this is definitely not required nor will extra points be awarded for doing so.)

Giving a Hint:

If the user selects that they want a *hint* on the current word, then you should randomly pick one of the positions in the "correct" word for the round and give the user the character at that position. Use a random number generator.

If the user gets several hints on the same word, your program may end up giving them the same hint more than once (since the character chosen is random). That's fine, but if you want to modify your program to give them a different hint each time, then that's fine too.

Keeping Score:

Your program should keep track of the user's total score and their score for the current word. The user should start with 10 possible points for each new word. Every time they guess incorrectly, they should lose a point (unless their possible points are already at 0, in which case they should stay at 0). Every time they get a hint, the number of possible points for that word should be divided by two (using integer division). If the user guesses the word correctly at some point, the remaining possible points for that word should be added to the total score.

For example, if the user makes two wrong guesses, needs a hint, and then guesses the word correctly, then they should get 4 points for that word added to their total score. They lose two points for the two incorrect guesses and then the remaining 8 points are divided by two when the user needs a hint.

Error Checking: The only required error checking is verifying that the user enter one of the menu choices (upper OR lowercase) – 'g' (guess), 'n' (new word), 'h' (hint) or 'q' (quit). If invalid choice, display error message "Invalid Choice - 'g', 'n' 'h' or 'q' only".

Documentation:

At the top of ***EACH*** class, add the following comment block, filling in the needed information:

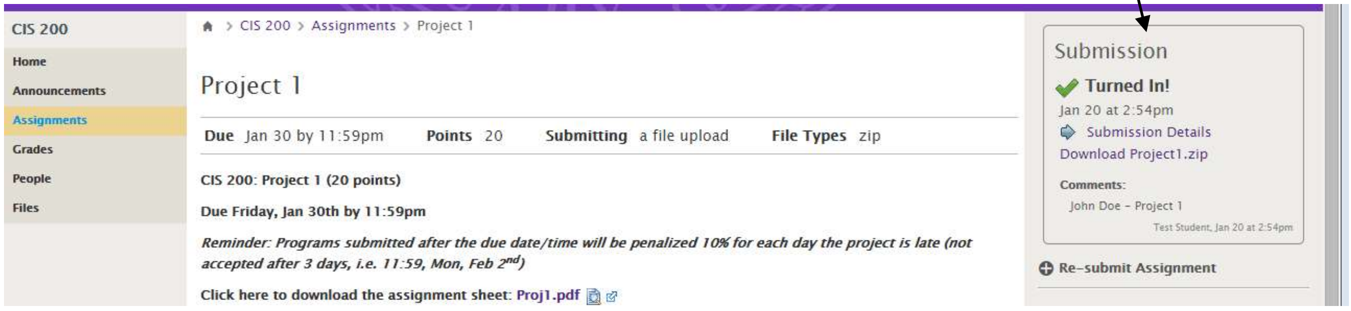
```
/**
   * <Full Filename>
 * <Student Name / Section Day/Time>
 *
 * <description of the project – i.e. What does the program do?>
*/
```

Submission – ***read these instructions carefully or you may lose points***

To submit your project, create a folder called **`Proj4`** and copy or move your *.java* file (**`Proj4.java`**) into that folder. **Do not need to submit the text file.** Then, right-click on that folder and select "***`Send To → Compressed (zipped) folder`***". This will create the file ***`Proj4.zip`***.

Log-in to Canvas and upload your ***`Proj4.zip`*** file. **Only a .zip file will be accepted for this assignment in Canvas. Put your full name and Project 4 in the comments box.**

Important: It is the ***student's responsibility*** to verify that the ***correct*** file is ***properly*** submitted. If you don't properly submit the *correct* file, it will not be accepted after the 3-day late period. No exceptions.



CIS 200
Home
Announcements
Assignments
Grades
People
Files

CIS 200 > Assignments > Project 1

Project 1

Due Jan 30 by 11:59pm **Points** 20 **Submitting** a file upload **File Types** zip

CIS 200: Project 1 (20 points)

Due Friday, Jan 30th by 11:59pm

Reminder: Programs submitted after the due date/time will be penalized 10% for each day the project is late (not accepted after 3 days, i.e. 11:59, Mon, Feb 2nd)

Click here to download the assignment sheet: Proj1.pdf

Submission
✓ **Turned In!**
Jan 20 at 2:54pm
Submission Details
Download Project1.zip
Comments:
John Doe – Project 1
Test Student, Jan 20 at 2:54pm
Re-submit Assignment

Grading:

Programs that do not compile will receive a grade of 0, so make sure you submit the *correct* file that *properly compiles*. Programs that *do* compile will be graded according to the following rubric:

Make sure and test your solution with both the given text file and another that you create with, maybe, only 20 words. GTAs will be testing your program using both!

Requirement	**Points**
Proper Documentation / ***Project Submission*** (zip file with .java file, submitted to correct folder with *Name* and *Project 4* in the description box)	**2**
Correctly reads ***words.txt*** input file into a String Array	**3**
Repeatedly gets user option (*guess*, *new word*, *hint*, or *quit*); Invalid error message if menu choice is not chosen	**5**
Randomly picks and jumbles word at start of game, after a correct guess, and if the user wants a new word.	**5**
Algorithm/Code to jumble the word developed by the student – no 'pre-built' methods (i.e. *if you didn't develop it, consider it pre-built*)	**5**
Guess option works correctly	**4**
Hint option works correctly	**4**
Quit option works correctly	**2**
Score is kept correctly (overall and for the current word)	**3**
Program output exactly matches screenshot	**2**
Correctly works when a different text file is used	**5**
Minus Late Penalty (10% per day)	
Total	**40**