CIS 200: Project 5 (50 + 5 points)
Due Friday, Mar 11th 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, March 14th)

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:

To simulate a *real world* programming project, the IT department gives you a *general description* of what is needed to help automate *scantron grading*.

Assume a n-question scantron exam is given in a class. When the scantron sheets are run through the scantron machine, the machine simply scans each sheet and records the Student's 9-digit WID number and the answers from the sheet, one per line, in a text file. Each line in the file is delimited by a comma. The answers are recorded as a series of numbers, where 1=true, 2=false (on the true/false portion of the quiz) and A=1, B=2, C=3, D=4, E=5 on the multiple choice section.

For example, say a 15-question exam is given to 5 students, the file generated would look like:

```
819283761, 2112211241
834987341, 2221213241
899991828, 1122113241
888181818, 2121113131
892823736, 1111111111
```

The professor will then use that text file as input into *your* program that will do the following:

- Ask the user for the name of the text file. If the file does *NOT* exist, display an error message and have the user enter in the file name until a valid file name is entered.

- Ask how many questions were on the exam and the number of points per question.

- Allows the user to create an answer key that will be used to grade the exams. The user will enter 'T' for true, 'F' for false, or the character A-E for multiple choice questions. Of course, number of answers is dependent on what the user enters for the number of questions given on the exam.

-Read in the scantron generated text file (***QuizScores.txt*** - available in Project folder on Canvas), '*grade*' the quizzes and display each student's WID number, followed by the number correct, percentage correct (to one decimal), score and their grade based on the percent correct (see below).

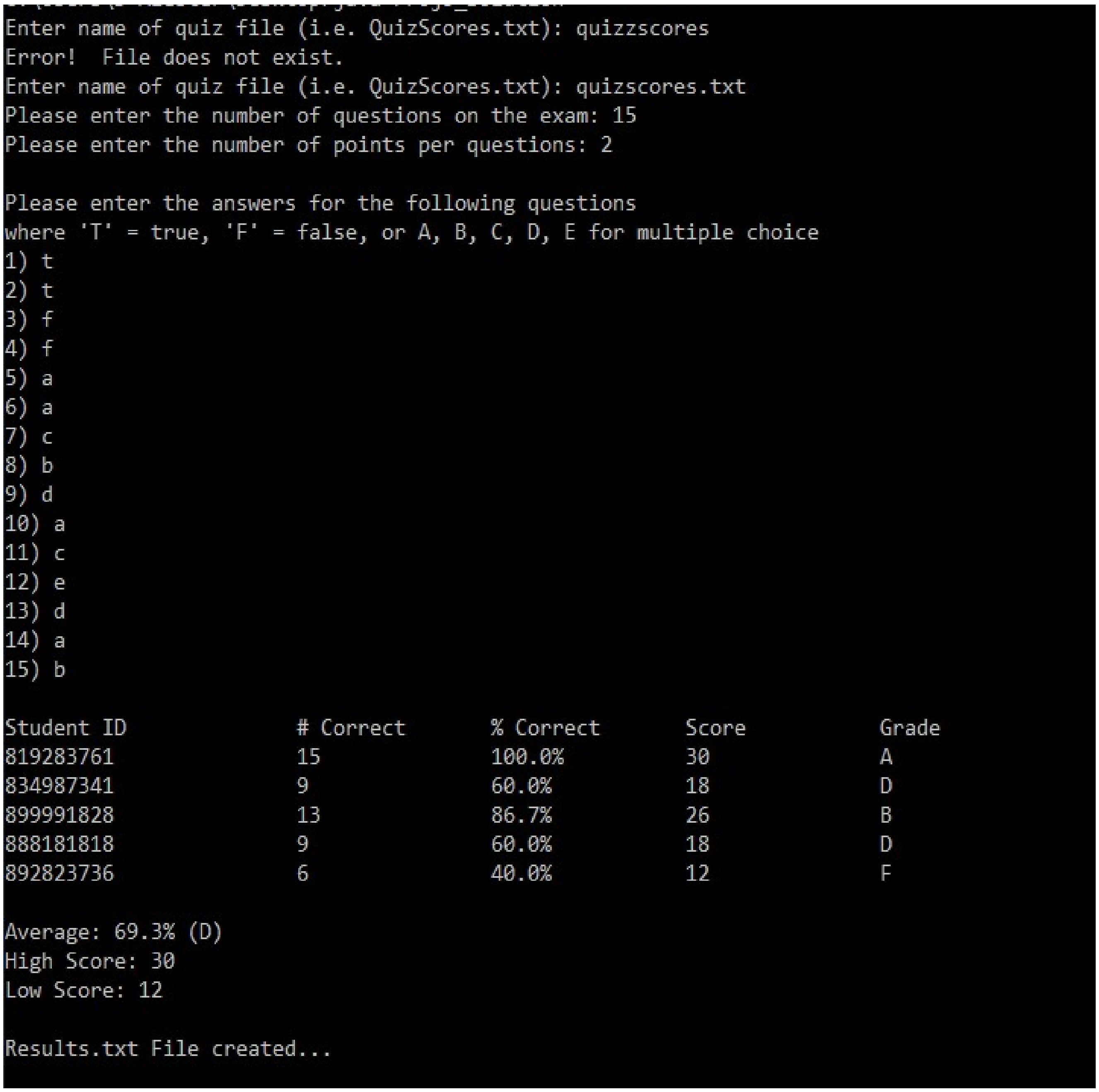
90-100	**A**
80-89.9	**B**
70-79.9	**C**
60-60.9	**D**
Below 60	**F**

- After displaying all student information, display the class average on the quiz to 1 decimal with the grade in parenthesis, the high score and the low score.

- Lastly, create a comma delimited text file called ***Results.txt*** containing only the WID, # Correct, % Correct, Score, and Grade, one student per line. You file would look like the following:

```
819283761,15,100.0,30,A
834987341,9,60.0,18,D
899991828,13,86.66666666666667,26,B
888181818,9,60.0,18,D
892823736,6,40.0,12,F
```

Here is an example run of the program, based on the given *QuizScores.txt* file. Your output screen should appear exactly as shown to maximize points:



```
Enter name of quiz file (i.e. QuizScores.txt): quizzscores
Error!  File does not exist.
Enter name of quiz file (i.e. QuizScores.txt): quizscores.txt
Please enter the number of questions on the exam: 15
Please enter the number of points per questions: 2

Please enter the answers for the following questions
where 'T' = true, 'F' = false, or A, B, C, D, E for multiple choice
1) t
2) t
3) f
4) f
5) a
6) a
7) c
8) b
9) d
10) a
11) c
12) e
13) d
14) a
15) b

Student ID              # Correct       % Correct       Score           Grade
819283761               15              100.0%          30              A
834987341               9               60.0%           18              D
899991828               13              86.7%           26              B
888181818               9               60.0%           18              D
892823736               6               40.0%           12              F

Average: 69.3% (D)
High Score: 30
Low Score: 12

Results.txt File created...
```

Here are some additional *requirements* and *tips*:

- Although *QuizScores.txt* only contains 5 students, ***your program must work for any number of students in the file*** (from 1-50).
- Although the example shown contains 15 questions, ***your program must work for any number of questions*** (from 1-50).
- **You are *NOT* allowed to use *class* (i.e. *global*) variables in your program**. Only variables local to methods are allowed. This forces you to correctly pass values back and forth between the methods.

To get *full* credit, your program *must include* the following methods:

Since the intent of this project is the *creation* and *implementation of methods*, you must include working implementations of at least ***6*** of the ***9*** methods listed below and call and use those methods to be considered for *partial credit* (i.e a program with less than 6 methods will not be considered for any credit)

1) Method that reads in name of the file and validates the file exists (otherwise, loop until valid).
2) Method that reads in the information from the file
3) Method that creates the key to grade the exam
4) Method that 'grades' a single quiz and returns the number correct for that SINGLE quiz
5) Method that returns the appropriate letter grade based on the percentage
6) Method that calculates and returns the overall average on the quiz
7) Method that displays the required output to the screen
8) Method that creates the ***Results.txt*** file
9) A ***main*** method that basically is an *outline* of method calls.

This is the ***minimum*** number of methods required. You can, of course, have more. I would suggest you use the "*divide and conquer*" approach – get one method working before you start another.

Before you code, I strongly recommend that you (*at a minimum*) determine the *method signature* for each of the methods described above and decide on what you want to happen within each method.

Documentation: Put a description of the project at the top of the **file** **and** at the top of **each method**.

Please use this template for the top of the **file:**

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

Please use this template for the top of each **method**:

```
/** Method name
 * (description of the method)
 *
 * @param (describe first parameter)
 * @param (describe second parameter)
 * (list all parameters, one per line)
 * @return (describe what is being returned)
 */
```

<u>**OPTIONAL Extra Credit**</u> **(+5 pts.)**
Extra Credit is an all or nothing option – it either works correctly (+5) or it doesn't (+0)

Develop your ***own*** algorithm to ***sort* by the *percent correct*** (descending, high to low) before you both display your output *and* write to the file. Although not required, I would suggest you develop another method to handle the sorting.

Note: You are ***NOT* allowed to use *Arrays.sort*** for the extra credit. ***You must develop your own algorithm to sort*** (i.e. use a *bubble* sort, *exchange* sort, etc.)

<u>Requirements</u>

This program should contain a single class (called Proj5) with a main method. Your program must compile (by command-line) with the statement: **javac Proj5.java**

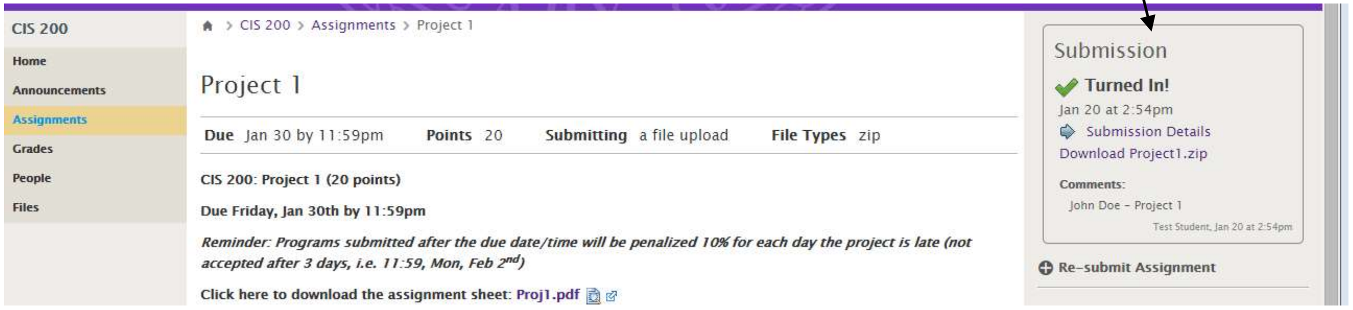
It must then run with the command: **java Proj5**

Submission – *<u>read these instructions carefully or you may lose points</u>*

To submit your project, create a folder called ***`Proj5`*** and copy or move your *.java* file (**`Proj5.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 ***`Proj5.zip`***.

Log-in to Canvas and upload your ***`Proj5.zip`*** file. <u>**Only a .zip file will be accepted for this assignment in Canvas**</u>**. Put your full name and Project 5 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:

Reminder - Make sure and test your solution with both the given text file and another that you create with, maybe 10 students with 5 questions. GTAs will be testing your program using an additional file.

Programs that do not compile or don't contain at least 6 of the 9 methods described 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:

Requirement	Points
***Proj5 (Part 1)* – Execution (25 pts.)**	
Properly verifies input file exists	**2**
Properly reads in number of questions and points	**1**
Properly allows user to create answer key	**4**
Produces correct screen output including Quiz Average (1 decimal) and high/low scores using given *QuizScores.txt* file	**7**
Correctly creates comma-delimited file *Results.txt*	**4**
Produces correct results using *Instructor created text file* with a differing number of students and questions	**7**
Proj5 Code (Part 2) – Contains the following methods that MUST be called to get any credit (no GLOBAL variables allowed) **(25 pts.)**	
Method that reads in name of file and validates file exists (otherwise, loop until valid)	**2**
Method that reads in the information from the file	**3**
Method that creates the key to grade the exam	**3**
Method that 'grades' a single quiz and returns number correct for that SINGLE quiz	**3**
Method that returns the appropriate letter grade based on the percentage	**2**
Method that calculates and returns the overall average on the quiz	**2**
Method that displays the required output to the screen	**3**
Method that creates the ***Results.txt*** file	**3**
A ***main*** method that basically is an *outline* of method calls.	**1**
Proper Documentation on both the File Header and ON EACH METHOD	**2**
Proper Submission with full name & project # in Comment box. Correct Filename (Proj5.java)	**1**
Extra Credit: Output is sorted by *% correct* (high to low). Must develop OWN algorithm to sort – use of *Arrays.sort* not allowed. **Extra Credit is an all or nothing option** – it either works correctly (+5) or it doesn't (+0)	**+5**
Minus Late Penalty (10% per day)	
Total	**50 + 5**