CIS 200: Project 8 (50 points + 10% Extra Credit)
Due Fri, Apr 24th by 11:59pm
LAST JAVA Project!

(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, Apr 27th)

Reminder: Plagiarism software is being used in this course on all source code submitted. As stated in the *Academic Integrity Policy* in the syllabus for this course, unless otherwise stated, ALL projects are intended to be done *individually*. Do NOT share your code with anyone or submit any work that is not your own. 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*)."

Note: If you use code that YOU did not develop (i.e. from another text or found it on the Internet) *you must cite your source* in a comment above the line(s) of code that is not your own work, otherwise it will be considered *plagiarism* (*representing others' work, whether copyrighted or not, as one's own*).

Assignment Description: This project is a modification of your most recent lab (Lab 10) requiring a simple modification of the *Student* class, modification of the data validation section, adding checking for duplicates and then using a HashMap to easily search for entered students.

First, modify your **Student** class from Lab 10, as follows:

- Add a private data property to hold a student's 9-digit WID number. Realize that the WID could contain preceding zeroes (i.e. 009374625) and that is must not contain any characters.
- Add an *equals* method that test two *Student* objects for equality, based on WID number.
- Make additional modifications as needed, including your *Constructor* and *toString* method to generate the desired output, as shown on the next page.

Next, the **Proj8** class is a modification of the class created for Lab10. It should contain the following:

- Create an *ArrayList of Students* and allow the user to enter as many students as they wish, as shown on the next page. No credit will be given for a program that does not utilize an ArrayList.
- Preform the following data validation. You can decide how the following will be done, using either *exception handling*, a *validation loop*, or both. However, as in the extra credit portion for the lab, structure your error handling so user does NOT need to re-enter valid data. For example, if *name* is valid but *user name* is not, re-enter *user name* only. If *name* and *user name* is valid but WID is not, re-enter *WID only,* and if *GPA* is not valid, then re-enter *GPA only*. This will take some thought and carefully planned out logic.

  Data validation will include…

  - SOMETHING must be entered for the *name, username,* and *WID* (can't just press enter). If the user just hits enter on any of these, display an error message and force user to re-enter.
  - WID must contain 9-digits and must contain *only* digits (no characters), else display an error message and force user to re-enter.
  - Display an error message and force user to re-enter value if a *non-numeric* value is entered for GPA (i.e. a char or a String).
  - Only allow valid GPA values (between 0.0-4.0 / inclusive), else display an error message and force user to re-enter.
- After a data validation is complete and all values are valid, use your *equals* before storing a Student object into the Arraylist, to avoid duplicate students. If a duplicate is found, simply display "*Duplicate Student found*" and have the user re-enter the information.

- Once ALL input has been entered into the ArrayList, use the *toString* method in the *Student* class to display each object in the Arraylist, displayed as shown below. (You do NOT need to display one object at a time, as we have done on previous projects.) When all have been displayed, display the message "*All students displayed*."
- Lastly, add all Students stored in the *ArrayList* to a *HashMap*, using the WID as the key value.

Hint: Use a loop to add the WID of each student (as the key) along with that student's object (the value).

- Using the HashMap, allow the user the ability to *search* for as many students as they desire (one at a time) by requesting a *WID Number*. If found, display as shown on below. If not found, display message shown below. If user just presses enter, display message shown below. Have user enter 'exit' to quit (you only need to check for lowercase 'exit').

** No credit for a search that does not utilize a HashMap **

Example run of the program. Your screen should appear exactly as shown to maximize points.

```
Enter the student's name: <Enter pressed by user>
Name is required (Please re-enter)

Enter the student's name: <Enter pressed by user>
Name is required (Please re-enter)

Enter the student's name: John Doe
Enter student's USER name: <Enter pressed by user>
User name is required (Please re-enter)

Enter student's USER name: <Enter pressed by user>
User name is required (Please re-enter)

Enter student's USER name: JDoe
Enter student's WID #: <Enter pressed by user>
WID is required (Please re-enter)

Enter student's WID #: 123
WID must contain 9-digits - no characters (Please re-enter)

Enter student's WID #: 12345678J
WID must contain 9-digits - no characters (Please re-enter)

Enter student's WID #: J98765432
WID must contain 9-digits - no characters (Please re-enter)

Enter student's WID #: 123456789
Enter student's GPA: 32
GPA must be between 0.0-4.0 (inclusive).
Please re-enter: 33
GPA must be between 0.0-4.0 (inclusive).
Please re-enter: 3.3
Student added to the ArrayList...

Add another student? ('Y' or 'N'): y
Enter the student's name: Bill Smith
Enter student's USER name: BSmith
Enter student's WID #: 987654321
Enter student's GPA: 4
Student added to the ArrayList...
```

```

Add another student? ('Y' or 'N'): y
Enter the student's name: Harold Jones
Enter student's USER name: HJ
Enter student's WID #: 123456789
Enter student's GPA: 3.2
Student already exists.

Add another student? ('Y' or 'N'): n

John Doe
WID #123456789
JDoe@ksu.edu
GPA: 3.3

Bill Smith
WID #987654321
BSmith@ksu.edu
GPA: 4.0
All students displayed.

Search for a Student by entering Student's WID number
(or enter "exit" to exit the program.)
Input Student WID # (or "exit"):
Nothing entered
Input Student WID # (or "exit"): 123
Student not found
Input Student WID # (or "exit"): 123456789
Student Info:
John Doe
WID #123456789
JDoe@ksu.edu
GPA: 3.3
Input Student WID # (or "exit"): 987654321
Student Info:
Bill Smith
WID #987654321
BSmith@ksu.edu
GPA: 4.0

Input Student WID # (or "exit"): exit
```

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

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

```
/**
* <Full Filename>
* <Student Name / Lab Section Day and Time>
*
* <COMPLETE description of the project – i.e. What does the program do? Must be detailed enough so
outside reader of your code can determine the specifics of the program>
*/
```

Use this template for the top of each method (including your constructors):

```
 /** 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)
 */
```

Requirements

This program will contain TWO separate class files (3-4 if doing the extra credit) and EACH must compile (by command-line) with the statement: **javac <filename>.java**

It must then run with the command: **java Proj8**

Make sure and test this before submitting. Submit ALL needed files and submit the CORRECT files (i.e. files that compile and run). It is very important that you learn to submit what is needed to your *client*.

OPTIONAL Extra Credit Challenge: (10% extra credit – +5 points)

Make the following modifications to add to the functionally to Project 8. You will submit a SINGLE version of this Project to Canvas. If doing the extra credit, simply add the functionality (requires additional file(s) to be submitted). You can do either #1 or both #1 *and* #2 below. Please indicate in a comment or when you submit the assignment if you did the extra credit (#1 or Both) so that the GTA knows to test your program for these options.

1) Use *MVC architecture* by creating a separate class to handle the VIEW using Console I/O. Then use the VIEW class to handle all I/O within the Project. There will be no print statements or input statements in the Controller class (Proj8). (FYI: *Student* will be your *model* class). (3 pts.)

2) Using *MVC architecture*, by creating *another* separate class to handle the VIEW using GUI – this will give you a total of 4 classes. Since this is a GUI, you can take some leeway in the final output, as long as it is relatively close and is complete. Again, no I/O statements in the Controller class (Proj8). Include a line that is commented out that creates a *VIEW_GUI* object rather than a View_Console object. GTAs should be able to comment out one line and uncomment the other and the view should change from Console to GUI. (2 pt.)

Submission – read these instructions carefully or you may lose points

Programs that do not compile will receive a grade of ZERO, regardless of the simplicity or complexity of the error, so make sure you submit the *correct* file that *properly compiles*.

To submit your project, first create a folder called *Proj8* and copy ALL *.java* files into that folder. Then, right-click on that folder and select "*Send To → Compressed (zipped) folder*" to create the file *Proj8.zip*

Log-in to Canvas and upload your *Proj8.zip* file. Only a .zip file will be accepted for this assignment in Canvas. Put your full name and Project 8 in the comments box.

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

Grading: Since the grader will be testing your program using different data, make sure and test your solution with both the given data and other data.

Only what it submitted to Canvas before the deadline will be considered for grading, so submit the CORRECT files. Files can be re-submitted until the deadline but only the LAST submission is graded.

Programs that do not compile/run from the command line will receive a grade of ZERO, regardless of the simplicity or complexity of the error, so make sure you submit the *correct* files that *properly compile from the command line.* Programs that *do* compile/run will be graded according to the following rubric:

Requirement	**Points**
**To be considered for grading, program must include a working ArrayList	
Documentation – includes documentation on EACH file (class) and above EACH method (including Constructors) in all classes	**2**
Student.java (Code) – WID data property and equals method properly added. Constructor and toString method properly modified.	**7**
Proj8.java (Code) – Create and properly use an *ArrayList* of *Student* objects, adding each object to the ArrayList. Properly uses equals method to check for duplicates.	**8**
EXECUTION – Allows user to enter info for as many Students as desired and does the following data verification: SOMETHING must be entered for the *name, username,* and *WID* (can't just press enter); WID must contain 9-digits and must contain *only* digits (no characters); does not allow a *non-numeric* value for GPA (i.e. a char or a String) or values outside of 0.0-4.0. Must be structured so user does NOT need to re-enter valid data	**11**
Properly checks and does not allow duplicates before adding object to the ArrayList	**5**
Properly displays all students currently in the ArrayList using the *toString* methods of the Student class.	**5**
Add all Students stored in the ArrayList to a HashMap. Must use a HashMap to get credit for the search potion. Allows user to SEARCH for as many students as desired – properly displays the found student or message "student not found". Also handles just enter key being pressed. Quits on "exit" to end.	**12**
Extra Credit #1: MVC architecture is used…Console VIEW class created. No print or input statements in the controller class (Proj8)	**+3**
Extra Credit #2: Additional GUI VIEW class created. No I/O statements in the controller class (Proj8). GTA comment out creation of a GUI_VIEW object.	**+2**
Minus Late Penalty (10% per day)	
Total	**50 + 5**