Project 8 (40 + 5 points)
Due Friday, Apr 20th 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. 11:59pm, Mon, Apr 23rd)

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)."

Avoid temptation to violate policy out of desperation – **YOU** must understand the concepts covered in this project to be successful on the next exam and in subsequent course work.

Assignment Description: Download and look over the provided **Payroll** class. You may modify this class however you see fit. Since this is the final **Java** Project, **you** will mostly choose how to design your program (i.e. *what* goes *where*) and the design/content of your output screen. However, your program must contain the following:

1) Within your **PayrollApp** class:

- Create and properly use an **ArrayList of *Payroll* objects**. You will allow the user to enter the information for as many employees (i.e. *payroll* objects) as they would like. You can choose how to end input. Add each object to the *ArrayList*. Input will include all info needed to create a *Payroll* object.

** Since a primary objective of this project is to work with ArrayLists, NO credit will be given for a program that does not utilize an **ArrayList** **

2) Within your **Payroll** class, add the following method: (*feel free to add more, as needed…*)

-***toString***: used to display the ***name***, ***idNumber***, and ***employee's pay*** for the week with a $-sign and 2-decimals. Must be declared as ***public String toString()*** and contain no *println*s).

3) You will decide where to do the following (**Payroll** or **PayrollApp** class)

a) **Data Validation: Use Exception Handling** to check for and *properly handle* the exceptions below. To properly handle an exception (*if it occurs*), display an error message and allow the user to re-enter input until the exception is no longer thrown. Since a secondary objective of this project is to work with exceptions, you must use **exception handling** to get points for this portion of the assignment.

-A *character* or *double* is entered for *idNumber* (int)

-A *character* is entered for *payRate or hoursWorked* (double)

-An *empty string* (i.e. enter is pressed) for the employee's name, throw an exception (yes, this is better handled with a *while* loop but for practice purposes, *throw* and *catch* an exception)

- **Once data is validated** and all input has been read in from the user and stored in the *ArrayList*, **use the *toString* method** on each object in the *ArrayList* to display the required output.
- **Once all output is displayed**, prompt the user to enter a 6-digit *idNumber* – **if found**, ***delete*** that employee from your *ArrayList* and indicate that the employee has been deleted from the *ArrayList*. **If *not* found**, display an appropriate error message and allow user to re-enter until a valid id number is entered. (One option is to create a *HashMap* from the info in the ArrayList for an easy search, but how you achieve this search is up to you).

- Lastly, allow the user to enter in one additional *employee* and place that employee at the END of the *ArrayList*. Again, use your *toString* method to display all current employees in the *ArrayList*.

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:

Please use this template for the top of the file:

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

Please use this template for the top of each method:

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

Extra Credit (+5 pts / +12.5%): Make the following modifications to add 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 (#2 requires an additional file to be submitted). You can do either #1 **or** #1 **and** #2 below. Please indicate when you submit the assignment if you did the extra credit (#1 or both) so that the GTA knows to look for / test your program for these options.

1) Use **MVC architecture** by creating a separate class to handle the VIEW (Console I/O). Then use the VIEW class to handle **all** I/O within the Project – **no print statements or input statements in the Controller class** (PayrollApp). (FYI: Payroll is the *model* class). (3 pts.)
2) Use MVC architecture by creating **another** separate class to handle the VIEW (GUI) – this will give you a total of 4 classes. From the command line, user will type in **java PayrollApp Console** to use the *Console* View class or **java PayrollApp GUI** to use the *GUI* view class. Ignore case within your program (GUI, gui, Gui, etc.). Again, **no I/O statements in the Controller class** (**PayrollApp**). (If done properly, *changing the view* should have very minimal effect on the Controller class). (2 pts.)

This program will contain at least TWO classes (possibly 3 or 4, if doing the extra credit) – **Payroll** and **Payroll App**. All classes must compile (by command-line) with the statement: **javac filename.java**

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

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

Submission – read these instructions carefully or you may lose points

To submit your project, first create a folder called Proj8 and move **ALL** completed class files into that folder. Then, right-click on that folder and select "Send To->Compressed (zipped) folder". This will create file **Proj8.zip**. 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 ALL of the **correct** files are **properly** submitted. If you don't properly submit the *correct* file, *it will not be accepted after the 3-day late period*. **No exceptions**.

Grading: Only what it submitted to Canvas before the deadline will be considered for grading.

Programs that do not compile will receive a grade of 0. ***Minimal*** partial credit awarded if ALL classes do not compile. Programs that *do* compile will be graded according to the following rubric:

Requirement	Points
**** No credit will be given for a program that does not utilize an ArrayList ****	
Proper Documentation on the File Header and on *each method* in ALL files / Proper Submission with full name & project # in Comment box. Correct Filenames.	**4**
Payroll.java (Code)	
toString method added and correctly defined	**3**
PayrollApp.java (Code)	
Create and properly use an **ARRAYLIST** of Payroll objects. Add each object to the ArrayList.	**4**
Properly call the methods in the PAYROLL class through the Payroll objects	**4**
Within Payroll.java or PayrollApp.java (Code/Execution)	
Allows user to enter as many employees as they desire	**2**
EXCEPTION HANDLING added to handle a *char* or *double* entered for *idNumber* (int)	**3**
EXCEPTION HANDLING added to handle a *char* entered for *payRate or hoursWorked* (double)	**4**
EXCEPTION HANDLING added to handle an *empty string* entered for the employee's name	**3**
Once all employees initially entered, properly displays all employees currently in the ArrayList using the *toString* method	**4**
Properly prompts for an idNumber, searches ArrayList, and deletes if found; otherwise prompts for a valid idNumber … loops until valid idNumber is entered	**5**
Properly prompts for final employee, adds to the end of the ArrayList, and then displays all current employees in the ArrayList	**4**
Extra Credit: MVC architecture is used; Console VIEW class created; **no print or input statements in the controller class** (PayrollApp)	**+3**
Extra Credit: Additional GUI VIEW class created; **no I/O statements in the controller class** (PayrollApp); user choose an interface at the command line (Console or GUI)	**+2**
Minus Late Penalty (10% per day)	
Total	**40 + 5**