Project 9 (40 points)
Due WED, April 27th 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, SAT, Apr 30th)

Reminder: ALL projects are **intended to be done individually. Don't 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, 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*)."

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 information needed to create a *Payroll* object.

** ***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*** that can be used to display the *name*, *idNumber*, and the *employee's pay* for the week with a $-sign and 2 decimals.

(Reminder: *toString* method 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. 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 (*name*, *idNumber*, and *employee's pay* for the week)

- ***Once all output is shown***, 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 more *employee* and place the employee at the END of the *ArrayList*. Again use your *toString* method to display all current employees in the *ArrayList*.

Documentation: You must put a description of the project at the top of the file **and a description of the method at the top of each method**.

Please use this template for the top of the file:

```
/**
 * (description of the project)
 *
 * @author (your name)
 * @version (which number project this is)
 */
```

Please use this template for the top of each method:

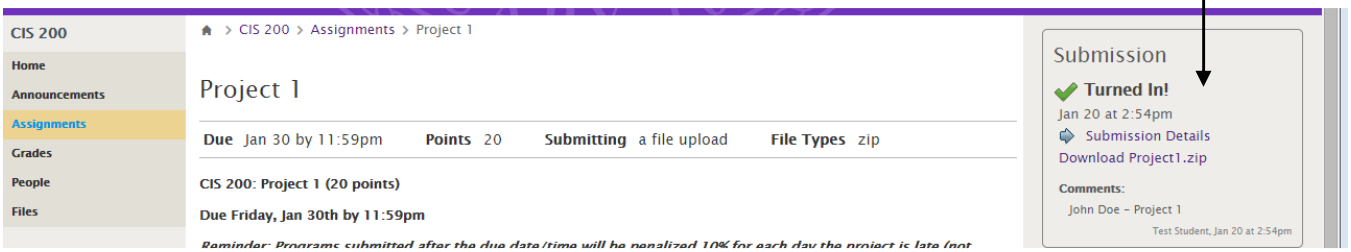
```
/**
 * (description of the method)
 *
 * @param (describe first parameter)
 * @param (describe second parameter)
 * (list all parameters, one per line)
 * @return (describe what is being returned)
 */
```

Submission – *read these instructions carefully*

To submit your project, first create a folder called **Proj9** and copy your completed *Payroll.java* **and** *PayrollApp.java* files into that folder. **Make sure all files are included and compile first**! Then, right-click on that folder and select "*Send To->Compressed (zipped) folder*". This will create file ***Proj9.zip***.

Log-in to Canvas and upload your ***Proj9.zip*** file. Only a .zip file will be accepted for this assignment in Canvas. **Put your name and Project 9 in the comments box. If you did the extra credit,** include this in your comments box when submitting.

Reminder: **It is the student's responsibility to verify that 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.**



Grading: You must submit **BOTH classes** and ***ALL*** must compile to be considered for partial credit. **Programs that do not compile will receive a grade of 0**. 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 *****	
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 *character* or *double* entered for *idNumber* (int)	**3**
Exception handling added to handle a *character* 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	**3**
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**
Documentation – includes documentation on ***EACH file*** and above ***EACH method***	**3**
Project Submission (zip file with name and Project # in description box)	**2**
Minus Late Penalty (10% per day)	
Total	**-40-**