**Assignment 1 Due Date: 7:30 am 2/12**

**Submission Instructions:**

1. Submit your classes saved in a file named: your first name and last initial **AshleyS.java**
2. Submit you Name UML in the text box of the blackboard submission.
3. Submit a brief description of your addition in the text box of the blackboard submission
4. Your file should be able to be compiled using the command: **javac AshleyS.java**
5. I will test your classes with a test class. You do not need to submit your own test class but I would suggest you test your classes before submitted them.

Based on the following class diagram and specifications generate java code.

* All classes should have a **get** and **set** method for each attribute. All classes should also have **toString** and **equals** methods. You should choose public or private for these methods appropriately. You can determine how to check equality for each class.
* The id attribute should be automatically assigned when objects are created. (**Hint:** remember what static does.)
* **Name** is a class that you should design and provide a UML class diagram for. You should have at least 2 constructors. The Name class should make sure that the attributes only contain letters. It should also store the data with the first letters capitalized and the rest lowercase. If the name is invalid the program should generate an **exception** and display some message. The output of the Name **toString** should be: “last,first, m.”
* The **User** constructor that takes the password as a parameter should do some verification. If the password is not at least 8 characters long and does not have at least 2 numbers and 1 uppercase letter it should prompt the user to enter a new password. This process should repeat until a valid password is entered. (**Hint:** remember to handle the case where the User was constructed without a password)
* The **setPassword** method does not need verification because it should only be called from the **changePassword** method.
* **changePassword** should make sure the parameter oldPass and the current password are the exact same. Then it should check that the newPass and newPass2 parameters are the exact same. Lastly it should make sure that newPass is a valid password and update the password attribute.
* computeGpa should return the average of the parameter.
* Undergraduates constructor should check that the year is either freshman, sophomore, junior, or senior. If it isn’t any of these it should be set to unknown.
* All of Graduates attributes should be initialized to 0 and false.
* **updateExamGrade** is used to change the examGrade if it is better than the previous attempt and increase the attempts, or just increase the attempts. This method should also call the **calculateAbleToGraduate** method and return the appropriate String. (“this student is able to graduate” or “this student is not able to graduate”)
* **calculateAbleToGraduate** should set the attribute ableToGraduate to true if the attempts is less than or equal to 2 and the grade is eighty or better.
* The Faculty constructor should use the following conversion table:

|  |  |
| --- | --- |
| dean | 0 |
| chair | 1 |
| professor | 2 |
| secretary | 3 |
| teaching assistant | 4 |
| otherwise | -1 |

* The faculty emails are generated by the first 6 letters of their last names followed by their first initial followed by @newpaltz.edu
* Ex. Thomas Edwards -> [edwardt@newpaltz.edu](mailto:edwardt@newpaltz.edu)

 **toString** method for User:

User id: #

Name: “last,first,m.”

* **toString** for Undergraduate:

User id: #

Name: “last,first,m.”

Year: “…”

* **toString** for Graduate:

User id: #

Name: “last,first,m.”

Exam grade: # with # attempts

* **toString** for Faculty:

User id: #

Name: “last,first,m.”

Position: #

Email: “generated email”

* You may add any extra methods you need to fulfill the above requirements.
* You may explain reasoning for public vs. private or explain equals methods using comments.

**Your own addition:**

You will expand this class system in some way:

* You can add another class
* Add another feature (an attribute with some methods)
* Create a nice test class
* Add another exception

You may use only the following for reference:

* Class notes and examples
* any other docs.oracle.com

<https://docs.oracle.com/javase/7/docs/api/java/lang/String.html> <https://docs.oracle.com/javase/7/docs/api/java/lang/Character.html>

**User**

* name : Name
* id : long
* password : String
* User ( )
* User (n : Name)
* User (n: Name, p : String)
* changePassword (oldPass : String, newPass : String, newPass2 :String) : boolean
* checkPassword ( ) : boolean

**Student**

* Student ( n : Name, p : String)
* computeGpa (grades : double [ ])

**Faculty**

* position : int
* Faculty ( n : Name, p : String, pos : String)
* generateEmail ( ) : String
* convertPosition (p : String) : int

**Undergraduate**

* year : String
* Undergraduate (n : Name, p : String, y : String)
* checkYear( ) : String

**Graduate**

* examGrade : double
* attempts : int
* ableToGraduate : boolean
* Graduate (n : Name, p : String)
* updateExamGrade(g : double) : String
* calculateAbleToGraduate( )

You can fill in the following template for the Name UML and copy and paste it to blackboard submission.