CS603 Programming Assignment 6

Due by end of day on **Wednesday 12/6**

1. Getting Started

This assignment involves an application for managing faculty in a department within a university. The following files have been posted to Blackboard and should be downloaded to the same folder on your computer. Make use of the information provided below concerning each class; additional details are provided in comments within the code.

(1) **FacultyType**: **Abstract** class – superclass for representing the various faculty types. Defines the title instance variable and has one abstract method, promote(), which must be overridden in subclasses. **Leave this class unchanged and do not submit**.

(2) **TenureTrack**: Concrete subclass of FacultyType. Tenure-track faculty members can be promoted and granted tenure.

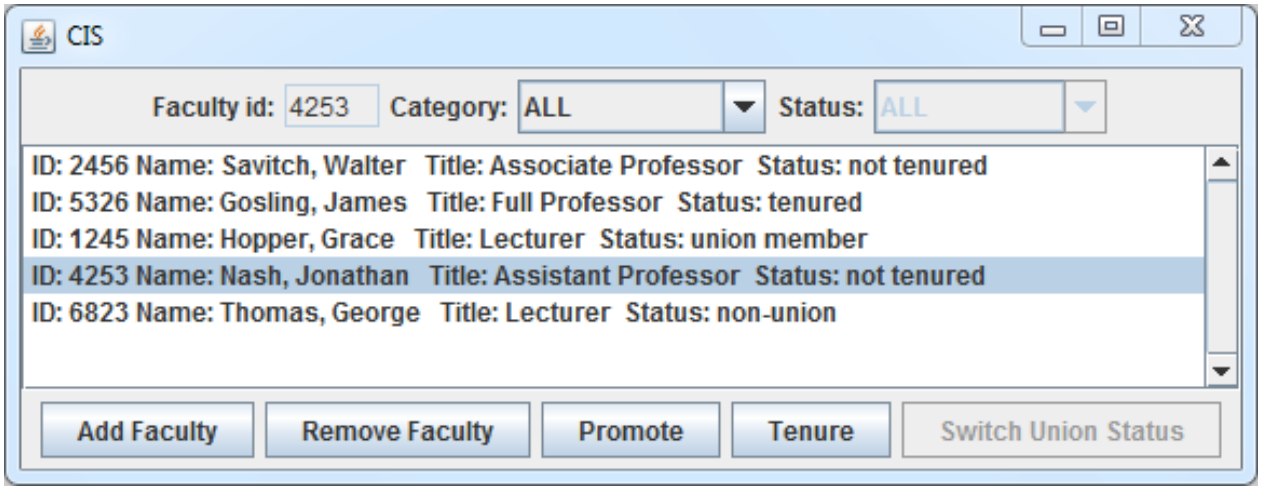
(3) **Adjunct**: Concrete subclass of FacultyType. Adjunct faculty members are not eligible for promotion or tenure, and their title is always lecturer. They are able to join a union if they so choose.

(4) **Faculty**: Represents faculty members by id, first and last names, and faculty type.

(5) **Department**: Maintains an ArrayList of faculty members in a department within a university. Provides functionality for finding a faculty member by id, adding new faculty, removing faculty, and generating lists of faculty by type.

(6) **DeptGUI**: Provides a GUI interface that works with the other classes. **Leave this class unchanged and do not submit**.

You will be adding code to all of the above classes except for FacultyType and DeptGUI. DO NOT make any changes to the instance variables or methods that have already been defined, and do not add any other instance variables or additional accessor (getter) or mutator (setter) methods – you should have all you need. Add your name to the start of each file along with header comments on methods you provide and explanatory comments on your code as needed. **Be sure to test each method after you have written it** (a main method has been provided in every class for this purpose). When your code has been completed, all of the functionality supported by the interface shown below should be working:



2. Programming Assignment (35 points total)

Add the following constructors and methods to the classes provided, as specified below. Be sure the method names, parameter types, and return value type **EXACTLY** matches the specification.

The TenureTrack class (8 points)

1. **2-arg constructor**: the first parameter is of type **String** and the second is of type **boolean**. The two instance variables for a TenureTrack object must each be set to the corresponding parameter value.
2. **promote()**: instance method that overrides the inherited abstract method and returns a **boolean** value of true if the invoking member object is promoted or false otherwise. It must provide the following functionality:
   - A tenure-track faculty member whose title is "Assistant Professor" is promoted. The title changes to "Associate Professor" (exactly) and tenure status changes to true.
   - An associate professor is also promoted. The title changes to "Full Professor" (exactly), while the tenure status changes to true if it wasn't already.
   - A full professor cannot be promoted.
3. **toString()**: instance method that overrides the inherited toString() method. The **string** returned must include the **title** and **tenure status** of the invoking TenureTrack object. Make sure there is **at least one space** between each element (exact spacing up to you). See the examples below:

```
Title: Assistant Professor  Status: not tenured
Title: Associate Professor  Status: tenured
```

Add code to the **main method** to create instances of the TenureTrack class and test your methods.

The Adjunct class (5 points)

1. **1-arg constructor**: one parameter of type **boolean**. The title for the new Adjunct object should be "Lecturer," and the union member status should be set to the parameter's value.
2. **no-arg constructor**: no parameters. The title for the new Adjunct object should be "Lecturer" and the union member status should be true.
3. **toString()**: instance method that overrides the inherited toString() method. The **string** returned must include the **title** and **union status** of the invoking Adjunct object. Again, include at least one space between elements for clarity. See the examples below:

```
Title: Lecturer  Status: non-union
Title: Lecturer  Status: union member
```

Add code to the **main method** to create instances of the Adjunct class and test your methods.

The Faculty class (2 points)

1. **toString()**: instance method that overrides the inherited toString() method. The **string** returned must include the **ID**, **name**, **title**, and **either tenure status or union membership status**, depending on the faculty type. Include at least one space between elements for clarity:

```
ID: 1234 Name: Smith, Jane  Title: Full Professor  Status: tenured
ID: 1224 Name: Jones, Morris  Title: Lecturer  Status: non-union
```

The Department class (18 points)

This class has two instance variables: a string representing the department (such as CIS) and an array list of faculty members.

1. **findID(String)**: this instance method has one parameter of type **String** and returns a reference variable of type **Faculty**. The method must perform a **case-insensitive** search to determine if there is a faculty member in the array list of faculty defined for this class whose ID matches the value of the parameter. If found, return that faculty member. If not found, return a value of null.
2. **add(Faculty)**: this instance method is passed a reference variable of type **Faculty**. If no faculty member in the array list defined for this class has the same ID as the one associated with the parameter value, then add the Faculty parameter to the **end** of the array list. Hint: use the **findID()** method in making this determination. No value is returned by this method.
3. **remove(String)**: this instance method is passed a **string**. If a faculty member in the array list defined for this class has an ID whose value matches the parameter value, then that faculty member must be removed from the list. Note that IDs are unique, so one match at most can be found. Once again, the **findID()** method will be helpful here. No value is returned by this method.
4. List-generating methods: the following **six** instance methods have no parameters. Their purpose is to search the array list defined for this class for members of the type and description specified by the method name and return a new **ArrayList** of type **Faculty** containing any faculty members meeting the search criteria. **If no matching members are found, the list returned should be empty**.
    - **tenureTrackList()**: returns a list of type Faculty containing all tenure-track faculty members.
    - **tenuredList()**: returns a list of type Faculty containing all tenure-track faculty members who have tenure.
    - **notTenuredList()**: returns a list of type Faculty containing all tenure-track faculty members who do not have tenure.
    - **adjunctList()**: returns a list of type Faculty containing all faculty members who are adjuncts.
    - **unionList()**: returns a list of type Faculty containing all adjuncts who are members of the union.
    - **notUnionList()**: returns a list of type Faculty containing all adjuncts who are not members of the union.

The GUI interface should be fully functional after completing the Department class so will provide an additional means for testing.

3. Grading

Your programs must compile and run to receive any credit. Points will be allocated as follows:

8 points for TenureTrack class
- 2 points for 2-arg constructor
- 2 points for toString() method
- 4 points for promote() method

5 points for Adjunct class

- 3 points for constructors
- 2 points for toString() method

2 points for Faculty class – toString() method

18 points for Department class

- 10 points for findID(), add(), and remove() methods
- 8 points for list methods

2 points for meeting the style guidelines

4. Submitting Your Assignment

When you are satisfied that your code meets all requirements, submit **for** files: **TenureTrack.java**, **Adjunct.java**, **Faculty.java** and **Department.java**, via the submit link from the Blackboard Assignments page for Assignment 6. DO NOT submit any other files.