**Project 2: Classes and Objects (Bank Account and Course Registration)**

**Part 1: Bank Account (50 points)**

Write a class to describe a BankAccount object. A bank account is described by:

* the account owner's name
* an account ID (stored as text)
* the balance in the account

**BankAccount Class (35 points)**

In your class, include:

* + instance data variables
  + two constructors
    - one constructor takes an initial balance; the second constructor opens an account with no money
  + getters and setters
    - include appropriate value checks when applicable
  + a toString method
    - you can decide the output
  + a deposit method
    - include appropriate value checks
  + a withdrawal method
    - include appropriate value check

**Driver Program (15 points)**

Write a driver program to demonstrate your class, including:

* + creating bank account objects using both constructors
  + making deposits and withdrawals (with both valid and invalid values)
  + invoking other methods

Note: your driver program does **not** need to be interactive with the user. It only needs to create objects and invoke methods on those objects to demonstrate that your class works.

**Extra Credit (5 points)**

Include an interest rate as part of what describes a bank account. Update the BankAccount class as necessary. You should be able to adjust the interest rate and add interest. Add code to the driver program to demonstrate the interest rate functionality.

**Notes**

* For full credit, follow good principles of programming and object-oriented design.

**Part 2: Course Registration (50 points)**

Write a Student and Course class that might be part of a course registration system.

A student is described by:

* first name
* last name
* student ID (as text)
* whether or not tuition is paid

A course is described by:

* a name
* the maximum number of students that can be enrolled in the class
* a roster of students enrolled in the course (stored as a Student[])

**Student Class (10 points)**

In your class, include:

* + instance data variables
  + a constructor
  + getters and setters
  + a toString method

**Course Class (40 points)**

In your class, include

* + instance data variables
  + a constructor
    - a course is initially created with no students on the roster
  + getters and setters
    - this about which variables should have setters
    - include validity checks where appropriate
  + a toString method
  + an addStudent method
    - method header: public boolean addStudent(Student student)
    - if there is room for a student **and**the student has paid their tuition, add the student to the roster
  + a dropStudent method
    - method header: public boolean dropStudent(Student student)
    - if the student is on the roster, remove them
  + a toString method
    - include the name, number of students enrolled in the course, maximum number that can be enrolled, and a printed roster as part of the text representation
    - make sure that the roster does not print any "nulls"

**Notes**

* You can include additional instance data variables if it is helpful. If you include additional variables, include additional getters and setters as appropriate.
* I have provided a driver program you can use to test your code and sample output.
  + I also provided the sample output generated from the driver program.
  + You might add additional code to the driver to run additional tests to make sure your classes work.
* For full credit, follow good principles of programming and object-oriented design.

**Extra Credit (15 points)**

Write a class called CourseAL. This class has the same methods as Course. Instead of using a Student[] to store the roster, use an ArrayList<Student>. For full credit, take advantage of the methods in the ArrayList class to implement the methods.

**Submission**

* Zip all .java files together and upload that file.
* You will receive no credit for files that do not compile.
* You can work in groups of up to four students on all projects. If you work in a group, submit only one submission and include all names in the comments section of each .java file.
* Late projects can be submitted up to four dates late with a 10% late penalty.