SCRIPTING & PROGRAMMING APPLICATIONS

Competencies:
4016.1.1: Introduction to Java Programming - *The graduate defines programming languages, identifies common algorithms, and identifies the parts of the Java Programming Environment.*
4016.1.2: Object-Oriented Concepts - *The graduate understands the object-oriented programming paradigm and identifies its elements.*
4016.1.3: Variables and Data Types - *The graduate declares, initializes, and assigns values to a variable and differentiates between primitive and object data types.*
4016.1.4: Control Structures - *The graduate utilizes decision and loop constructs to control the flow of a program.*

Scenario:

For this assessment, you will write a program containing two classes named student and roster, respectively. The program will maintain a current roster of students within a given course. Student data for the program includes student ID, first name, last name, e-mail address, age, and array of grades. The program you create will read a list of five students, with one of the students being yourself, and use the series of method calls below (see part B3 below). A well-designed program would use the principles of encapsulation and information hiding. Here is the data that will be used from within the *main* method of your program:

Student ID	First Name	Last Name	E-mail	Age	Grades
1	John	Smith	John1989@gmail.com	20	88, 79, 59
2	Suzan	Erickson	Erickson_1990@gmailcom	19	91, 72, 85
3	Jack	Napoli	The_lawyer99yahoo.com	19	85, 84, 87
4	Erin	Black	Erin.black@comcast.net	22	91, 98, 82
5	Your first name	Your last name	Your valid e-mail address	Your age	3 test scores

The data should be input as follows:

```
String [] students = {"1,John,Smith,John1989@gmail.com,20,88,79,59",
             "2,Suzan,Erickson,Erickson_1990@gmailcom,19,91,72,85",
             "3,Jack,Napoli,The_lawyer99yahoo.com,19,85,84,87",
             "4,Erin,Black,Erin.black@comcast.net,22,91,98,82",
             "5,[firstname],[lastname],[emailaddress],[age],[3testscores]"};
```

Requirements:

A. Include your personal information in the last item of the table above.

B. Create a program that converts the array of Strings shown above to an ArrayList of Student objects. For the `Student` object class, do the following.
   1. Include the following instance variables that describe each student:
      - student ID
      - first name
      - last name
      - e-mail address
      - age
      - array of grades
   2. Include the following methods in the `Student` class:
      a. an accessor (i.e., getter) for *each* instance variable from part B1
      b. a mutator (i.e., setter) for *each* instance variable from part B1

Note: All access and change to the instance variables of the `Student` class should be through accessor and mutator methods.

c. constructor using *all* of the input parameters
d. `print()` to print specific student data (e.g., student ID, first name, last name) using accessors (i.e., getters)

Note: Printing out the grades is optional, not required.

3. Create a student `Roster` class with the following methods that contain all ArrayList method calls:
   a. `public static void add(String studentID, String firstname, String lastname, String emailaddress, int age, int grade1, int grade2, int grade3)` that sets the instance variables from part B1 and updates the roster
   b. `public static void remove(String studentID)` that removes students from the roster by student ID

      Note: If the student ID doesn't exist, the method should print an error message indicating that it is not found.

   c. `public static void print_all()` that prints a complete tab-separated list of student data using accessor methods

      Note: Tabs can be formatted as such: 1 [tab] First Name: John [tab] Last Name: Smith [tab] Age: 20 [tab] Grades: {88, 79, 59}. The `print_all()` method should loop through all the students in the student array list and call the `print()` method for each student.

   d. `public static void print_average_grade(String studentID)` that correctly prints a student's average grade by student ID
   e. `public static void print_invalid_emails()` that verifies student e-mail addresses and displays all invalid e-mail addresses to the user

      Note: A valid e-mail should include an at sign ("@") and period (".") and doesn't include a space.

C. Demonstrate the program's required functionality by running the following scenario:

```
print_all();
print_invalid_emails();
//loop through the ArrayList and for each element:
print_average_grade(current_loop_student);
remove("3");
remove("3");
//expected: this should print a message saying such a student with this ID was
not found.
```

1. Submit evidence of the program's required functionality by attaching the output along with the entire (zipped) project folder.

Note: The program may be created in any IDE, but BlueJ or NetBeans are recommended for faster evaluation.

D. Outside sources for the purpose of referencing are not required to complete this task, but if sources are used, include all in-text citations and references in APA format.

Note: For definitions of terms commonly used in the rubric, see the Rubric Terms web link included in the Evaluation Procedures section.

Note: When using sources to support ideas and elements in an assessment, the submission MUST include APA formatted in-text citations with a corresponding reference list for any direct quotes or paraphrasing. It is not necessary to list sources that were consulted if they have not been quoted or paraphrased in the text of the assessment.

Note: No more than a combined total of 30% of a submission can be directly quoted or closely paraphrased from outside sources, even if cited correctly. For tips on using APA style, please refer to the APA Handout web link included in the APA Guidelines section.