RIT | Golisano College of Computing and Information Sciences
School of Information

Name: _____

ISTE-120
Lab 09: Arrays and ArrayLists

Exercise 1 – GPA calculation using arrays (3 points)

The exercise must be completed during the lab period.

This lab requires the development of a java program to calculate the standard grade point average (GPA) for a student at RIT. The program will prompt the user to enter the credits and letter grade for each of **exactly** 4 courses.

The GPA is calculated by dividing the sum of points times credits by the sum of credits. To compute the sum of points, the letter grade must be converted to its equivalent number of points as follows: A is 4 points, B is 3 points, C is 2 points, D is 1 point, and F is 0 points. Then multiply the points by the number of credits and sum for the 4 courses. For example, if a student scores C in a 3-credit course, B in a 4-credit course, D in a 4-credit course, and A in a 3-credit course, the calculation would go as follows:

Sum of credits = 3 + 4 + 4 + 3 = 14
Sum of points times credits = 3 * 2 (=C) + 4 * 3 (=B) + 4 * 1 (=D) + 3 * 4 (=A) = 34
GPA = 34 / 14 = 2.42857... = 2.43 (rounded to 2 decimal places).

After the user enters the credits and letter grade for all four courses, print the sum of the credits for all four courses, the sum of the points times credits for all four courses and the GPA for all four courses.

For Exercise 1, write all of the code in the main method. Store the credits for each course in an array with 4 elements. Store the letter grades in another array with 4 elements. When computing the sum of the credits, an enhanced for loop **must** be used. This exercise uses "parallel" arrays; i.e. a data element in one array is matched to a related data element in the other array using the same index value. A constant **must** be defined and used for the number of courses.

Also, in Exercise 1, write a method:

```
public static int letterToNumeric(char letterGrade)
```

which, given a letter grade, returns the appropriate numeric grade. It must be static to be called from the main program.

In Exercise 2, develop a GPA **class** using "parallel" arrays to hold the grades and credits. In Exercise 3, develop a GPA class using an **ArrayList** to hold the course information.

Notes:

- Assume that the credits entered by the user will be a valid number between 0 and 9, inclusively
- Assume that the letter grade entered by the user will be a single character with value A, B, C, D, or F (upper or lower case)
- Read in the letter grade as a type String. Assume that the user enters only one letter
- Attempt to print the GPA to two decimal places using `printf`
- If the GPA cannot be calculated, set the GPA to zero

Sample Output

Command Prompt

```
dkpvcs> java GPA
Enter credits for course 1: 3
Enter grade for course 1: C
Enter credits for course 2: 4
Enter grade for course 2: B
Enter credits for course 3: 4
Enter grade for course 3: D
Enter credits for course 4: 3
Enter grade for course 4: A

Total number of credits: 14
Total number of points: 34
GPA: 2.43

dkpvcs>
```

Command Prompt

```
dkpvcs> java GPA
Enter credits for course 1: 4
Enter grade for course 1: f
Enter credits for course 2: 3
Enter grade for course 2: F
Enter credits for course 3: 2
Enter grade for course 3: f
Enter credits for course 4: 1
Enter grade for course 4: F

Total number of credits: 10
Total number of points: 0
GPA: 0.00

dkpvcs>
```

Fill in the two arrays to show their contents for the data in the above execution.

credits:

[0]	[1]	[2]	[3]

grades:

[0]	[1]	[2]	[3]

Signature: _____ Date: _____
Have your instructor or TA sign here when Exercise 1 works correctly.

Exercise 2 – GPA calculation using a class with arrays (4 points)

The exercise must be completed during the lab period.

Develop a class named **GPA** with the attributes and methods as described. Download the test class TestGPA.java (in Lab09Starter.zip). This class must not be modified. The code from Exercise 1 can be used as a starting point for this exercise.

Attribute	Description
credits	Array of ints to hold the credits for each course. The size is set in the constructor.
grades	Array of Strings to hold the letter grade for each course. The size is set in the constructor.
numCourses	Integer to hold the number of courses actually entered by the user.
maxCourses	Integer to hold the **maximum** of courses. Arrays declared to be of this size.

Method	Parameters/Return value	Description
Constructor	`_maxCourses` - Maximum number of courses. No return type.	Allocates space for the two arrays of the size of the parameter. Sets the `maxCourses` to `_maxCourses`. Sets the `numCourses` to zero
`addCourse`	`_credits` - number of credits for the course. `_grade` - course grade as a String (one letter only). No return value as it is assumed there is room in the arrays for the course to be added.	Stores `_credits` in the first available position in the credits array. Stores `_grade` in the same position in the grades array. Increment `numCourses` to indicate the user has entered one more course.
`calcGPA`	No parameters. Returns the GPA as type double.	Computes and returns the standard GPA based on A=4 points; B=3; C=2; D=1 and F=0. Return 0.0 if the GPA cannot be calculated.

There are more methods that could be added such as `getCredits`, `getGrade`, `calcSumCredits`, etc. For simplicity only the methods above are required.

Notes:

- Assume that the test program will not add more courses than the arrays can hold
- Assume that the credits and letter grade for each course are valid

Sample Output

Command Prompt

```
dkpvcs> java TestGPA
Enter number of courses: 2
Enter credits for course 1: 4
Enter grade for course 1: A
Enter credits for course 2: 3
Enter grade for course 2: b

GPA is 3.57

dkpvcs>
```

Sample #1

Command Prompt

```
dkpvcs> java TestGPA
Enter number of courses: 4
Enter credits for course 1: 4
Enter grade for course 1: B
Enter credits for course 2: 3
Enter grade for course 2: C
Enter credits for course 3: 3
Enter grade for course 3: D
Enter credits for course 4: 2
Enter grade for course 4: F

GPA is 1.75

dkpvcs>
```

Sample #2

Command Prompt

```
dkpvcs> java TestGPA
Enter number of courses: 2
Enter credits for course 1: 0
Enter grade for course 1: A
Enter credits for course 2: 0
Enter grade for course 2: B

GPA is 0.00

dkpvcs>
```

Sample #3

Signature: _____ Date: _____
Have your instructor or TA sign here when Exercise 2 works correctly.

1 BONUS POINT:
Modify the program to validate the number of credits as

```
0 <= numCredits <= 9
```

Validate the letter grade to be only: `A, B, C, D,` and `F,` upper or lower case.

Should the data validation be done in the test class provided by your instructor or the `GPA` class? You will need to modify the test class to complete this bonus part.

Exercise 3 – GPA calculation using a class with ArrayList (3 points)

If you do not complete this exercise during the lab period, you need to complete the work outside of the lab period and bring the completed work to the lab next week.

You are to modify the **GPA** class to use an `ArrayList` to store the course information. Since an `ArrayList` holds a collection of objects, you **must** create a new class named **Course** with the following attributes and methods.

Attribute	Description
`credits`	An `integer` to hold the credits for a course.
`grade`	A `String` to hold the letter grade for a course.

Method	Parameters/Return value	Description
Constructor	Credits and letter grade for the course.	Store the credits and letter grade in the new object.
`getGrade`	No parameters. Returns value of letter grade	Gets the letter grade from an object.
`getCredits`	No parameters. Returns value of credits.	Gets the credits from an object.

Sample Execution

NOTE: Even though the number of courses is NOT needed for the `ArrayList`, keep the constructor from Exercise 2 so your new code will work with the supplied `TestGPA.java`. The output will be the same as in Exercise 2 as no new functionality is being added.

Sample Execution

The output will be the same as in Exercise 2 as no new functionality is being added.

Complete the ArrayList diagram to show its contents for **sample #2** execution in Exercise 2.

(ArrayList)

0	1	2	3

Signature: _____ Date: _____

Have your instructor or TA sign here when Exercise 3 works correctly.