Name: _____

ISTE-120
Lab 08: Loops

Overview:

This lab involves the development of a program to compute the Grade Point Average (GPA) of a student based on the grade and credits for each of student's courses. For each course, the needed information is the final letter grade for the course and the credits for the course. A letter grade can be converted to "points" for the GPA calculation using the following table:

Multiplying the credits for one course by the letter grade points gives the "weighted points." For example, a grade of B in a 4-credit course yields 12 weighted points. The formula to compute the GPA for all courses is as follows:

Grade	Points
A or a	4
B or b	3
C or c	2
D or d	1
F or f	0

(sum of weighted points) ÷ (sum of credits)

Your problem is to develop the class **GPA** that will compute a student's GPA, and a test class **ComputeGPA** to prompt for the student's grade and credits for each course. The main method will then print the GPA after all course data has been entered.

Based on the description above, what are the attributes of the class **GPA**?

Based on the description above, what are the methods of the class **GPA**?

Exercise 1 - Testing Conversion of Letter Grade to Points (3 points)
The exercise must be completed during the lab period.

Write a class `GPA` that has **no** attributes and the following method:

Method	Parameters/Return	Purpose
calcPoints	Letter grade, type char as parameter. Returns the points for the grade	Converts the grade to its equivalent point value.

A switch statement **must** be used in `calcPoints` to convert a grade to its equivalent number of points. Return a -1 if the letter grade is not valid.

Write a class named `ComputeGPA` with the `main` method. Prompt the user to enter a letter grade. Use the method `calcPoints` to compute the number of points. An infinite loop **must** be used so that the user can enter all of the valid grades, both upper and lower case, and a selection of invalid grades to thoroughly test the method `calcPoints`.

Notes:
- The return value of -1 will be used in other exercises to validate the letter grade entered by the user; at this point, simply print the -1
- There is no method in the Scanner class to read a character. Read a string (using `next` or `nextToken`) and then select the first character as the grade using the `charAt` method (see the Java API for the `String` class)
- Assume that the user will enter at least one character for each letter grade
- The infinite loop is easy to write using a while loop; i.e. use `while (true)`. This allows for testing a large number of inputs without having to re-execute the program. jGRASP provides a button to halt the execution. If you run the program from a Terminal shell, instead of jGRASP, use control-Z followed by ENTER (or control-C)

Sample Output

```
Command Prompt

dkpvcs> java ComputeGPA
Enter grade (one character): a
Points: 4
Enter grade (one character): b
Points: 3
Enter grade (one character): c
Points: 2
Enter grade (one character): d
Points: 1
Enter grade (one character): e
Points: -1
Enter grade (one character): f
Points: 0
Enter grade (one character): A
Points: 4
Enter grade (one character): B
Points: 3
Enter grade (one character): C
Points: 2
Enter grade (one character): D
Points: 1
Enter grade (one character): E
Points: -1
Enter grade (one character): F
Points: 0
Enter grade (one character): X
Points: -1
Enter grade (one character):
dkpvcs>
```

Note: In order to thoroughly test the method `calcPoints`, the testing must include entry of all valid upper case and lower case letters and some invalid letters.

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

Exercise 2 - Computing GPA (4 points)
The exercise must be completed during the lab period.

Based on the formula above, the class `GPA` will need to keep track of the sum of weighted points and sum of the credits (we need not keep track of all of the grades). Add the following two attributes to the class `GPA`:

Attribute	Meaning
`sumCredits`	Sum of credits for all courses entered by the user of type `int`.
`sumWeightedPoints`	Sum of the weighted points for all courses entered by the user of type `int`.

Also, add an **accessor** for each of the two new attributes. Then, add the following methods to the class `GPA`:

Method	Parameters/Return	Purpose
Constructor	No parameters. No return type.	Set both sums to 0.
`addToTotals`	2 parameters. Param1 is the letter grade as type `char`. Param2 is the number of credits as type `int`. No return value.	Add the credits to the sum of credits. Convert the letter grade to points via `calcPoints`. Then, add the product of the points and credits to the sum of weighted points.
`calcGPA`	No parameters. Returns the GPA as type `double`.	Compute the GPA by dividing the sum of weighted points by the sum of credits. Beware of integer division.

To test the class GPA, add more code to the class ComputeGPA. After prompting for the letter grade, prompt the user for the number of credits.

Add a loop to allow the user to enter the letter grade and credits for multiple courses. There are many loops that could be used. <u>For this exercise, assume that each student takes exactly 3 courses.</u>

After prompting for the information for the three courses, print the GPA. See if you can format the output so that only two fractional digits are printed (see `printf` in `PrintWriter`, again, or use the `String.format`). See first Sample Output below, which correctly computes the GPA for the three courses.

The limited amount of output makes it difficult to determine if the sum of the points and sum of the weighted points have been calculated correctly. Modify the two classes so that the sum of the credits and the sum of the weighted points are printed after each course is entered. See

the two additional outputs below. With the extra information, it is then possible to determine if both sums are computed correctly.

Notes:
- Assume the user enters a valid letter grade
- Assume the user enters a non-negative number of credits, that is, 0 or greater
- Provide access to the two sums by two accessor methods
- The main method no longer directly calls the method `calcPoints`, which now should be called by the method `addToTotals`

Sample Output

```
Command Prompt

dkpvcs> java ComputeGPA
Enter grade (one character): a
Enter credits: 4
Sum Points: 16  Sum Credits: 4
Enter grade (one character): b
Enter credits: 4
Sum Points: 28  Sum Credits: 8
Enter grade (one character): C
Enter credits: 2
Sum Points: 32  Sum Credits: 10
GPA: 3.20
dkpvcs> _
```

```
Command Prompt

dkpvcs> java ComputeGPA
Enter grade (one character): D
Enter credits: 4
Sum Points: 4  Sum Credits: 4
Enter grade (one character): F
Enter credits: 3
Sum Points: 4  Sum Credits: 7
Enter grade (one character): a
Enter credits: 4
Sum Points: 20  Sum Credits: 11
GPA: 1.82
dkpvcs>
```

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

Exercise 3 – Data Validation and User-entered Number of Courses (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.

Now that the GPA is being calculated correctly, validate the user input and allow the user to enter the number of courses to be averaged.

A programmer needs to assume that any user input can be entered incorrectly. In the previous exercises, the user knew how to enter valid input. Now each input must be validated.

1. For the number of courses, the user must enter a number greater than 0. If the number of courses is invalid, print an appropriate error message and prompt the user to re-enter the number of credits

2. For the letter grades, the user must enter exactly one letter and the grade must be one of the valid letter grades (A, B, C, D, F, upper or lower case). The validation is partly completed as the method `calcPoints` returns a value of -1 if the letter grade is invalid

3. To check that the user enters exactly one letter, after reading the string that is the letter grade, check that it's length is exactly 1. For example, assuming `in` is a `Scanner`:
   ```
   String grade = in.next();
   if(grade.length() != 1) { … }
   ```
   If the input is invalid, print an appropriate error message and prompt the user to re-enter the letter grade

4. For the number of credits, the user must enter a number for the credits between 0 and 9, inclusively. If the number of credits is invalid, print an appropriate error message and prompt the user to re-enter the number of credits

Sample Execution

```
Command Prompt

dkpvcs> java ComputeGPA
Enter number of courses: 0
Invalid number of courses - must be greater than 0
Enter number of courses: -2
Invalid number of courses - must be greater than 0
Enter number of courses: 2
Enter grade (one character): a
Enter credits: 4
Enter grade (one character): B
Enter credits: 4
GPA: 3.50

dkpvcs>
```

```
Command Prompt

dkpvcs> java ComputeGPA
Enter number of courses: 2
Enter grade (one character): x
Invalid grade - must enter A,B,C,D,F (upper or lower case)
Enter grade (one character): xx
Invalid grade - must enter exactly one letter
Enter grade (one character): xxx
Invalid grade - must enter exactly one letter
Enter grade (one character): a
Enter credits: 4
Enter grade (one character): b
Enter credits: 99
Invalid credits - must be between 0 and 9, inclusively
Enter credits: -1
Invalid credits - must be between 0 and 9, inclusively
Enter credits: 10
Invalid credits - must be between 0 and 9, inclusively
Enter credits: 4
GPA: 3.50

dkpvcs> _
```

```
C:\. Command Prompt

dkpvcs> java ComputeGPA
Enter number of courses: 2
Enter grade (one character): a
Enter credits: 0
Enter grade (one character): b
Enter credits: 0
GPA: 0.00

dkpvcs> _
```

```
C:\. Command Prompt

dkpvcs> java ComputeGPA
Enter number of courses: 4
Enter grade (one character): a
Enter credits: 4
Enter grade (one character): a
Enter credits: 4
Enter grade (one character): b
Enter credits: 4
Enter grade (one character): c
Enter credits: 4
GPA: 3.25

dkpvcs> _
```

Signature: _____ Date: _____
Have your instructor or TA sign here when Exercise 3 works correctly.