Concordia University
COMP 248 – Fall 2019
Assignment 3

Due Date:	By 11:59pm November 1, 2019
Evaluation:	5% of final mark (see marking rubric at the end of handout)
Late Submission:	none accepted
Purpose:	The purpose of this assignment is to have you manipulate, 1 and 2-dimensional arrays of strings and to design and use a simple class.
CEAB/CIPS Attributes:	Design/Problem analysis/Communication Skills

General Guidelines When Writing Programs:

- Please refer to the handout of Assignment #1.

Question 1 (Single/multi Dimensional Arrays)

In this question you are to generate random passwords which will be composed of words from a few selected pages from "Alice in Wonderland" by Lewis Carroll, as available at: http://www.gutenberg.org/cache/epub/19033/pg19033.txt. The selected text is provided as a multidimensional String array – the first level (dimension) contains the pages, the second level contains the paragraphs in each page, and the third level contains the lines in each paragraph.

You can copy and paste the following array declaration into your own program

```
// http://www.gutenberg.org/cache/epub/19033/pg19033.txt
final String[][][] book = {
        {
                {"ALICE was beginning to get very tired of sitting by her sister on the\n",
                "bank, and of having nothing to do. Once or twice she had peeped into the\n",
                "book her sister was reading, but it had no pictures or conversations in\n",
                "it, \"and what is the use of a book,\" thought Alice, \"without pictures or\n",
                "conversations?\"\n"},

                {"So she was considering in her OWN mind (as well as she could, for the\n",
                "day made her feel very sleepy and stupid), whether the pleasure of\n",
                "making a daisy-chain would be worth the trouble of getting up and\n",
                "picking the daisies, when suddenly a White Rabbit with pink eyes ran\n",
                "close by her.\n"},

                {"There was nothing so very remarkable in that, nor did Alice think it so\n",
                "very much out of the way to hear the Rabbit say to itself, \"Oh dear! Oh\n",
                "dear! I shall be too late!\" But when the Rabbit actually took a watch\n",
                "out of its waistcoat-pocket and looked at it and then hurried on, Alice\n",
                "started to her feet, for it flashed across her mind that she had never\n",
                "before seen a rabbit with either a waistcoat-pocket, or a watch to take\n",
                "out of it, and, burning with curiosity, she ran across the field after\n",
                "it and was just in time to see it pop down a large rabbit-hole, under\n",
                "the hedge. In another moment, down went Alice after it!"}
        },
        {

                {"\"WHAT a curious feeling!\" said Alice. \"I must be shutting up like a\n",
                "telescope!\"\n"},

                {"And so it was indeed! She was now only ten inches high, and her face\n",
                "brightened up at the thought that she was now the right size for going\n",
                "through the little door into that lovely garden.\n"},

                {"After awhile, finding that nothing more happened, she decided on going\n",
                "into the garden at once; but, alas for poor Alice! When she got to the\n",
                "door, she found she had forgotten the little golden key, and when she\n",
                "went back to the table for it, she found she could not possibly reach\n",
                "it: she could see it quite plainly through the glass and she tried her\n",
                "best to climb up one of the legs of the table, but it was too slippery,\n",
                "and when she had tired herself out with trying, the poor little thing\n",
                "sat down and cried.\n"},

                {"\"Come, there's no use in crying like that!\" said Alice to herself rather\n",
```

```
            "sharply. \"I advise you to leave off this minute!\" She generally gave\n",
            "herself very good advice (though she very seldom followed it), and\n",
            "sometimes she scolded herself so severely as to bring tears into her\n",
            "eyes.\n"},

            {"Soon her eye fell on a little glass box that was lying under the table:\n",
            "she opened it and found in it a very small cake, on which the words \"EAT\n",
            "ME\" were beautifully marked in currants. \"Well, I'll eat it,\" said\n",
            "Alice, \"and if it makes me grow larger, I CAN reach the key; and if it\n",
            "makes me grow smaller, I can creep under the door: so either way I'll\n",
            "get into the garden, and I don't care which happens!\"\n"},

            {"She ate a little bit and said anxiously to herself, \"Which way? Which\n",
            "way?\" holding her hand on the top of her head to feel which way she was\n",
            "growing; and she was quite surprised to find that she remained the same\n",
            "size. So she set to work and very soon finished off the cake."}
},
{
            {"The King and Queen of Hearts were seated on their throne when they\n",
            "arrived, with a great crowd assembled about them--all sorts of little\n",
            "birds and beasts, as well as the whole pack of cards: the Knave was\n",
            "standing before them, in chains, with a soldier on each side to guard\n",
            "him; and near the King was the White Rabbit, with a trumpet in one hand\n",
            "and a scroll of parchment in the other. In the very middle of the court\n",
            "was a table, with a large DISH of tarts upon it. \"I wish they'd get the\n",
            "trial done,\" Alice thought, \"and hand 'round the refreshments!\"\n"},

            {"The judge, by the way, was the King and he wore his crown over his great\n",
            "wig. \"That's the jury-box,\" thought Alice; \"and those twelve creatures\n",
            "(some were animals and some were birds) I suppose they are the jurors.\"\n"},

            {"Just then the White Rabbit cried out \"Silence in the court!\"\n"},

            {"\"HERALD! read the accusation!\" said the King."}
}
};
```

The password generation process is as follows:

1. A password is composed of three words.
2. The page number, paragraph number, and line number are chosen randomly. You must use the java.util.Random class to generate random numbers using the nextInt() method. You can look it up in the Java API online.
3. Convert the String from the chosen line into an array of words using the split(" ") method (space as the separator).
4. Choose a random word from the array in Step 3. (**Hint**: the length attribute of an array will tell you how many words are in the array.)
5. If a word is composed of only a single character (e.g., "a"), or the word contains the newline character ('\n'), go back to Step 2.
6. When you have three words, concatenate them to form a candidate password, and apply the following steps. You can use the compareTo() method from the String class (but no HashSet). Restriction: Don't use any method from the Character class.
    a. No two words can be exactly the same (e.g., "theAlicethe" isn't allowed, but "the" and "The" are accepted). Otherwise, go back to Step 2.
    b. The password must be of at least 8 characters long but under 16 characters. Otherwise, go back to Step 2.
    c. The password must contain an uppercase character. Otherwise, go back to Step 2.
    d. The password must contain a lowercase character. Otherwise, go back to Step 2.
    e. The password must contain only one special (non-alphabetic) character. Otherwise, go back to Step 2. Restriction: Don't use any method from the Character class.

f. If all conditions are met, you print the password, along with the number of password generation attempts that failed due to: (i) having a newline character, (ii) having a single-character word, (iii) having the same words, (iv) not meeting the uppercase requirement, (v) not meeting the lowercase requirement, (vi) not meeting the special character requirement.

7. You continue generating passwords until you generate a total of 10,000 passwords, OR if you generate a password where the count for item (v) in Step 6 f) is non-zero.

Your program's output must be formatted as given below (you can use the Sting.format() method). Two examples to follow:

Sample output 1: When the 10,000 password limit is reached (truncated output, the highlighted line is explained below):

```
Welcome to the password generator game!

Password = Whiteonce;the     Newline = 3   Single = 0   Equal = 0   Length = 1   Upper = 4   Lower = 0   Special = 1
Password = intowereKing.     Newline = 9   Single = 2   Equal = 0   Length = 5   Upper = 17  Lower = 0   Special = 2
Password = Shekey;Just       Newline = 17  Single = 2   Equal = 4   Length = 10  Upper = 11  Lower = 0   Special = 4
Password = theKing.and       Newline = 4   Single = 3   Equal = 0   Length = 5   Upper = 8   Lower = 0   Special = 3
……
Password = upKing.leave      Newline = 5   Single = 0   Equal = 1   Length = 2   Upper = 5   Lower = 0   Special = 2
Password = so"WHATand        Newline = 8   Single = 3   Equal = 1   Length = 7   Upper = 4   Lower = 0   Special = 3
Password = "Whichinshe       Newline = 4   Single = 0   Equal = 0   Length = 1   Upper = 3   Lower = 0   Special = 0
Password = WhiteKing.seen    Newline = 0   Single = 0   Equal = 1   Length = 2   Upper = 1   Lower = 0   Special = 0
Password = totheKing.        Newline = 1   Single = 1   Equal = 0   Length = 2   Upper = 0   Lower = 0   Special = 1
Password = Justofwig.        Newline = 25  Single = 7   Equal = 1   Length = 17  Upper = 31  Lower = 0   Special = 11

Total passwords generated: 10000

Thanks for using the password generator game. Good bye.
```

Explanation of the highlighted line in the output above:
Before the valid password *Whiteonce;the* was generated, there were 3 passwords with a newline character generated; one whose length was not within the requested limits; 4 that did not have an upper case letter; and 1 that did not have a special character. Also 10,000 passwords were generated as there was never a password generated with no lower case letters in it.

Sample output 2: When the lowercase condition is reached (as mentioned in Step 6f; truncated output):

```
Welcome to the password generator game!

Password = King.wasand       Newline = 17  Single = 0   Equal = 1   Length = 8   Upper = 12  Lower = 0   Special = 3
Password = went"Ohcried      Newline = 5   Single = 0   Equal = 0   Length = 1   Upper = 4   Lower = 0   Special = 1
Password = must"WHATthe      Newline = 1   Single = 1   Equal = 1   Length = 1   Upper = 5   Lower = 0   Special = 0
Password = "OhSheas          Newline = 14  Single = 2   Equal = 0   Length = 2   Upper = 15  Lower = 0   Special = 4
Password = thedaisies,Once   Newline = 2   Single = 1   Equal = 0   Length = 2   Upper = 4   Lower = 0   Special = 0
Password = andButdon't       Newline = 1   Single = 0   Equal = 0   Length = 2   Upper = 1   Lower = 0   Special = 2
Password = byKing.and        Newline = 0   Single = 0   Equal = 0   Length = 0   Upper = 0   Lower = 0   Special = 0
Password = SoRabbitI'll      Newline = 1   Single = 0   Equal = 0   Length = 0   Upper = 0   Lower = 0   Special = 0
……..
Password = of"Silencethe     Newline = 0   Single = 0   Equal = 0   Length = 0   Upper = 3   Lower = 0   Special = 1
Password = outME"either      Newline = 1   Single = 1   Equal = 0   Length = 0   Upper = 2   Lower = 0   Special = 0
Password = wholebutKing.     Newline = 0   Single = 0   Equal = 0   Length = 0   Upper = 0   Lower = 0   Special = 0
Password = King.betwelve     Newline = 5   Single = 1   Equal = 0   Length = 3   Upper = 5   Lower = 0   Special = 1
Password = toandKing.        Newline = 2   Single = 0   Equal = 0   Length = 1   Upper = 1   Lower = 0   Special = 2
Password = throughME"in      Newline = 1   Single = 1   Equal = 0   Length = 0   Upper = 0   Lower = 0   Special = 0
Password = intheKing.        Newline = 4   Single = 1   Equal = 0   Length = 0   Upper = 2   Lower = 1   Special = 2

Total passwords generated: 4320

Thanks for using the password generator game. Good bye.
```

Question 2 (Simple Class)

1. Define a class named CPU for properties and methods related to an Intel CPU. The class should be defined as follows:
    a) Private instance variables to store the following: CPU generation (allowed values: 1 to 10), CPU series (allowed values: “i3”, “i5”, “i7”, “i9”), suggested price in USD (positive real numbers), CPU speed in GHz (positive real numbers), launch date of the CPU with quarter and year (e.g., “Q1’18”, “Q3’19”, note that a year has four quarters), and whether the CPU supports Intel Software Guard Extension (SGX).
    b) 2 constructors: No argument (sets generation to 1, series to “i3”, suggested price to 117 USD, CPU speed to 2.93 GHz, launch date to “Q1’10”, and SGX is not supported); and six argument constructor (sets variables as supplied).
    c) 6 Accessor methods to return the value of each variable.
    d) 1 Mutator method to set the suggested price.
    e) A public method called priceNow(String sQuarterYear) that returns the estimated price of the CPU based on the supplied sQuarterYear variable. In each quarter after launch, the suggested price is reduced by 2%. The returned price cannot be under 0.0USD. If the supplied quarter/year in sQuarterYear is before the launch date of the CPU, there is no depreciation and the suggested price is returned.
    f) A toString() method that returns all the values of a CPU object (see output below for formatting).
    g) An equals() method to test for equality of two CPU objects based on generation, series (ignore case), and SGX support.
    h) A boolean isHigherGeneration() method that compares two CPU objects based on their generations.
2. Write code for:
    a) Declare 2 CPU objects using both constructors, and output descriptions of the CPU objects.
    b) Test your accessor methods (any 2).
    c) Test the mutator method, the equals() and isHigherGeneration() methods.
    d) Calculate the current price for both objects.

3. No input validation is necessary. You may find String.split() and Integer.parseInt() methods useful.

Sample output:

```
Welcome to the simple class example!

CPU 1: This CPU is 1th generation i3 (2.93GHz), launched: Q1'10 with price: 117.00 USD. SGX is not supported.
CPU 2: This CPU is 10th generation i9 (3.1GHz), launched: Q2'19 with price: 449.00 USD. SGX is supported.
CPU 1 Series: i3
CPU 1 Suggested price: 117.00 USD
CPU 1 Suggested price (after mutator call): 110.00 USD
Are CPU 1 and CPU 2 equal? NO
Is CPU 1 of higher generation than CPU 2? NO
CPU 1 Price at Q3'19 :26.40 USD
CPU 2 Price at Q3'19 :440.02 USD

Thank you for testing the simple class example!
```

Submitting Assignment 3

- Zip the source code (the .java file only please) of this assignment.
- Naming convention for zip file: Create one zip file, containing the source files for your assignment. The zip file should be called *a#_studentID*, where # is the number of the assignment and *studentID* is your student ID number.
  For example, for the third assignment, student 123456 would submit a zip file named a3_123456.zip
- For sections EE, FF, P and R refer to your section's Moodle page for instructions on where to submit your assignment. For section EC, refer to your eConcordia webpage.

Evaluation Criteria for Assignment 3 (25 points)

Source Code	
Comments for all 2 questions (1 pt.)	
Description of the program (authors, date, purpose)	0.5 pt.
Description of variables and constants	0.5 pt.
Programming Style for all 2 questions (1 pt.)	
Use of significant names for identifiers	0.5 pt.
Indentation and readability	0.5 pt.
Question 1 (13 pts.)	
Generating random indices for word selection	2 pt.
Generating a password meeting all requirements	5.5 pt.
Display formatted outputs with statistics	3.5 pt.
Reaching the terminating conditions	2 pt.
Question 2 (10 pts.)	
Class definition, constructors, accessors and mutator	2 pt.
The priceNow() method	3 pt.
The toString() method	2 pt.
Other methods	1 pt.
Test code with output formatting	2 pt.
TOTAL	**25 pts.**