PROG2007 Assignment 2

Weight: 40% of your final mark

Due: Week 6 Monday (11/04/2022) at 11:00 PM

Specifications

Your task is to complete various exercises in BlueJ, using the Java language, and to submit these via the MySCU link created for this purpose.

Marking criteria includes:

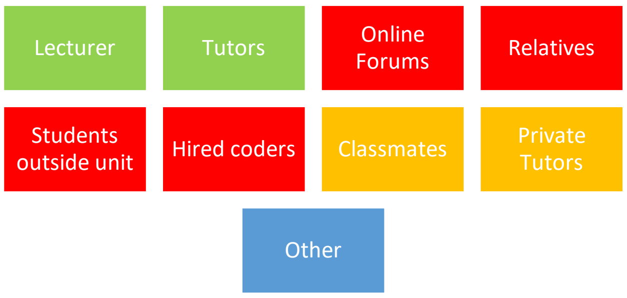
- Use of **correct coding style**, including the **use of comments**;
- **Accuracy** of coding;
- Use of **suitable coding structures**;
- **Correct submission** and **naming conventions** of assessment items as required.

Getting Help

This assignment, which is to be completed individually, is your chance to gain an understanding of the fundamental concepts of Java programming language on which later learning will be based. It is important that you master these concepts yourself.

Since you are mastering fundamental skills, you are permitted to work from the examples in the MySCU site or textbook, but you must acknowledge assistance from other textbooks or classmates. In particular, you must not use online material or help from others, as this would prevent you from mastering these concepts.

This diagram will help you understand where you can get help:



Encouraged

Attribution Required

Not acceptable

Ask tutor

Be aware that if you do get help from one of the red sources, you will be reported for academic misconduct, which may have serious penalties. Please visit the following link for the guidelines: https://bit.ly/scuAcadMisconduct

Setting up your assignment

To set up your assignment, you will need to do the following:

- Create a BlueJ project called **scuusername-A2-WordAndNumberGame**.

Task Description

Your task is to create a game where the users can play with words and numbers. The game has three consecutive levels (Level 1, Level 2, and Level 3). Users can proceed to the next level after successfully finishing the current level. At each level, the users will be asked to perform a task. Users will be notified about the outcome (success or failure) of the task. Upon a failure, the user can try the task one more time. Once the task is (successfully) completed, the user will be proceeded to the next level and notified. Two consecutive failures for a task (in a particular level) will **end** the game. A user will **win** if he/she finishes all four levels successfully.

Game class: You need to define a dedicated class (Game.java) to store the commonly used variables and methods (e.g., showing notification/instruction) that you will use in other parts of the program.

Use of inheritance and comments: You must apply inheritance in your program. You need to also add comments for each of your classes, methods, and key variables/fields.

Input/output: You can use standard Java I/O streams such as *System.in* and *System.out* to take input from the users and show output to the users. The input and output of your program should be continuous until the game finishes with a failure or win.

Level 1 - Anagram

What is an anagram: An anagram consists of creating a new word/phrase by rearranging the letters of a given word/phrase. Some examples of anagrams are given below.

elblow: below, night: thing, meteor: remote

It is possible that a single word/phrase will have multiple anagram words/phrases.

Program features: At the beginning of the level, your program will notify users about this level and the task the users need to complete. Following that, your program will display the letters of a **randomly** selected word for each play and attempt. The users will have to input all meaningful (not just combinations of letters) words resulting from rearranging the letters (anagrams). For example, "laets" is not a valid anagram of "tales". Therefore, the following cases would be considered as an invalid word input if the user enters:

- an invalid anagram (random combination of the word letters)
- a valid anagram that does not exist in the dictionary
- any valid or invalid word that does not consist of the given letters

The users complete this task if at least **two words** entered by the users are valid. Your program will display which of the inputted words are valid and not. Upon completion of the task, your program will notify the users by displaying "Congratulations! You completed level 1 successfully". If zero or only one inputted word is valid, this would be a failed attempt, and the program must display "Fail! at least two input words must be valid". Two consecutive failures in this level will end the game.

Users can input a sequence of words separated by spaces.

Anagram dictionary: You need to use the below anagram dictionary of 20 words. Using an appropriate collection, you must implement this dictionary in the Game class.

Anagram dictionary

Source word	**Anagrams**
late	tale, teal.
pate	peat, tape.
pare	pear, reap.
parse	asper, pares, pears, prase.
rate	tare, tear.
tales	stale, slate, tesla, steal, least.
east	eats, sate, seat, seta, teas.
alerts	alters, artels, estral, laster, ratels, salter, slater, staler.
post	pots, spot, stop, tops.
alerting	altering, integral, relating, triangle.
steak	skate, stake, takes.
live	evil, veil, vile.
meat	mate, team, tame.
insert	estrin, inerts, inters, niters, nitres, sinter, triens, trines.
laser	earls, reals, rales, lares.
lapse	peals, pleas, pales, sepal.
angle	angel, glean, genal.
super	purse, sprue.
scrape	capers, crapes, spacer, pacers.
retains	stainer, starnie, resiant, nastier, retinas.

This dictionary can be used in many parts of your program for different purposes, e.g., choosing the desired alphabets for the users and comparing if an anagram inputted by the user is valid.

Randomly selecting a word: The Random class (`java.util.Random`) can be used to generate a random number within a range (0 - 20). This randomly generated index can be used to select the word from the dictionary.

```
Select a random word:
    //Step 1. Create a new instance of the Random class;
    //Step 2. Generate an integer between 0 and 19 using nextInt() method
    //Step 3. Use the integer as index and select the word from the array
    //Step 4. Return the selected word as a string
End
```

Retrieving anagram list: Once you have identified the word, you can easily retrieve the corresponding anagrams using the `get()` method of the `dictionary` instance.

```
Get anagram lists (input is the word):
    //Step 1. Use the get() method to retrieve anagram lists from the dictionary
    //Step 2. Returns the anagram lists as a string array
End
```

Displaying letters of the selected word: You can use the `toCharArray()` method to convert a string into a list of characters.

Level 2 – Word Scrambler

What is a scrambled word: In the scambled version of a word, the letters are scrambled (i.e, organised in a different way). For example, below is the scrambled version of the word “student”.

dtsunet

The scrambled version of a word does not need to have any meaning.

Program features: Your program will notify users about this level and the task the users need to complete. Following that, your program will display a sequence of five (5) words, with the letters of each word scrambled. Your program must **randomly** select these words from the anagram dictionary stored in the “Game” class for each play and attempt. Your program then needs to **generate** scrambled versions of the selected words for the users. The users need to input the original version of each word as a sequence of words. The users complete this task if at least **three words** entered by the users are valid. Your program will display which of the inputted words are valid and not. Your program will notify the users upon completion of the task.

Users can input a sequence of words separated by spaces.

Generating scrambled words: There are many ways to scramble the letters in a word. One way is to swap the adjacent letters in pairs.

```
Generate scrambled word (input is the word):
    //Step 1. Convert the string into a char array
    //Step 2. Initiate an index variable, idx with 1 (not 0)
    //Step 3. Loop till the length of array
```

```
        //Step 4. Exchange the letters in position idx-1 and idx
        //Step 5. Increment idx by 2
    //Step 6. Convert the modified char array into string
    //Step 7. Return the string
End
```

Level 3 – Encoder

Your program will notify users about this level and the task the users need to complete. Your program will display a hashcode generated for a sequence of (**randomly** selected) **five** words. These selected words should be randomly generated for each play and attempt. Users need to decode the hashcode and input the original series of words. Your program will also display some sample inputs and outputs as a hint for the users. The anagram dictionary in the "Game" class would be useful to select the words. Your program will generate the hashcode for a series of words following the algorithm below. Please note that the sequence of words **does not need to be a meaningful sentence**.

1. Each alphabet would represent an integer, such as a, b, ……, y, z -> 0, 1, ……, 25. Only small letters will be considered, as the anagram dictionary has only small letter words.

For example, consider this sequence of words – "i saw a dog"

2. Your program will replace each letter in a word with the corresponding integer value. Your program will then insert one of these symbols: **!**, **@**, **#**, **$**, **%**, **&**, ***, after each interger value. The symbol has to be **randomly selected** each time.

For example, "i" would be replaced by "8*", and "saw" would be replaced with "18@0#22!", and so on.

3. Your program will concatenate all these strings with a comma as a separator.

For example, the generated string in this step would be "8*,18@0#22!,0%,3@14&6^" and your program will display the following samples:

Sample hashcode for user: 8*,18@0#22!,0%,3@14&6^

Sample original word sequence: i saw a dog

………………

Your program will display the hashcode and ask the user to decode that. The users will complete this task once they decode the hashcode and correctly retrieve the whole series of words. Your program will notify the users upon completion of the task or failure.

Users can input a sequence of words separated by spaces.

Randomly selecting multiple words: The Random class (`java.util.Random`) can be used to generate a random number within a range (0 - 20). This randomly generated index can be used to select the word from the dictionary.

```
Select multiple random words (input is the number of words required):
    //Step 1. Create a new instance of the Random class;
    //Step 2. Generate an integer between 0 and 19 using nextInt() method
    //Step 3. Initiate a string array
    //Step 4. Loop for the number of random words needed (5)
         //Step 5. Use the integer as an index and select the word from the array
         //Step 6. Add the selected word into the String array
    //Step 7. Return the string array
End
```

You can use a similar algorithm to randomly select symbols.

Generate hashcode: You need to follow the algorithm mentioned above to implement this using the Random (`java.util.Random`) class. Following is a possible solution.

```
Generate hashcode:
    //Step 1: Define variable to store the indices of the alphabets
    //Step 2: Define variable to store symbols
    //Step 3. Call the method for selecting multiple random words
    //Step 4. Store the selected words into a string array
    //Step 5. Initiate an empty string for hashcode
    //Step 6. Run loop for the number of random words
        //Step 7. Convert each random word into a char array, use toCharArray()
        //Step 8. Run loop for the length of char array
            //Step 9. Identify the index of the current character, use indexOf()
            //Step 10. Convert the index into a string and concatenate with hashcode
string
            //Step 11. Call method for selecting a random symbol and concatenate
with hashcode string
        //Step 12. Concatenate a comma with hashcode string

    //Step 13. Return the hashcode string
End
```

Submission

Please zip up your **scuusername-A2-WordAndNumberGame** project folder, which contains your source code, and upload it onto MySCU.