ECCS 1621 – Programming 2 – Spring Semester 2020
MP1 – "Let's play LaurieMOO!!!"
Due by end of lab on Tuesday 4 February 2020

For this assignment you are to write a Java application that will implement the (in)famous game of "LaurieMOO!" The game is simple: you have ten attempts to guess a four-digit number randomly generated by the game. For each digit correctly specified by position, you get a "`MOO!`" displayed by the program. For each digit specified that is not in the correct position but is in the solution, a "`moo.`" is displayed. You are ***NOT*** to indicate the location of the correct digits through the "`MOO!`" or "`moo.`" items displayed by your program. If the number is correctly guessed, then in addition to all four "`MOO!`" strings being displayed you also get a "`LaurieMOO!!!`" as a reward. If the user fails to guess the number after ten attempts, the message "`Boo hoo -- no LaurieMOO.`" is displayed within a message dialog along with the four-digit number (so that the user knows what the answer was).

Feel free to be creative with the graphical interface for this program; if adventurous, try to incorporate sound and/or graphics. At a minimum, there should be a text field for user input, and at least one label for displaying text output. Note that it is acceptable to build a string such as "MOO! moo. moo." and display that string on one label. Please note that the number generated by the program can be any four-digit number: it can have leading zeros, it can have multiple instances of the same digit, and so on. For example, the following secret values are all possible: 0000, 0123, 3455, 7870.

Please play Dr. Estell's implementation of LaurieMOO!!! to get a feel for the game; you do not have to go as overboard as he did in implementing your version. ☺

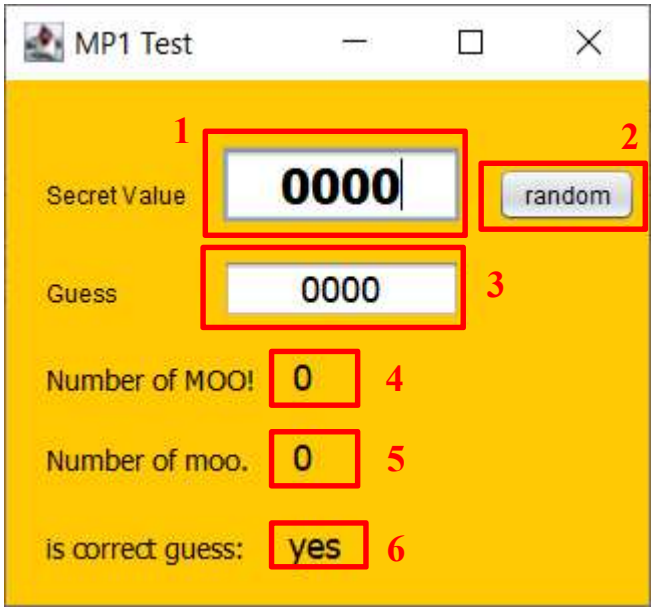
You are to use appropriate OOP-design principles for this assignment by creating the RandomMooValue class. An instantiation of this class will create a guessable 4-digit value as an instance variable. You must, at a minimum, implement the constructor and methods as indicated by the Javadoc for this class provided at the end of this assignment.

You are to submit your assignment via Dropbox – please use "LaurieMOO" as the name of your project/folder.

You are to break down the development of this MP into two parts.

Part A - This section is for verifying the correctness of your RandomMooValue class. You will be asked to demo this in lab on Tuesday, 28 January. Part A of the lab will walk you through the development of the GUI; please work on writing the methods for this class prior to lab.

Create an application named MP1Test that implements the "RandomMooValue" class and has the following GUI set-up (note that items in red are not part of the GUI):



Implement the following functionality for MP1Test:

- The 'Secret Value' text field [1] is used to both enter and display the secret 4-digit value to be guessed. If a number is entered into this text field, the method **`setSecretValue( int newValue )`** is to be invoked and the text in the 'Secret Value' text field is updated via a call to **`getSecretValue()`**.
- The 'Random' button [2] is used to generate a new 4-digit value to be guessed. When clicked, the method **`setSecretValue()`** is used to generate the new value to be guess and the text in the 'Secret Value' text field is updated via a call to **`getSecretValue()`**.
- The 'Guess' text field [3] is used to guess the answer value. When a value is entered into this text field, the resultant event must perform the following updates:
  - The 'Number of MOO!' result label [4] with a call to **`getBigMooCount( int guess )`**
  - The 'Number of moo.' result label [5] with a call to **`getLittleMooCount( int guess )`**
  - The 'is correct guess:' result label [6] with a call to **`isCorrectGuess( int guess )`**

Spend time **thoroughly** testing your code using this testing app, where you KNOW and can even SET the answer to be guessed, and then go on to **Part B**: writing the actual LaurieMoo! app, **where the answer is hidden and cannot be set** and the "MOO!" and "moo." items are printed out and NOT enumerated as shown above. The moos can be displayed as a single string for output, like "MOO! moo. moo.", or on separate labels like that shown in the example LaurieMOO! app.

Class RandomMooValue

Creates a 4-digit secret value (0000 through 9999) for a player to guess. Feedback is returned in the form of big and little moos. Each "MOO!" indicates a digit correctly guessed in both value and position. Each "moo." indicates a digit correctly guessed in terms of value, but not position. If no digits are correctly guessed, then all the user hears are cowbells... Please note that the number generated by the program can be any four-digit number: it can have leading zeros, it can have multiple instances of the same digit, and so on. For example, the following values are all possible: 0000, 0123, 3455, 7870. When generating big (MOO!) and little (moo.) moos, each guessed digit can only match at most one digit in the secret value. For example, if the secret value is 0055 and the user's guess is 5550, our favorite cow should be uttering "MOO! moo. moo." as there is an exact match in digit position 3, plus two inexact matches.

- ***Constructor Summary***

Constructor and Description
RandomMooValue() Creates a new RandomMooValue object containing a secret value to be guessed.

- ***Method Summary***

Modifier and Type	Method and Description
int	**getBigMooCount**(int guess) The number of digits in the user's guess that exactly (i.e., same position) matches digits in the secret value.
int	**getLittleMooCount**(int guess) The number of digits in the user's guess that match digits in the secret value, but are not at the same position.
int	**getSecretValue**() Access the value that the user is trying to guess.
boolean	**isCorrectGuess**(int guess) Determines if the user correctly guessed the secret value .
boolean	**setSecretValue**() Generates a new secret value to be guessed in a game of LaurieMOO.
boolean	**setSecretValue**(int n) Sets the "secret" value to be guessed in a game of LaurieMOO to a known 4-digit quantity.

- ***Constructor Detail***

- **RandomMooValue**

```
public RandomMooValue()
```

Creates a new RandomMooValue object containing a secret value to be guessed.

- ***Method Detail***

- **setSecretValue**

```
public boolean setSecretValue()
```

Generates a new secret value to be guessed in a game of LaurieMOO.

Returns:

`true in all cases.`

- **setSecretValue**

```
public boolean setSecretValue(int n)
```

Sets the "secret" value to be guessed in a game of LaurieMOO to a known 4-digit quantity. This method is for testing purposes only.

Parameters:

```
n - The number that will be set as the secret value, if it is within the inclusive
range of 0000-9999.
```

Returns:

```
true if the secret value was reset; false if the passed value was outside of the
allowed range of values.
```

- **getSecretValue**

```
public int getSecretValue()
```

Access the secret value that the user is trying to guess, primarily to show the user after running out of guesses.

Returns:

```
the secret value that the user is/was attempting to guess.
```

- **getBigMooCount**

```
public int getBigMooCount(int guess)
```

The number of digits in the user's guess that exactly (i.e., same position) matches digits in the secret value.

Parameters:

```
guess - The number that the user guessed.
```

Returns:

```
a number 0-4 representing how many digits the user guessed correctly by position.
```

- **getLittleMooCount**

```
public int getLittleMooCount(int guess)
```

The number of digits in the user's guess that match digits in the secret value, but are not at the same position.

Parameters:

```
guess - The number that the user guessed.
```

Returns:

```
a number 0-4 representing how many of the guessed digits match, but are in different
positions. Note that a guessed number cannot have any one digit generate both a "MOO!"
and a "moo." as a result.
```

- **isCorrectGuess**

```
public boolean isCorrectGuess(int guess)
```

Determines if the user correctly guessed the secret value.

Parameters:

```
guess - The number that the user guessed.
```

Returns:

```
true if the guess is correct, false otherwise.
```