**Reminder**

* [Asking Questions for Assignments](https://learn.uco.edu/d2l/common/dialogs/quickLink/quickLink.d2l?ou=376620&type=content&rcode=uco-3257182)
* Plagiarism (Academic Integrity paragraph) at [Syllabus](https://learn.uco.edu/d2l/common/dialogs/quickLink/quickLink.d2l?ou=376620&type=content&rcode=uco-3197635)

**Demo:**[**https://youtu.be/Gq2Owm\_1R5k**](https://youtu.be/Gq2Owm_1R5k)

**Create a VSCode project to play a word guess (a.k.a. hangman) game as follows:**

* project name: project1
* Create a Java window as shown in the demo program
  + At “North”: 2x1 GridLayout with 2 non-editable JTextField’s – one for the game key, the other for player’s guess
  + At “Center”: a canvas for drawing health level and game over message
    - Size: 500 x 500
    - Health level begins at 5
    - At each wrong guess, the health level decreases
    - The game ends when the health level is 0; displays game over message.
  + At “South”: 4x7 GridLayout with 27 JButtons (26 alphabets and 1 “New”)
  + JButtons are enabled/disabled automatically as the game progresses
  + **For other details not specified here, make your program like the demo.**
* As implementation begins, enable git and add remote repo at GitHub (**private** repo)
  + GitHub remote repo naming requirement: *<YourUCOEmail>*-se3103-project1
  + E.g., my UCO email is hsung@uco.edu, then the remote repo name should be: **hsung-se3103-project1**
* You are free to choose any names for Java Classes, variables, and methods. However, those names must be in compliance with the Java naming convention.
  + Use class name **Main** that has the main method as a starting point of execution.
  + **Java naming convention**: <https://www.oracle.com/java/technologies/javase/codeconventions-namingconventions.html#:~:text=Class%20names%20should%20be%20nouns,such%20as%20URL%20or%20HTML).>

**The Program Structural Requirements 1 (No credits may be given if violated)**

* **Java GUI** window and other components should be created and initialized as shown in Lessons videos using Java Swing (and some AWT). The use of JavaFX is not allowed.
* The use of **MVC architecture**: Separation codes among model, view, controller functions.
* **Use of "drawing canvas"** in the same way Lecture2 uses.
  + Use the canvas in the same way Lecture uses it especially the use of **paintComponent( )** method in **JPanel** class to implement the canvas.
  + Use of **JavaFX** or **java.awt.Canvas** class is not allowed.
* Use a **JButton** array (or ArrayList) of the length of 26 for the alphabet buttons (not including “New” JButton)
* **In any part of the program, DO NOT** write code for alphabet buttons by listing 26 different cases using *switch* or *if-else-if* statements. Use a loop or some other means instead of listing all possible 26 cases. For example, but is not limited to:
  + to assign labels of JButton array.
  + to find which JButton is pressed

**The Program Structural Requirements 2: The Game Key Generation:**

* Define an ArrayList<String> to hold the game keys (a word pool).
* Add the provided list of words below into the word pool ArrayList
  + Do not read the words from text file, but hard code in the program (e.g., wordPool.add("XYZ") for each word
* Generate a random number based on the size of the ArrayList, and use the random number as an index to get the game key from the word pool
  + Use **Random** class (from **java.util** package). Do not use Math.random( )
  + Generate a random integer from 0 to Arraylist's size - 1 to pick a word in that index as a game key
  + Study the API of the **Random** class to learn how to use it.

**Submission:**

* Download the source code from GitHub
* Submit the zip file to [Project1](https://learn.uco.edu/d2l/common/dialogs/quickLink/quickLink.d2l?ou=376620&type=dropbox&rcode=uco-2461101)
  + Late submission of up to 12 hours is accepted with an hourly late penalty (2 points per hour out of 100)
* Post the video demo link at the corresponding Student Video Presentation Forum (Project1 demo)
  + The same video demo requirements as Video 1 of Lesson Assignments.
    - The running program demo only. Do **not** show/explain the source code
  + Using the running app, show that all project requirements (except program structural requirements) are met
  + Max video length: 3 min

**Grading Policy:**

* **Zero** **credits** if
  + compile errors or program crashes at runtime
  + the program produces trivial or no meaningful display on canvas
  + the program does not use MVC architecture as shown in the lecture
  + the program does not use the drawing canvas as described above
  + the submitted program is not a download from GitHub
  + the video demo is not submitted
* Source program requirements: Partial credits if not met
  + Use of array (or ArrayList) for JButtons
  + Use of array (or ArrayList) for the word pool, use of the provided words
  + Use of Random class to pick a game key randomly
  + Keeping Java naming convention
* UI: Partial credits if not met
  + Window layout as specified above
  + Automatically enable/disable of JButtons
  + Game progress messages (player's guess status)

**Words list for the word pool (33 words)**

communication  
science  
programming  
language  
difficulty  
artificial  
intelligence  
attempts  
screenshot  
baseball  
windows  
learning  
electronics  
beautiful  
internet  
database  
organization  
application  
network  
friendly  
validation  
attempts  
statistics  
physics  
chemistry  
engineering  
school  
industry  
revolution  
progress  
characters  
heavily  
graphics