# Project 1 (Part D): Hangman Game

The goal of this project is to become more familiar with creating and using classes and objects in Java, to further introduce you to the concepts of abstraction and modularization, and to introduce you to collections of items. In this project you will build on your Hangman project. You will complete your Hangman so it contains everything we need to display and control your game.

# Instructions

1. You now have all of the pieces needed to complete the game, now your job is to make them work together.
	1. Coordinate the user input to appropriately change the alphabet panel, the person, and the guess phrase based on each letter pressed. You will need to modify your AlphabetPanel key listener code to turn incorrect letters a different color (like red) after they have been guessed. Similarly, correct letter guesses should also be turned a different color (like green) after being entered by the user.
		1. *Hint 1*: you may need to move the AlphabetPanel’s key listener into your HangmanGame class.
		2. *Hint 2*: you may need to make your HangmanGame a JFrame, and put the code that normally goes in main for a JPanel into Hangman’s constructor. Then the HangmanGame main simply becomes: **new** HangmanGame();
	2. Ask the user if they want to play again. If they do, reset the guess phrase to a new phrase and reset the rest of the game.
	3. NOTES:
		1. Clicking in the window should not advance the person (so delete the mouse event handling you previously had in person)
		2. Only incorrect letters should make a part of the person show up and **ONLY** the first time the incorrect letter is pressed (pressing the same incorrect letter over and over should not cause any more body parts to show).
2. Be sure to comment everything
3. Each of your methods (including their parameters).
4. Anything else that needs explanation.
5. Use proper indentation
6. Use appropriate variable/field names
7. Group like things together (items that are part of the same objects)
8. Use appropriate comments – including comments for each method
9. Check your curly braces
10. For this project you will submit the following (see “Saving and Submitting your Work” for details):
11. A general archive file (e.g. a zip)
12. A runnable JAR file
13. At least one JPG showing your application
14. EXTRA CREDIT: Add comments for correct/incorrect answers. NOTE: You can use two files for this and reuse your RandomString class from Project 5. Display a corresponding comment from the person being hung, negative if the user/player guessed wrong, positive or encouraging if the user/player got it right.

# Hints / tips

* For the comments positive and negative you can reuse your RandomString class from Project 5
* When incorporating multiple panels in a single frame, you will need to set a layout scheme, such as BoxLayout.
* As part of project 1 you need to make the logic of the game work properly as well as ask the user if they would like to play again. You may find it useful to use the following logic:
 **if** person or phrase is revealed
 process the user's Y/N answer
 **else**
 process the input as usual for the game, updating the person, guess phrase panel,
 & alphabet panel as appropriate
 **if** person or phrase is revealed
 ask the user if they would like to play again
* To create a runnable JAR file:
	+ Follow the saving and submitting instructions to create a runnable JAR and make sure you select the HangmanGame class as your launch configuration.
	+ Upload the runnable JAR file as an attachment to your project submission, so your project submission should have at least 3 files: an archive file, a runnable JAR file, and an image of your game (perhaps part way through a game).
	+ The JAR file is basically an executable that you will be able to double click to launch your program. Note: you will still need to have your "guess\_phrases.txt" file in the same directory as your JAR file. **You must name the file with your guess phrases: “guess\_phrases.txt”**

# Saving and Submitting Your Work

* An archive file (often called a zip file) can be created directly from Eclipse by going to File  Export and then selecting General  Archive.
* Upload as a secondary file a JPG or PNG picture/image of your Picture class when it is drawn. To do so run your program, then:
	+ For Windows: Select the window that has the drawing in it then press the ALT and Print Screen buttons at the same time. This puts a copy of the active window in memory. You can open Microsoft Paint or any image editor and paste the image in and save it as a JPG or PNG file in the directory of your project. Upload that saved file.
	+ For Mac: Press Command – Shift – 4, followed by the spacebar, and then select the desired application window. This puts an image file on your desktop, which you can then upload.
* For Java application projects you will also create a runnable JAR. To do this you will again select File  Export, and then select Java  Runnable JAR file.
* When submitting online be sure all files are uploaded. These files include: an archive file [e.g. zip], an image file [e.g. PNG], and for Java applications a runnable JAR. Once these files are uploaded to the Blackboard project submission page, press the “Submit” button. You may submit as many times as necessary, but your **last** submission is the only one that will be evaluated – so it must have **all** the necessary file attachments.