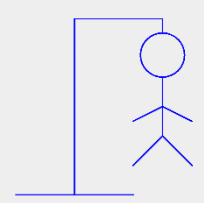
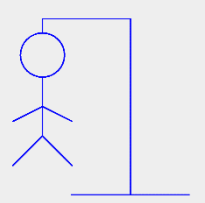


HANGMAN



Project by Riley Schnee

For this project, we will be programming Hangman! No longer will you need a piece of paper, or even a partner, to play this classic game. We have covered all the concepts you will need to use in previous labs. I have broken the project up into several smaller units to make it more manageable; you will be doing them over the next few classes. If you work on the project unit each day in class and make sure each unit is working before proceeding to the next, you will see the game coming together before your eyes! You will end up with a real game you can package into a standalone program anyone anywhere can play.

Before You Get Started

Necessary Files

There are only 4 starter files for this project and they are in our directory on the server:

`HangmanDisplay.java`
: This is the file that you will modify.

`HangmanPanel.java`
: This `JPanel` class contains `HangmanDisplay.java` and an exit button.

`HangmanDriver.java`
: This is the driver. This is the code you will run.

`words.txt`
: A sample list of words .txt file that is used for playing against the computer.

The majority of code for this project is contained in `HangmanDisplay.java`. Your first task is to add your last name to the file and class name. Your class name should look like `LastName_HangmanDisplay`. Then save the file with the new name so your source file name matches the internal class name as required by Java. Also, remember to update the name of the display source in `HangmanPanel.java`. The `HangmanDisplay` class includes:

Import statements for the various packages
Recall from class that a GUI application needs to access a lot of packages to function properly; this class imports them all.

Variables that appear in or control the game
These are variables for buttons, labels, Strings, arrays, booleans, etc. Your code should use these constants internally so that changing them in one location in your file changes the behavior of your program accordingly.

How to Succeed with this Project

Remember to focus and work on your project in class. Make sure that each stage covered by an activity is working properly before moving on to the next activity. *Do not try to get everything working all at once*. Leave time for learning things and asking questions. Finally, do not try to extend the program until you get all of the basic functionality working. If you add extensions too early, debugging may get very difficult.

Honor Code

You are expected to do your own work on this project. The only way to truly learn is to do the work on your own. The teacher will try to give you all the help you need to succeed at this project. Cheating includes, but is not limited to:

- giving or receiving of any work other than your own
- copying another person's work or allowing another to copy your work

You may not show or share your code electronically with another student. As with any programming assignment, it is highly unlikely that your code will look exactly like someone else's. The teacher will be checking for instances of plagiarism which includes code taken from sources outside of LVHS. Violations will receive a zero for the project. The project has been broken down into 5 manageable tasks so ask for help as needed along the way and have fun with it!

So…What is Hangman?

If you don't already know the rules to Hangman, here they are!

Objective

Guess the word/phase before the stick figure is hung!

How to Play

One player thinks of a word or phrase and draw a number of dashes equivalent to the number of letters in the word, leaving spaces where appropriate. The other player tries to guess what the word or phrase is, one letter at a time. If the guessing player suggests a letter that occurs in the word, the other player fills in the blanks with that letter in the right places. As the game progresses, a segment of the gallows and of a victim is added for every suggested letter not in the word. The number of incorrect guesses before the game ends is up to the players, but completing a character in a noose provides a minimum of six wrong answers until the game ends in the loss of the guessing player. However, if the guessing player figures out the word or phrase before the hangman is finished, they win!

The Project Version

For the version we will be creating, there will be the option to play with one player (playing against the computer) or with two (one chooses the word, the other guesses.) This will be the first thing the user selects. If they choose one player, the word will be randomly selected from the words.txt file. If they choose two player, a prompt will appear asking for player one to look away and player two to enter a word between 4 and 20 characters. From here, the guessing player will type in either a single letter or a full-word guess. If a full-word guess is correct, they will win the game, otherwise, a piece of the hangman will appear. If a single letter guess is correct, the dashes in the word will change to the letter where appropriate. If the single letter guess is incorrect, a piece of the hangman will appear. Either way, when a letter guess is made, the letter will appear in a "letters guessed" list. When the guessing player has either won or lost, a prompt will appear, asking if the user wants to play again. A "Yes" response will restart the game and a "No" response will quit the game.

Make sense?

…

Fantastic! Let's get started!

Activity 1: File Input and Initializing

Tour the Shell Code

Take some time to familiarize yourself with the Shell code. The better you understand the layout of the code now, the easier it will be for you to figure out where changes need to be made to accomplish what you need to do (requirements) and want to do (enhancements). For the most part, these activities will start at the top of the file and work downwards.

If you haven't yet, copy the files locally to your flash drive and add your initials to the class and file name for HangmanDisplay.java. Use the "Find" tab in jGRASP to help find both times "`HangmanDisplay`" is included in the file because you will need to change both.

Now tour the code and answer the following questions:

1. Are all labels, textboxes and buttons visible upon startup? How many are?
2. How many methods are there? Are you supposed to touch them?
3. What is the purpose of the `btnGuess`? What does its listener do?
4. Review `listenerOnePlayer`, `listenerTwoPlayer`, and `listenerEnter2PlayerWord`. Make sure you understand how these work.

File Input for words.txt

In order for the `listenerOnePlayer` to pick a random word, the `arrWords` array must be filled with the words from `words.txt`. Use `infile`, which is initially set to `null`. This way, you can be scanning from the file to fill in `arrWords`. Be sure to utilize try-catch for this, so that the game will quit if the file cannot be found. The first line of `words.txt` has an `int`, which is the number of words in the file. You will want to read this in using `infile` and set it equal to a new `int` variable to set the length of `arrWords`.

Once you have done that, initialize `arrWords` using your new variable for its length. Then fill the array with the contents of the `txt` file. If you would like, you may sort the array using the method in the `Arrays` class.

Give the Main Panel `null` Layout

This part is easy. Just set the layout of the panel to `null`.

Initialize Strings and Word Arrays

Refer to the code comment in the Shell. You will want to use for-loops for the arrays.

You should now run `HangmanDriver.java` to make sure the game starts! If it doesn't, something may be wrong with your code. Also check button functionality, especially "# of Players" and entering in a word for 2 Player games. These parts should both work at this point.

Understanding Dimensions

Below is a box that represents the frame created in `HangmanDriver.java`.

What are the dimensions? _____ x _____

This is the size of `HangmanPanel`.

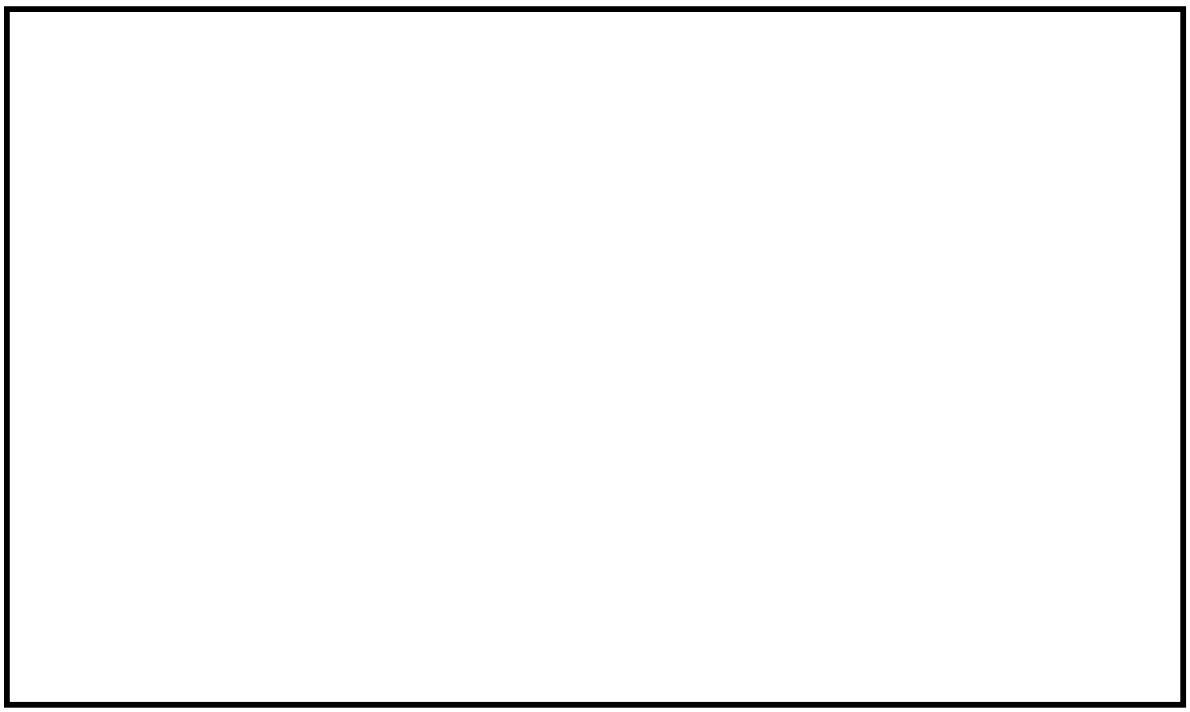
Label the box with these dimensions outside the perimeter.

Now look at `HangmanPanel.java`.
It contains another panel, `HangmanDisplay`.

Bounds are set using the parameters (x, y, width, height). Remembering that (0,0) is the top left corner, make a point on the box where the `HangmanDisplay` begins.

What is x + width? _____
What is y + height? _____

Now, draw a box within the box that represents the `HangmanDisplay` within `HangmanPanel`.
(It doesn't have to be perfect!)



GREAT! Now you understand how `HangmanDisplay` relates to `HangmanPanel`. Even though we use sometimes use 0 as a starting x-coordinate in `HangmanDisplay`, nothing is on the edge of the frame because we have some buffer room thanks to the `HangmanPanel`!

Activity 2: Processing Guessing

This listener is the brunt of the functionality of Hangman, processing a guess and doing something in response to whether it is or is not in the goal word.

Checking the Guess

When a player makes a guess, they click on `btnGuess`, which calls `ListenerGuess`. We are going to write the majority of this. Some is already given. Make note of the booleans at the top of `ListenerGuess`; found is used for single-letter guesses and `correctGuess` is for full-word guesses. This listener will be sure something has been typed in and that the guess is not in the `guessedLetters` String. It will then determine if the guess is a word or a letter.

- If it is a word, it will see if it matches the goal word and, in that case it does, will make the `correctGuess` boolean true.
- Should the guess be a single letter, the guess will be added to `guessedLetters`. Then it will iterate through `goalWordArr` to see if there are any matches. If there are, found will become true and that index of `guessWordArr` will now contain the guessed letter.

Using the Booleans

Staying within `ListenerGuess`, the next part checks the booleans. `if(found)` is completed for you, so do not touch it. In the `else if(correctGuess)`, update `lblCurGuess` to display the goal word, change the win boolean to `true`, and call `repaint()` so the win message will now appear. In the case that `!found && !correctGuess`, do as the code comments instruct and increment `numIncorrectGuesses`. Then update the guess booleans (`g1`, `g2`, `g3`, etc.) based on `numIncorrectGuesses`. You will need to call a `repaint()` after this. Finally, reset the contents of the textbox `txtGuess` so it is empty. Make sure to do this outside of the if-else structure.

Test and DEBUG

At this point, we have enough coded to test the game. Remember that body parts for the hangman will not appear and the “Play Again?” buttons at the end will have no functionality, so you will need to close out of the game. Check not only what you have completed in this activity, but also what was done in Activity 1. You may want to insert some `System.out.println(*whatever*)` in spots where you want to check values. Just remember to take them out when you have finished creating the game!

Activity 3: Time to Doodle!

Next you need to create the graphics for the hangman that will appear throughout the game. Go to the bottom of the Shell to `PaintComponent(Graphics g)`. Remove lines "`/* <- BEFORE DOING ACTIVITY FOUR, TAKE OUT THIS COMMENT START`" and "`BEFORE DOING ACTIVITY FOUR, TAKE OUT THIS COMMENT END -> */`" around `PaintComponent`. This section was commented out so that you could check your functionality on Activities 1 and 2.

Drawing the Hangman

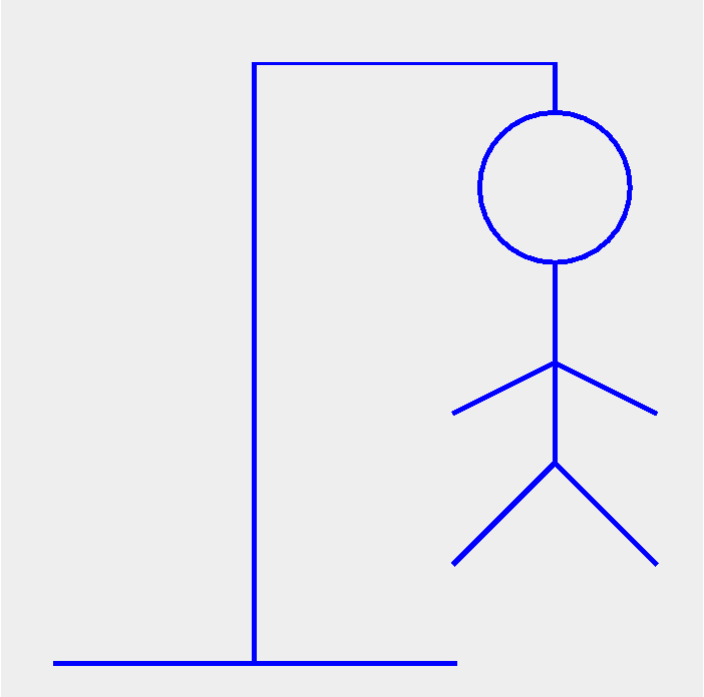
I have already given you the gallows. The if-statements add onto the stick figure when the booleans within them turn true. So, for example, when there is 1 incorrect guess, the `ListenerGuess` makes `g1` true. When we call `repaint()` anywhere in the code, it executes `PaintComponent`. If the player has made 6 incorrect guesses, all of the booleans from `g1` to `g6` will be true and all of those body parts will show up. Look at the gallows code for reference, so that your stick figure will look attached to the "downward notch."

You may want to refer to the Graphics class API.

Test That You Can Hang a Man

You will want to test that the body parts come up when you get certain numbers of guesses wrong and that they show up where you want them to.

For reference, here is how your hangman could look upon making 6 incorrect guesses:



Activity 4: To Replay or Not to Replay

When the game is finished, a prompt will appear that asks the user if they want to play again. Here, we will write the listeners for the "Yes" and "No" buttons that appear as part of the prompt.

`ListenerYesReplay`

If a player wants to replay the game, upon selecting "Yes," all variables of the game that may have changed need to go back to their initial value. Refer to the top of `HangmanDisplay.java` and what we did in Activity 1 regarding the arrays and Strings to help you.

`ListenerNoReplay`

This part is quite simple compared to `ListenerYesReplay`. Should the user not want to replay your amazingly fun game, the game will simply quit. If you do not know how to quit the program using code, refer to `ListenerExit` in `HangmanPanel.java`.

Test Basic Hangman!

You have completed basic Hangman! Test your Activity 4 code by playing a few games in a row against the computer, then pretend you are two people and play against yourself.

Activity 5: Time for Your "A" Game

Enhancing the Game

If you have worked on each activity the day it was assigned, then you are ready to start enhancing the game. The first section has the enhancements you must add as part of the project. But you are welcome and encouraged to extend the game and try to make it more fun. In doing this, you might find yourself reorganizing `run`. That is okay. However, we highly suggest that you **save a copy of the basic game in a separate folder before you start to make major changes**. That way you have something to revert to if you really screw up the program.

Required Enhancements

The following 3 features of the game must be completed for you to get full credit on this assignment.

Keep score

Design some way to score the player performance. You may want to have a scoreboard for "Player 1 vs Player 2" or "Player vs Computer," depending on their One Player/Two Player selection.

Add more Hangman parts

Want to keep the guesses coming? For harder or longer words, 6 incorrect guesses may be too few before game over. Want to add eyes? A mouth? Shoes? Add up to four more parts.

Switching who chooses the word

When a user is playing Two Player, if more than one game is played, have the person who enters the word switch back and forth. You will need to address the `lblEnterWord` and, if you have done the "Keep score" enhancement, the scoring. The best way to do this would probably be to simply keep track of the "guesser."

Bonus Enhancements

There are many fun ways to extend this assignment. Here are some ways to enhance the game that I have thought of. I have rated the difficulty of them based on either the amount or difficulty of the code required or the amount of change it will require to the code base. If you do an extension beyond Color Enhancement, it is required that you make a copy of your files first, in case you cannot complete the extension by the time the project is due. Either way, the original will need to be turned in to be sure you understand to complete the project as outlined.

Color Enhancement (easy)

Change the background and color scheme to be your own theme.

Word Categories (medium)

Wouldn't it be nice to choose a category you want your word to be from when playing against the computer? For this enhancement, you may need to create new .txt files in Notepad to input. Be sure to brainstorm how you would like to implement this before beginning.

Change guess input to buttons (hard)

While the provided Shell heavily supports a textbox-oriented guessing system, you may try changing this to a button system. Try having an array of letter `JButtons` that, when clicked, are disabled so the user knows they have guessed that letter.

Use your imagination

What else have you always wanted to do with Hangman? At some point your game might stop being like *Hangman* and be more like *Wheel of Fortune*.

Create a Jar File (Must be included with files you turn in)

How cool would it be to package up the game you just wrote so that anyone can play it without having to have a Java IDE installed? That is exactly what a jar file will do for you. The `jar` file extension indicates a Java Archive which collects all the files needed by the application into one place. Creating a `jar` file, `LastName_Hangman.jar`, allows you it to be run as a stand-alone application (e.g. by double-clicking on its icon). *(Note: be sure to set the title of the window to have "by Your Name" in it.)*

To create a jar file with jGRASP, follow this procedure:

1. Select **Project**->**New** from the menu; navigate in the file tree to the directory where your source code and extra files are; enter a **Project Name**; press **Next**.
2. Check the **Add Files to Project Now** box; press **Next**.
3. Navigate in the file tree to the directory where your project source is; in the **Files of Type** dropdown menu select **All Files**; select all files that contain "Hangman" and have a .class file extension (these are for the listeners) and press **Add**; select your .txt file(s) and press **Add**; press **Done**.
4. Select **Project**->**Create JAR or Zip File For Project** from the menu; click the **JAR File (Executable or Library)** radio button at the top; press **Next**; make sure the dropdown at the top has **HangmanDriver** selected; press Create JAR.

Now you can send your jar file to your friends!

Completing the Assignment

Before submitting anything, test your program to see that it works. Play for a while and make sure that as many parts of it as you can check are working.

Give Yourself Credit

Your program needs to have a header that contains the following:

- Your name
- An estimate of the time you spent on the project **outside of class**
- A brief description of the enhancements you implemented. This should include a description of any additional instructions needed to play the game.
- Your review of this assignment: what you liked, what would have helped you more.

As always, you should:

- Make sure that your code is properly indented (use the CSD button in jGRASP)
- Comment the code you write!

Turning it In

Take all the files related to the project and put together in a folder. Do not include `.class` files and other stuff not used in your application. Name this folder `LastName_Hangman`. Be sure to include your `.jar` file and any extra files you used in your project (e.g. custom `.txt`, sound or art files). Zip the folder. It should have the name `LastName_Hangman.zip`. Then, submit the zip to the teacher's Inbox.