In order for someone who has never before seen my game to play it, they should follow these simple steps:

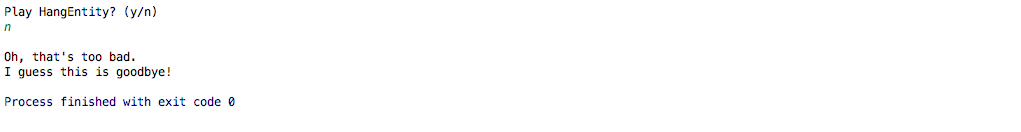
1. Before anything, one should download a Java IDE that will enable users to actually run the program. The one that we use at school is [NetBeans](https://netbeans.org/), and that is available online with a quick Google. Any IDE will work, including [IntelliJ](https://www.jetbrains.com/idea/download/) and [Eclipse](http://www.eclipse.org/downloads/packages/eclipse-ide-java-developers/keplersr1).
2. Unzip the file “Muir, Thomass – ICS3U – HangEntity.zip” Within it, one should be able to find two documents (“Muir, Thomass – HangEntity– User Documentation”; “Muir, Thomass – HangEntity– Work Log”). In order to get to the User Documentation, one can easily access this document “Muir, Thomass – HangEntity– User Documentation”). Along with the two documents is a compressed zip file of the code itself.
3. In order to get to the actual java file, one should open up their IDE, decompress the “HangEntity” file, and open up their java source file (“HangEntity.java”).
4. In order to get the game running, one should first save the file to a specified Java directory before compiling the java file. This is dependent on the IDE, but most have a button at the top that will quickly compile the file; in DrJava, a button at the top clearly labelled “Compile” will do the trick.
5. After compiling, run the file. Again, this button is dependent on the IDE; DrJava will have a button in the upper right corner.
6. To play the game, click either “Play” or “Rules” that will appear in the JFrame that pops up when one presses Run, and henceforth follow the prompts that will be displayed in the Event Log.
7. The rules of HangEntity are simple! When pressing “Rules”, this is what the program will print:

|  |  |
| --- | --- |
| Here are the rules:  This game is a classic version of hangman.  If you accept to play, a word will be randomly selected from a premade list.  And your game will begin!  You will have 5 chances to guess the word.  You may only guess character by character.  If you get a character wrong, you will lose a 'limb.'  (Don't worry, it's not graphic.)  If you lose all your limbs, you will die.  Kthxbye, let's just play.  Play HangEntity? (y/n)   |  | | --- | |  | |

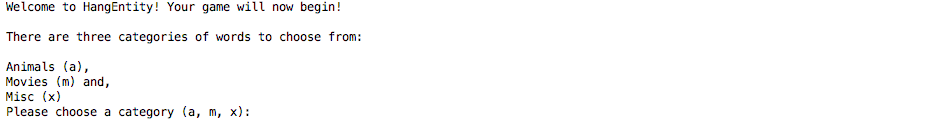
1. If anything other than ‘y’ or ‘n’ is entered, they are redirected back to the initial JFrame:



1. The user is given the option to enter yes or no (case ignored). If someone inputs ‘n’, the program will be terminated.



1. If ‘y’ is entered, the Option() method is brought up. (also what will happen if a player originally chooses “Play Game” in the original JFrame. The user is given three options of categories to choose from.



1. If the user enters anything other than ‘a’, ‘m’, or ‘x’, the following code is displayed and they are redirected back to the beginning of the Option() method:



1. When any category is chosen, the game will officially begin and the Game() method is brought up. The user automatically has all their 5 limbs (don’t question it). They are prompted to enter a letter or a word:



1. When the user enters a letter, the program will go through various loops and if statements to determine if the letter is correct, if it is not, if the user has lost all their limbs, or if they won the game. If they guess incorrectly, they lose a limb and this is displayed:



1. If they guess correctly, then an underscore is replaced and they do not lose a limb. The following code is displayed:



1. If they guess incorrectly too many times and they die, then the following code is displayed:



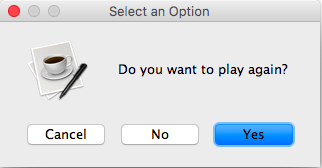
1. If they guess all the letters correctly and they solve the word, then the following code is displayed:



1. If they choose to guess a word at once and get it correct, then the same code is displayed:



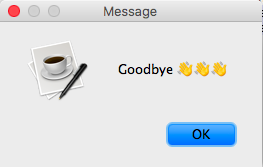
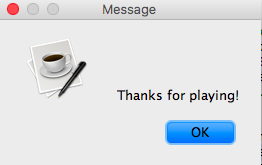
1. In either case of winning or losing, the PlayAgain() method is called up, in which a JOptionPane dialogue box is displayed. The dialogue box looks like this:



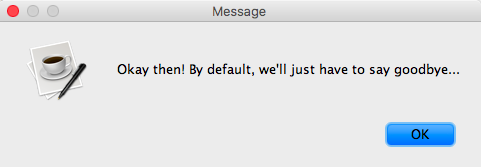
1. If the user clicks yes, then the amount of limbs is reset to 5 and the Options() method is called back up. The game restarts:

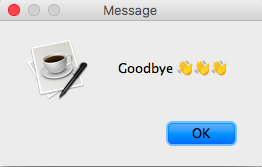


1. If the user clicks no then the following dialogue boxes appear, signaling the end of the game:



1. If the user clicks cancel then the following dialogue boxes appear, signaling the end of the game:





1. And voila! Players are finished the game ☺

**PROBLEMS AND SOLUTIONS**

* Detailed in the rightmost column of the work journal, day by day.

In general:

|  |  |
| --- | --- |
| **PROBLEM** | **SOLUTION** |
| JFrame doesn’t appear automatically. | * Always set JFrame to setvisible(True) * And put it right after the method declaration * Or else It might not show?? * Second button not originally working   + "Rules” button * Forgot to implement addActionListener button   + And then I accidentally named it in regards to the first button and forgot to add button2 |
| Painfully discovered that **.contains** can really only be used for Strings. Nothing was working for a while, my life was crumbling all around me ☹ | * Just kidding, figured out fairly quick that that only pertained to Strings. * Ultimately changed the way that I changed the characters to underscores. Instead of using System.out.print (which would would have been more helpful if my words were strings and not characters), I just set all the guessArray characters to \_ * Took me an absurd amount of time to realize how to fix the lack of spaces between hidden characters. Just had to print the guessArray and add a " " to represent the space. |
| Had issues with the **if / else if statements**, in which the user would still lose limbs even if they entered a correct character. | * Added in Boolean variable correctLetter, which is automatically set to False * When the user guesses a correct character, the variable is set to True and the loop resets |
| Didn't understand how to properly implement this in a short and concise way | * Researched ArrayLists * Implemented ArrayList ‘guessedBefore’ |