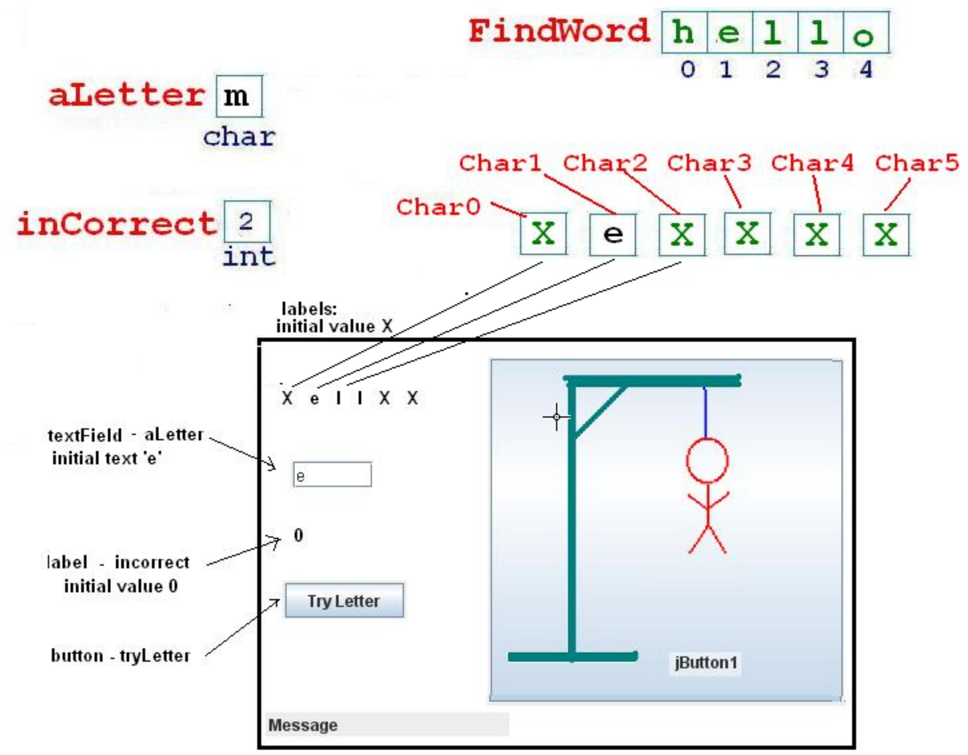
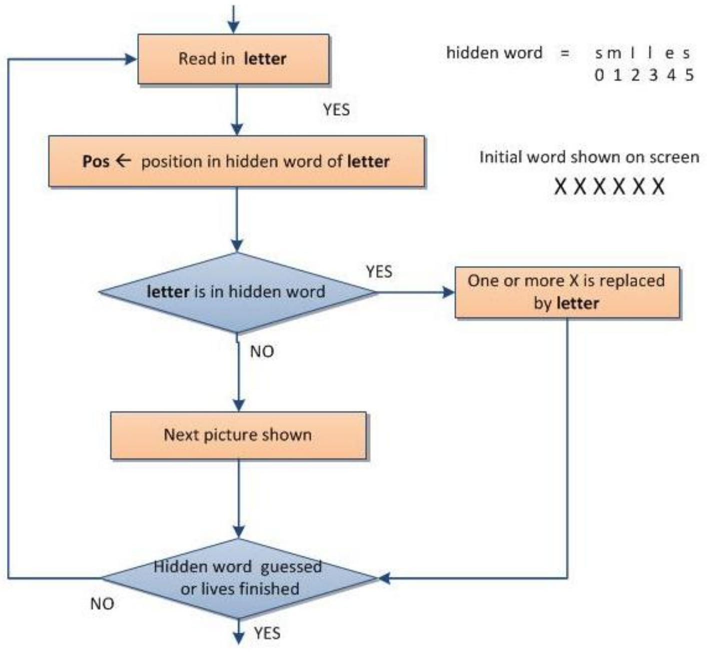
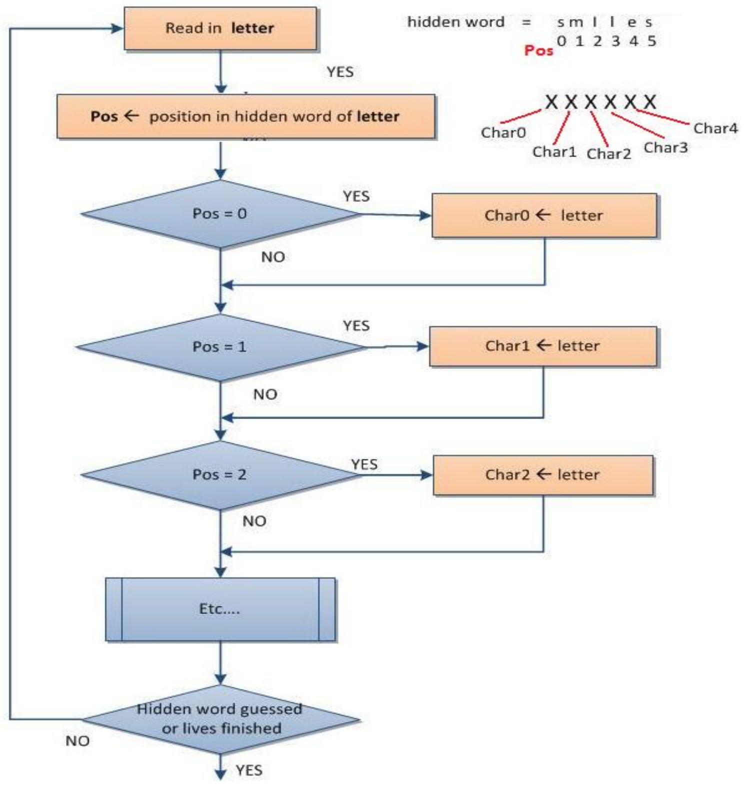
Guess the Word - Start Guide.

Note: *there is an opportunity to be creative and use any sequence of images as an alternative to the traditional 'hang man' pictures shown.*

- For 5 minutes play 'guess the word / hang man' with the person next to you to understand the general principles of the game. We should be able to work out a general algorithm to specify how the game should work (with any word).
- Your tutor will show the *guessWord* video (Blackboard).
- Then see the *guessDesign* video which shows design using UML activity diagrams. Stop *guessDesign* at 9 mins 25 seconds and create an activity diagram to show how the correct picture image is chosen and shown. Then watch the rest of *guessDesign*.
- Then watch *guessWord2* video to see how to implement this in Java Windows.




Read in letter
hidden word = s m i l e s
0 1 2 3 4 5
YES
Pos ← position in hidden word of letter
Initial word shown on screen
X X X X X X
letter is in hidden word
YES
One or more X is replaced by letter
NO
Next picture shown
Hidden word guessed or lives finished
NO
YES



File - New Project - [Java Application] - [Next] – (don't create main!)
Give it the name: *GuessWord* then click [Finish]

File - New File - (make sure the project shows your project name e.g., GuessWord)
Choose: *Swing GUI Forms - JFrame Form* - [Next] – call it **GuessForm** then click [Finish]

Look at the 'Title' property of the JFrame (see right of screen).
Give it the title: **Guess the Word Game**

- Drag 6 Labels on to the screen and name them Char0, Char1, Char2 etc.
- Arrange them in order left to right on the screen – Char0, Char1, Char2 etc.
- Give each label the initial text as the character **X**

These will show the letter characters of the word that the user will have to guess. They all start with an X but if the user guesses a correct letter then the X will be changed to the letter they have guessed correctly. E.g.,

X X X X X X initially for 'hello' , but
X e X X o X if the user has guessed e and o correctly .

- Add a textfield – call it **aLetter** and put the text character '**e**' as the initial text .
- Add a label – call it **inCorrect** and put '**0**' as the initial text (this will count incorrect guesses).
- Add a label called **MyMessage** and give it the text 'look here' (this is for our messages).
- Add a button. Rename it as **tryLetter.** The button text should say 'Try Letter'.

Double click on the button to see the code.
On the line after **public class GuessForm extends javax.swing.JFrame {** type this in:

```
String FindWord = "hello";
```

This is a global setting outside your frame code which all the program functions can access. The end user will have to guess this word!

Double click on the button on the frame again to add some java code behind the button.

```
int charPos = 2;
String letter = aLetter.getText();
if (charPos == 1) Char1.setText(letter);
if (charPos == 2) Char2.setText(letter);
```

add 4 extra lines to this code for charPos 0, 3, 4.

- **Run the program**, and you will notice that when the button is clicked the Char2 position shows 'e'
- Stop the program. Change charPos to be 1 and run the program. Click the button and Char1 will show the 'e'
- Run the program and change the 'e' in the textfield **aLetter** to be an 'h' and click the button.
- Try charpos=4 and enter an 'o' into the aLetter.

In general this code will place any letter from **aLetter** into the character position of charPos.

We want to show the letter on the screen only when it appears in the word to be guessed, and we want to show the letter in the correct position in the word to guess e.g., X e X X o X

The method **indexOf( )** can be used to find the position of a specified character in a string. The word we are guessing is in the string FindWord.

```
charPos = FindWord.indexOf(letter);
MyMessage.setText("position is " + charPos);
```

The **IndexOf( )** method can find the position (index) of the letter within the string but will return minus 1 (-1) if the letter is not found.

Look carefully and work out where to add these extra 2 lines of code to the button code. Run the program and try various letters such as 'e' and 'h' and 'o' and see if it shows them in the correct character position for the word to guess 'hello'.

Incorrect guesses

If the letter in **aLetter** is not in the guess word then the count of incorrect guesses should be increased by 1. Add code to the end of the button code to increase the label **inCorrect** by 1 when a wrong letter is guessed.

Checking for second occurrence of a letter

The **aLetter** could appear more than once in the guess word.
We need to ask a second time if the letter appears in the guess word, and we need to find the second occurrence.
If charPos has the position of the first occurrence then we can look again starting in the text string at a position one more than the first occurrence.

```
FindWord.indexOf(letter, <new start position>)
```

Copy and paste most of the button code so it appears twice. The code should find the first occurrence and display the letter found, and if it finds a letter then it should go on to look for a second occurrence of the same letter.
Code to look for the second occurrence: `FindWord.indexOf(letter, charPos+1)`

Guess the Word image display

Use a **JButton** to display the images of the hang man (or alternative image).
if incorrect = 1 then display the first image.
if incorrect = 2 then display the second image.

Additional work

See the Guess Word specification (on Blackboard). The specification lists details for additional work - extras 1, 2 and 3.