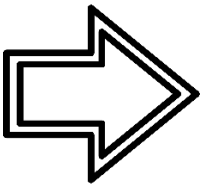
Assignment

Overview

For this assignment we will be recreating the 20 Questions game. This is a game where someone "thinks" of something and another person asks a bunch of questions to try and guess the thing someone thought of. To help you complete this you will construct a basic decision tree on paper and translate it to the program that you will write. *NOTE: you are not necessarily having 20 questions built into your game per say, unless you have fun with this and make it that way.*



Directions

To Get Started:

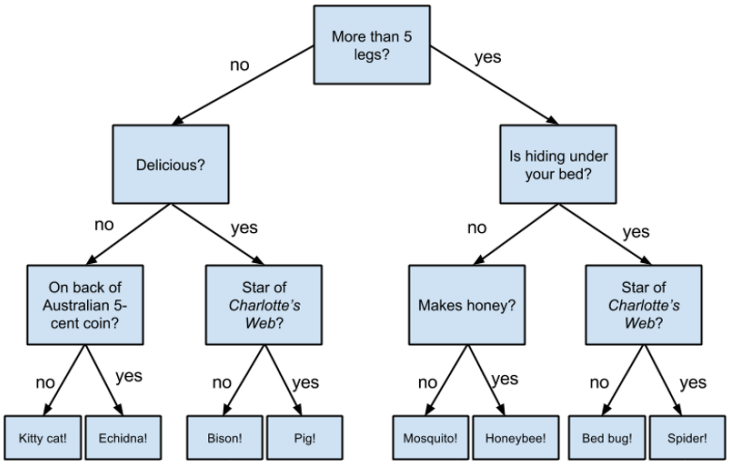
1. Create a new project in IntelliJ(or your preferred IDE) called YOUR_MCC_Name-Assignment2
2. Then create a new java class file named **GuessingGame.java**
    - This will contain your main method and all the code for the assignment

Part 1 - Planning out your game

To plan out the game you should first construct a decision tree on paper or via draw.io

A decision tree shows the simple flow of possible next steps/consequences/chance outcomes based on what decision is made. It is a good way to map out an algorithm that contains only conditional control statements

The following is an example of a decision tree with yes/no questions.



For this you need to pick a category or group of things (even at random) for the computer to guess. The computer should then ask questions for the user to answer. These questions should be a combination of yes/no, text based and number based. YOU HAVE TO INCLUDE AT LEAST 2 OF EACH.

These following guidelines should be followed, when all these are met the rest of the questions can be of whatever type you prefer.

- The decision tree should be **at least** 4 questions deep (for reference, the image above is 3 questions deep)
- 2 String values **other than simple yes/no checks**
- 2 int and/or double values
  - remember double values to check equality (==) should use a range
    - double num = 2;
    - if (num > 1.9999 && num < 2.0001) { *** code here ****}
- 1 comparison of at least && or || logical operators
- Lastly it should contain at least two questions that have **more than two outcomes**. For example if you ask "How many legs does it have" consider 0, 2, 4, > 4 as possible branches. The decision tree doesn't have to be binary/just two responses

Draw the decision tree out on a piece of paper OR use draw.io (from our INFO 1003 class) to draw out the decision tree. Take a picture or save a copy of the decision tree. Insert it into your project folder. **This doesn't have to be in the src folder, just in the main project folder.**

Part 2 - Coding the game

Now that you have the decision tree complete you should use it to guide for writing a series of nested if statements that complete the game.

Your game should:

- Prompt the user with a message to "Think of a ___, press enter when ready " using a print statement.
- Then make the program pause before asking the first question.
  - To do this we can simple call the scanner .next(); method that is basically asking for input for which the user will just hit enter and continue the program. This call should not save to any variable just a simple call. (So for example if I had a Scanner object "input" simple type the code input.next();)

```
Think of an animal or insect. I'll try to guess it!  Press enter when
ready.
```

- Next use a Scanner object ask questions that follow the logic of your decision tree. You'll need to use many nested if/if-else/ and if-else-if statements to do this
  - Along the way make sure that String comparisons are NOT case sensitive when checking
  - Here is an example of what this would look like if we only asked yes/no questions.

```
More than 5 legs? (yes/no): no
Delicious? (yes/no): YES
Star of Charlotte's Web? (yes/no): yes
```

- Once your program comes to a conclusion about the user's object, it should print out the guess and prompt the user for confirmation if the guess was correct.

```
I guess that you are thinking of a pig!
```

- Based on the user's response to the question here, the game should either celebrate (if it guessed correctly) or lament (if incorrectly). Display an appropriate message to the user.

```
woo hoo!! I got it right  \~~/
```

Submission

1. Compress the IntelliJ project folder and submit it to this assignment.

   to compress: On Windows, right click -> send to -> compressed .zip file

   on Mac, right-click -> Compress

Hints/Tips (Before Submitting):

- *Don't forget to have a header file at the top of your file and include a **Resource statement**.*
- *Use comments throughout for full points.*
- *Follow all Java Styling Guides as covered*
- *Please make note of the guidelines for the types of questions/input you have to get*
- *Note that String inputs should be checked regardless of case sensitivity (equalsIgnoreCase())*
- *Be creative and have fun with the assignment*
- *Test the program several times with different inputs to make sure the program runs as intended.*