**Question**

WAP in any high level language to implement the animal game.

**Background**

The animal guessing game is a program that finds the animals which is thinking by asking of question. We asked a different questions that are pretty close to the animal so that next time the game is played the animal is guessed correctly. This game is implemented using binary tree.

**Methodology**

The program is coded in java and linked list data structure was used to implement the binary tree i.e node. The node consist of the head node which is the question and two leaf nodes which may consists of further question if they are non-leaf node or animal name is case they are leaf node.

**Program**

**import** java.util.Scanner;  
  
**public class** labOne  
{  
 **private static** Scanner *stdin* = **new** Scanner(System.***in***);  
  
  
 **public static void** main(String[ ] args)  
 {  
 BTNode<String> root;  
  
 *instruct*( );  
 root = *beginningTree*( );  
 **do** *play*(root);  
 **while** (*query*(**"Shall we play again?"**));  
  
 System.***out***.println(**"Thanks for teaching me "**);  
 }  
  
  
  
 **public static void** instruct( )  
 {  
 System.***out***.println(**"Please think of an animal."**);  
 System.***out***.println(**"I will ask some yes/no questions to try to figure"**);  
 System.***out***.println(**"out Then what you are."**);  
 }  
  
  
  
 **public static void** play(BTNode<String> current)  
 {  
 **while** (!current.isLeaf( ))  
 {  
 **if** (*query*(current.getData( )))  
 current = current.getLeft( );  
 **else** current = current.getRight( );  
 }  
  
 System.***out***.print(**"My guess is "** + current.getData( ) + **". "**);  
 **if** (!*query*(**"Am I right?"**))  
 *learn*(current);  
 **else** System.***out***.println(**"I knew it all along!"**);  
 }  
  
  
  
 **public static** BTNode<String> beginningTree( )  
 {  
 BTNode<String> root;  
 BTNode<String> child;  
  
 **final** String ROOT\_QUESTION = **"Are you a mammal?"**;  
 **final** String LEFT\_QUESTION = **"Are you bigger than a cat?"**;  
 **final** String RIGHT\_QUESTION = **"Do you live underwater?"**;  
 **final** String ANIMAL1 = **"Dog"**;  
 **final** String ANIMAL2 = **"Ant"**;  
 **final** String ANIMAL3 = **"FISH"**;  
 **final** String ANIMAL4 = **"RAM"**;  
  
 *// Create the root node with the question ?Are you a mammal??* root = **new** BTNode<String>(ROOT\_QUESTION, **null**, **null**);  
  
 *// Create and attach the left subtree.* child = **new** BTNode<String>(LEFT\_QUESTION, **null**, **null**);  
 child.setLeft(**new** BTNode<String>(ANIMAL1, **null**, **null**));  
 child.setRight(**new** BTNode<String>(ANIMAL2, **null**, **null**));  
 root.setLeft(child);  
  
 *// Create and attach the right subtree.* child = **new** BTNode<String>(RIGHT\_QUESTION, **null**, **null**);  
 child.setLeft(**new** BTNode<String>(ANIMAL3, **null**, **null**));  
 child.setRight(**new** BTNode<String>(ANIMAL4, **null**, **null**));  
 root.setRight(child);  
  
 **return** root;  
 }  
  
  
 **public static void** learn(BTNode<String> current)  
 *// Precondition: current is a reference to a leaf in a taxonomy tree. This  
 // leaf contains a wrong guess that was just made.  
 // Postcondition: Information has been elicited from the user, and the tree  
 // has been improved.* {  
 String guessAnimal; *// The animal that was just guessed* String correctAnimal; *// The animal that the user was thinking of* String newQuestion; *// A question to distinguish the two animals  
  
 // Set Strings for the guessed animal, correct animal and a new question.* guessAnimal = current.getData( );  
 System.***out***.println(**"I give up. What are you? "**);  
 correctAnimal = *stdin*.nextLine( );  
 System.***out***.println(**"Please type a yes/no question that will distinguish a"**);  
 System.***out***.println(correctAnimal + **" from a "** + guessAnimal + **"."**);  
 newQuestion = *stdin*.nextLine( );  
  
 *// Put the new question in the current node, and add two new children.* current.setData(newQuestion);  
 System.***out***.println(**"As a "** + correctAnimal + **", "** + newQuestion);  
 **if** (*query*(**"Please answer"**))  
 {  
 current.setLeft(**new** BTNode<String>(correctAnimal, **null**, **null**));  
 current.setRight(**new** BTNode<String>(guessAnimal, **null**, **null**));  
 }  
 **else** {  
 current.setLeft(**new** BTNode<String>(guessAnimal, **null**, **null**));  
 current.setRight(**new** BTNode<String>(correctAnimal, **null**, **null**));  
 }  
 }  
  
 **public static boolean** query(String prompt)  
 {  
 String answer;  
  
 System.***out***.print(prompt + **" [Y or N]: "**);  
 answer = *stdin*.nextLine( ).toUpperCase( );  
 **while** (!answer.startsWith(**"Y"**) && !answer.startsWith(**"N"**))  
 {  
 System.***out***.print(**"Invalid response. Please type Y or N: "**);  
 answer = *stdin*.nextLine( ).toUpperCase( );  
 }  
  
 **return** answer.startsWith(**"Y"**);  
 }  
  
  
  
  
 **public static class** BTNode<E>  
 {  
 *// Invariant of the BTNode<E> class:  
 // 1. Each node has one reference to an E Object, stored in the instance  
 // variable data.  
 // 2. The instance variables left and right are references to the node's  
 // left and right children.* **private** E **data**;  
 **private** BTNode<E> **left**, **right**;  
  
  
 **public** BTNode(E initialData, BTNode<E> initialLeft, BTNode<E> initialRight)  
 {  
 **data** = initialData;  
 **left** = initialLeft;  
 **right** = initialRight;  
 }  
  
  
  
 **public** E getData( )  
 {  
 **return data**;  
 }  
  
  
 **public** BTNode<E> getLeft( )  
 {  
 **return left**;  
 }  
  
  
  
 **public** E getLeftmostData( )  
 {  
 **if** (**left** == **null**)  
 **return data**;  
 **else  
 return left**.getLeftmostData( );  
 }  
  
  
  
 **public** BTNode<E> getRight( )  
 {  
 **return right**;  
 }  
  
  
  
 **public** E getRightmostData( )  
 {  
 **if** (**left** == **null**)  
 **return data**;  
 **else  
 return left**.getRightmostData( );  
 }  
  
  
  
 **public void** inorderPrint( )  
 {  
 **if** (**left** != **null**)  
 **left**.inorderPrint( );  
 System.***out***.println(**data**);  
 **if** (**right** != **null**)  
 **right**.inorderPrint( );  
 }  
  
  
  
 **public boolean** isLeaf( )  
 {  
 **return** (**left** == **null**) && (**right** == **null**);  
 }  
  
  
  
 **public void** preorderPrint( )  
 {  
 System.***out***.println(**data**);  
 **if** (**left** != **null**)  
 **left**.preorderPrint( );  
 **if** (**right** != **null**)  
 **right**.preorderPrint( );  
 }  
  
  
  
 **public void** postorderPrint( )  
 {  
 **if** (**left** != **null**)  
 **left**.postorderPrint( );  
 **if** (**right** != **null**)  
 **right**.postorderPrint( );  
 System.***out***.println(**data**);  
 }  
  
  
  
 **public void** print(**int** depth)  
 {  
 **int** i;  
  
 *// Print the indentation and the data from the current node:* **for** (i = 1; i <= depth; i++)  
 System.***out***.print(**" "**);  
 System.***out***.println(**data**);  
  
 *// Print the left subtree (or a dash if there is a right child and no left child)* **if** (**left** != **null**)  
 **left**.print(depth+1);  
 **else if** (**right** != **null**)  
 {  
 **for** (i = 1; i <= depth+1; i++)  
 System.***out***.print(**" "**);  
 System.***out***.println(**"--"**);  
 }  
  
 *// Print the right subtree (or a dash if there is a left child and no left child)* **if** (**right** != **null**)  
 **right**.print(depth+1);  
 **else if** (**left** != **null**)  
 {  
 **for** (i = 1; i <= depth+1; i++)  
 System.***out***.print(**" "**);  
 System.***out***.println(**"--"**);  
 }  
 }  
  
  
  
 **public** BTNode<E> removeLeftmost( )  
 {  
 **if** (**left** == **null**)  
 **return right**;  
 **else** {  
 **left** = **left**.removeLeftmost( );  
 **return this**;  
 }  
 }  
  
  
  
 **public** BTNode<E> removeRightmost( )  
 {  
 **if** (**right** == **null**)  
 **return left**;  
 **else** {  
 **right** = **right**.removeRightmost( );  
 **return this**;  
 }  
 }  
  
  
 **public void** setData(E newData)  
 {  
 **data** = newData;  
 }  
  
  
  
 **public void** setLeft(BTNode<E> newLeft)  
 {  
 **left** = newLeft;  
 }  
  
  
  
 **public void** setRight(BTNode<E> newRight)  
 {  
 **right** = newRight;  
 }  
  
  
  
 **public static** <E> BTNode<E> treeCopy(BTNode<E> source)  
 {  
 BTNode<E> leftCopy, rightCopy;  
  
 **if** (source == **null**)  
 **return null**;  
 **else** {  
 leftCopy = *treeCopy*(source.**left**);  
 rightCopy = *treeCopy*(source.**right**);  
 **return new** BTNode<E>(source.**data**, leftCopy, rightCopy);  
 }  
 }  
  
  
  
 **public static** <E> **long** treeSize(BTNode<E> root)  
 {  
 **if** (root == **null**)  
 **return** 0;  
 **else  
 return** 1 + *treeSize*(root.**left**) + *treeSize*(root.**right**);  
 }  
  
 }  
  
  
  
}

**OUTPUT**

