APP PROJECT   
  
RIFATH HOSSAIN [RA2211003010628]

NIPUN DHIMAN [RA2211003010626]

AMAN SHAW [RA2211003010650]

ARYAN KUMAR [RA2211003010608]

DHRUV SWAMI [RA2211003010617]

PRANAV SRIVASTAVA [RA2211003010647]

AnimalGuess.java

import java.lang.\*;  
import java.util.\*;  
import java.io.\*;  
public class AnimalGuess{  
 // class fields  
 private static DecisionTree *animalDecisionTree*;  
 private static BinaryTree<String> *position*;  
 private static BinaryTree<String> *newNode* = null;  
 private static String *builder* = "";  
public static void main(String[] args) {  
  
 *animalDecisionTree* = new DecisionTree("AnimalTree.txt");  
 *playGame*();  
 *writeGameFile*(args[0]);  
 }  
 private static void playGame(){  
 // keeps track of spot in tree  
 *position* = *animalDecisionTree*.getTree();  
 System.*out*.println();  
 // start of game, prints first few lines  
 System.*out*.println("Think of an animal.");  
 System.*out*.println("I’ll try to guess it.");  
 System.*out*.println("Answer with 'yes' or 'no'");  
 System.*out*.println(*position*.getData());  
 // when you’re not at a leaf  
 while (!*position*.isLeaf()){   
 // reads user input  
 Scanner reader = new Scanner(System.*in*);   
 String s = reader.nextLine();   
 // checks if user entered "no"  
 if (s.equals("no") || s.equals("No")){   
 // accesses next node  
 System.*out*.println(*position*.getRight().getData());   
 // moves position  
 *position* = *position*.getRight();   
 }  
 // else =, same as above, but for "yes"  
 else{   
 System.*out*.println(*position*.getLeft().getData());  
 *position* = *position*.getLeft();  
 }  
 }   
 // after printing a leaf, the computer asks if its correct  
 System.*out*.println("Did I guess it?");  
 Scanner finalRead = new Scanner(System.*in*);  
 String finalS = finalRead.nextLine();  
 // if leaf is incorrect and user thought of unique animal  
 if (finalS.equals("no") || finalS.equals("No")){   
 System.*out*.println(*position*.isLeaf());  
 // gets animal from user  
 System.*out*.println("What animal were you thinking of?");   
 String animal = finalRead.nextLine();   
 // gets question from user  
 System.*out*.println("What is a question that will distinguish it from a "+*position*.getData()+"?");   
 String question = finalRead.nextLine();  
 // gets answer to question from user  
 System.*out*.println("Will the answer to that question be yes or no for your animal?");   
 String answer = finalRead.nextLine();  
 // based on answer to question, adds new info to tree  
 if (answer.equals("yes") || answer.equals("Yes")){  
 // new node for user’s animal  
 BinaryTree<String> newNode1 = new BinaryTree<String>(animal);  
 //moves old leaf to node opposite user’s animal  
 BinaryTree<String> newNode2 = new BinaryTree<String>(*position*);   
 // moves user’s question to old leaf  
 *position*.setLeft(newNode1);  
 *position*.setRight(newNode2);  
 *position*.setData(question);   
 }  
 // same this on opposite side  
 else {   
 BinaryTree<String> newNode1 = new BinaryTree<String>(animal);  
 BinaryTree<String> newNode2 = new BinaryTree<String>(*position*);  
 *position*.setLeft(newNode2);  
 *position*.setRight(newNode1);  
 *position*.setData(question);  
 }  
 }  
 // if the computer guesses the animal  
 else{  
 System.*out*.println("Hooray, I got it!");  
 System.*out*.println();  
 }  
 // asks user if they want to play again  
 System.*out*.println("Do you want to play again - yes or no?");  
 Scanner replay = new Scanner(System.*in*);  
 String replayString = replay.nextLine();   
 // either replay or exit method  
 if (replayString.equals("yes") || replayString.equals("Yes")){  
 *playGame*();  
 }  
 }  
private static void writeGameFile(String fname){   
 // holds the "yynn.." sequence  
 String builder = "";  
 // buffer for writing to file  
 PrintWriter outbuffer = null;   
 try {  
 outbuffer = new PrintWriter(new FileWriter(fname));  
 }  
 catch (IOException e){  
 System.*err*.printf("Problem writing file " + fname + "\n");  
 System.*exit*(-1);  
 }   
 // calls printNode to print all lines in file  
 *printNodes*(*animalDecisionTree*.getTree(), builder, outbuffer);  
 outbuffer.close();  
 } private static void printNodes(BinaryTree<String> btPointer, String builder, PrintWriter outstream){  
 // if the tree isn’t null  
 if (btPointer != null) {   
 // prints the data to file  
 outstream.print(builder+" " + btPointer.getData() + "\n");   
 // gets left and adds "Y"   
 *printNodes*(btPointer.getLeft(), builder.concat("Y"), outstream);   
 // gets right and adds "N"  
 *printNodes*(btPointer.getRight(), builder.concat("N"), outstream);   
 }  
 }  
}

BinaryTree.java

public class BinaryTree<E> {private E data;private BinaryTree<E> left;private BinaryTree<E> right;public BinaryTree(E data) {  
 this.data = data;  
 left = right = null;  
 }  
 public BinaryTree(E data, BinaryTree<E> left, BinaryTree<E> right) {  
 this.data = data;  
 this.right = right;  
 this.left = left;  
 }  
 public BinaryTree(BinaryTree<E> tree) {  
 this.data = tree.data;  
 this.left = (tree.left == null)? null:(new BinaryTree<E>(tree.left));  
 this.right = (tree.right == null)? null:(new BinaryTree<E>(tree.right));  
 }  
   
 public E getData() { // FILL IN RETURN TYPE  
 return data;  
 }public BinaryTree<E> getLeft() { // FILL IN RETURN TYPE  
 return left;  
 }public BinaryTree<E> getRight() { // FILL IN RETURN TYPE  
 return right;  
 }public void setData(E data) { // FILL IN ARGUMENT TYPE  
 this.data = data;  
 }public void setLeft(BinaryTree<E> left) { // FILL IN ARGUMENT TYPE  
 this.left = left;  
 }public void setRight(BinaryTree<E> right) { // FILL IN ARGUMENT TYPE  
 this.right = right;  
 }public static boolean isEmpty(BinaryTree node) {  
 return (node == null);  
 }public boolean isLeaf() {  
 if (left == null && right == null){  
 return true;  
 }  
  
 else {  
 return false;  
 }  
  
 }public boolean isBranch() {  
 if (left != null || right != null){  
 return true;  
 }  
 else {  
 return false;  
 }  
 }private static void indent(int n) {  
 for (int i = 0; i < n; i++) {  
 System.*out*.print(" ");  
 }  
 }public void print() {  
 System.*out*.print("Root: ");  
 printIndented(0);  
 }private void printIndented(int depth) {  
 System.*out*.println(data);  
 depth++;  
   
 if (left != null) {  
 *indent*(2\*depth);  
 System.*out*.print("Left: ");  
 left.printIndented(depth);  
 }  
 if (right != null){  
 *indent*(2\*depth);  
 System.*out*.print("Right: ");  
 right.printIndented(depth);  
 }  
 }  
 }

DecisionTree.java

import java.io.\*;  
import java.util.\*;  
import java.lang.\*;public class DecisionTree {  
   
 // var tree is the BinaryTree that will be built in class  
 private BinaryTree<String> tree;public DecisionTree(String fname){  
 // reads the file  
 BufferedReader br = null;   
 // keeps track of line  
 String currentLine = null;  
 // top of tree  
 BinaryTree<String> topNode = null;   
 // keeps track of location in tree  
 BinaryTree<String> currentNode = null;   
 // opens the file  
 try {  
 br = new BufferedReader(new FileReader(fname));  
 }  
 catch (IOException e){  
 System.*err*.printf("Problem reading file " + fname + "\n");  
 System.*exit*(-1);  
 }  
 //Read first line  
 try{  
 currentLine = br.readLine();  
 }  
 catch (IOException e) {  
 System.*err*.printf("Problem reading input\n");  
 System.*exit*(-1);  
 }  
  
 // keeps reading lines until done  
 while(currentLine != null){  
 // splits line at space  
 String[] tempList = currentLine.split(" ");   
   
 // lenth of "yynn.." sequence  
 int len = tempList[0].length();  
 currentNode = topNode;  
  
 // if the "yynn.." sequence is not empty  
 if (len > 0){  
 // traverse all nodes except last (up to insertion point)  
 for ( int i = 0; i<len-1; i++){  
 // current Y or N  
 char elem = tempList[0].charAt(i);  
   
 if (elem == 'y' || elem == 'Y'){  
 currentNode = currentNode.getLeft();  
 }  
 else if (elem == 'n' || elem == 'N'){  
 currentNode = currentNode.getRight();  
 }  
 else {  
 System.*err*.printf("Problem with data in input file\n");  
 System.*exit*(-1);  
 }  
 }  
 }  
 // gets question or answer from line  
 String question = currentLine.substring(len +1);  
   
 // sets first line to top node  
 if (len == 0) {  
 topNode = tree = new BinaryTree<String>(question);  
 }   
 // sets answer/question to appropriate node  
 else {  
 // elem = last letter in "yynn.. sequence"  
 char elem = tempList[0].charAt(len-1);  
   
 // if elem = y, it’s a left leaf  
 if (elem == 'y' || elem =='Y'){  
 currentNode.setLeft(new BinaryTree<String>(question));  
 }  
 // if elem = n, it’s a right leaf  
 else if (elem == 'n' || elem == 'N'){  
 currentNode.setRight(new BinaryTree<String>(question));  
 }  
 else {  
 System.*err*.printf("Problem with data in input file\n");  
 System.*exit*(-1);  
 }  
 }   
 //Read next line  
 try{  
 currentLine = br.readLine();  
 }  
 catch (IOException e) {  
 System.*err*.printf("Problem reading input\n");  
 System.*exit*(-1);  
 }  
 }  
 }public BinaryTree<String> getTree(){  
 return tree;  
 }  
}

AnimalTree.txt

Is it a mammal?  
Y Does it walk on two legs?  
YY Does it have a pocket?  
YYY Kangaroo  
YYN Gorilla  
YN Do people own them as pets?  
YNY Is it known as "man’s best friend"?  
YNYY Dog  
YNYN Is it a farm animal?  
YNYNY Pig  
YNYNN Cat  
YNN Lion  
N Is it a fish?  
NY Does it lie on the ocean floor?  
NYY Starfish  
NYN Shark  
NN Does it have a long tongue?  
NNY Frog  
NNN Does it fly?  
NNNY Is it an insect?  
NNNYY Fly  
NNNYN Bird  
NNNN Turtle

Output:

Think of an animal.

I’ll try to guess it.

Answer with 'yes' or 'no'

Is it a mammal?

yes

Does it walk on two legs?

no

Do people own them as pets?

yes

Is it known as "man’s best friend"?

yes

Dog

Did I guess it?

yes

Hooray, I got it!