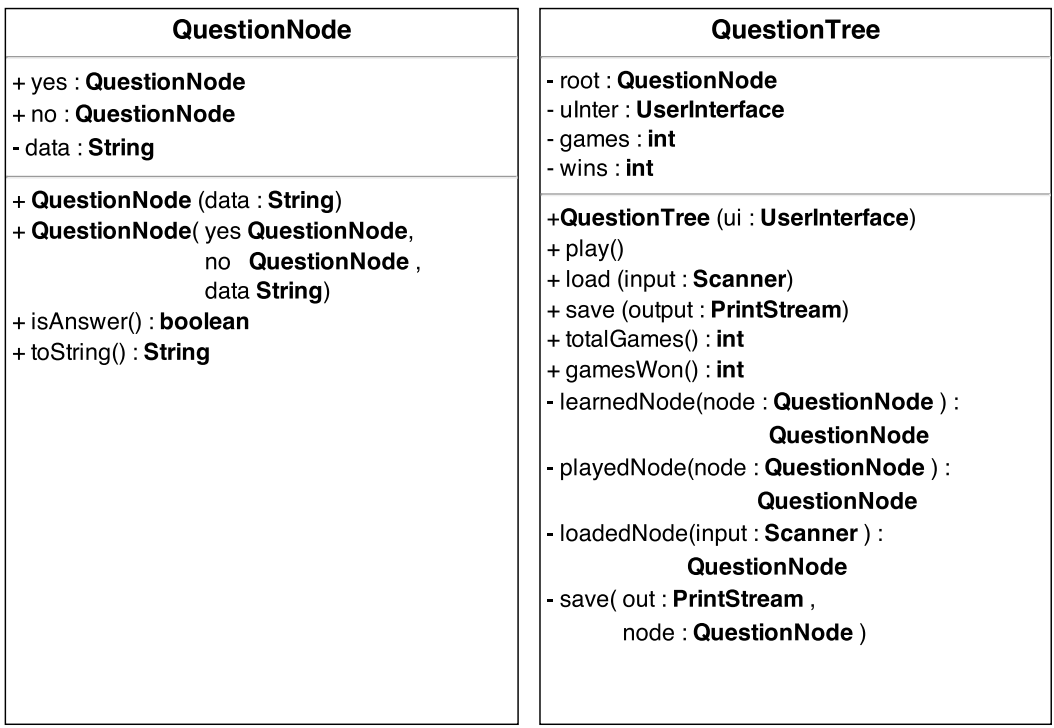
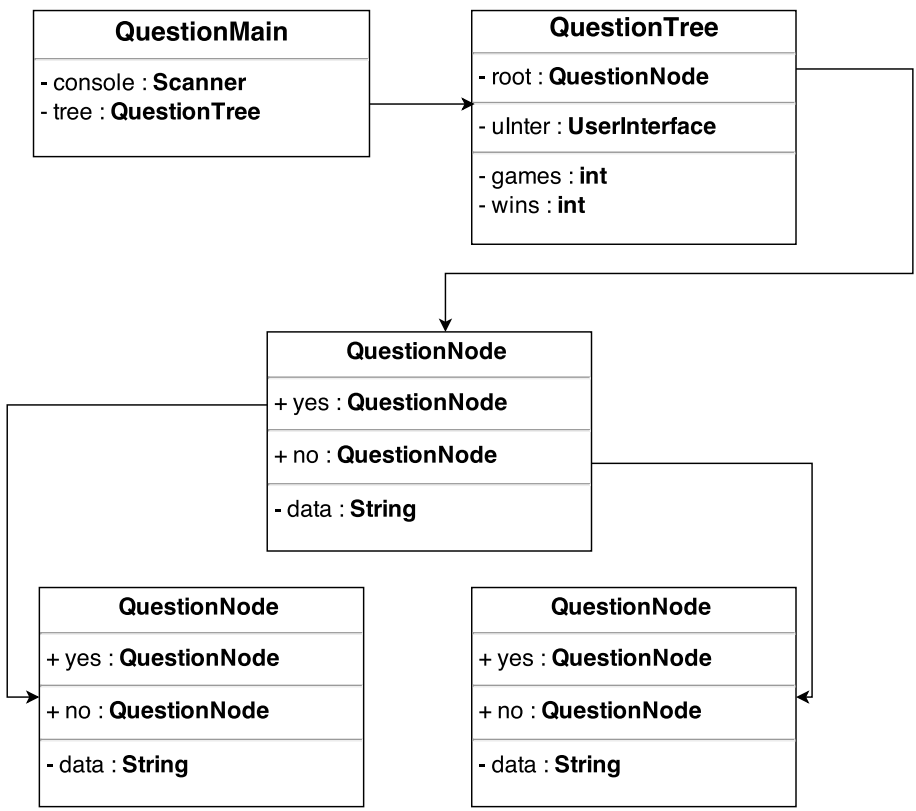
**HAI H NGUYEN**
**SID: 844 920 234**



**HAI H NGUYEN**
**SID: 844 920 234**



```
 1 import java.io.*;
 2 import java.util.*;
 3
 4 /**
 5  * This class models a Binary Question Tree.
 6  *
 7  *  It first initialize an empty Question Node as its root.
 8  *  If a previous game data exist, user can have it loaded into the tree
 9  *  When a game is finished, user can have the tree save its structure
10  *  When a game is played, the tree expands per lost game
11  *  using data given by user
12  *
13  * <ul>
14  * <li> Name: QuestionTree.java
15  * <li> Description: Question Tree
16  * <li> Class: Java 145
17  * <li> Instructor: Ken Hang && Janet Ash
18  * <li> Date: March 16 2015
19  * </ul>
20  * @author  Hai H Nguyen (Bill)
21  * @version Winter 2015
22  */
23
24 public class QuestionTree {
25
26     private QuestionNode root;
27
28     private UserInterface uInter;
29
30     private int games;
31
32     private int wins;
33
34     /**
35      * Constructor which initialize the QuestionNode
36      * and assign the given User Interface
37      * @param ui            The passed User Interface
38      */
39     public QuestionTree(UserInterface ui){
40         root = new QuestionNode("computer");
41
42         uInter = ui;
43
44         games = 0;
45
46         wins = 0;
47     }
48
49     /**
50      * Play a complete guessing game
51      * and rebuild the tree as necessary
52      */
53     public void play() {
54         root = playedNode(root);
55
56         ++games;
57     }
58
59     private QuestionNode playedNode(QuestionNode node){
60         if (node.isAnswer()) { // If a node is an Answer node
61             uInter.print("Would your object happen to be " + node + "?");
62
63             if (uInter.nextBoolean()) { // If user accepts the answer
64                 uInter.println("I win!");
65                 // Increase the amount of time the machine's won
66                 ++wins;
67             } else { // Else the machine learn a new question node
68                 node = learnedNode(node);
```

```
            }
        } else { // If node is a Question node
            uInter.print(node.toString());
            // Trace down to the node user wanted
            if (uInter.nextBoolean()) {
                node.yes = playedNode(node.yes);
            } else {
                node.no = playedNode(node.no);
            }
        }
        return node;
    }

    private QuestionNode learnedNode(QuestionNode node){
        uInter.print("I lose. What is your object? ");
        // Make new answer node for the new object
        QuestionNode newNode = new QuestionNode (uInter.nextLine());

        uInter.print("Type a yes/no question to distinguish " +
                "your item from " + node + ":");
        // Get user's question
        String query = uInter.nextLine();

        uInter.print("And what is the answer for your object?");
        // Make new Question Node per user's command
        return uInter.nextBoolean() ?
                new QuestionNode ( newNode, node, query ):
                new QuestionNode ( node, newNode, query );
    }

    /**
     * Invoke the Save Recursion and output
     * the Question Tree into the output Stream
     * @param output        Output object to Stream data
     */
    public void save (PrintStream output){
        save(output, root);
    }

    private void save(PrintStream out, QuestionNode node){
        if (node.isAnswer()){
            out.print("A:" + node);
        } else {
            out.println("Q:" + node);

            save(out, node.yes);

            out.println();
            // Avoid new line at the end
            save(out, node.no);
        }
    }

    /**
     * Invoke the Load recursion and build
     * the Question Tree in a binary fashion
     * @param input         Scanner Object to get Data
     */
    public void load (Scanner input){
        root = loadedNode (input);
    }

    private QuestionNode loadedNode (Scanner input){
        QuestionNode node = null; // Initialise a null node.

        if (input.hasNext()){
            String[] data = input.nextLine().split(":",2); // Get Data and Split Once
            // If Begins with A
```

```
137             if (data[0].equals("A")){// Create Answer node
138                 node = new QuestionNode(data[1]);
139             } else { // Else Create Question node
140                 node = new QuestionNode (   loadedNode(input),
141                                             loadedNode(input),
142                                             data[1]);
143             }
144         }
145 
146         return node;
147     }
148 
149     /**
150      * @return      How many games had been played
151      */
152     public int totalGames(){
153         return games;
154     }
155 
156     /**
157      * @return      How many games the machine had won
158      */
159     public int gamesWon(){
160         return wins;
161     }
162 }//IS29
```

```
 1
 2 /**
 3  * This class Models both an Answer OR Question node of a Question tree.
 4  *
 5  * The data field was restricted in order to distinguish
 6  * between a Question node and an Answer node.
 7  *
 8  * The Constructor with only data as variable generates
 9  * an Answer Node whereas the other generates
10  * a Question Node which spans to two
11  * other Nodes.
12  *
13  * <ul>
14  * <li> Name: QuestionNode.java
15  * <li> Description: Question Node
16  * <li> Class: Java 145
17  * <li> Instructor: Ken Hang && Janet Ash
18  * <li> Date: March 10 2015
19  * </ul>
20  * @author  Hai H Nguyen (Bill)
21  * @version Winter 2015
22  */
23 public class QuestionNode {
24
25     /**
26      *  Pointer to the Yes node
27      */
28     public QuestionNode yes;
29
30     /**
31      *  Pointer to the No node
32      */
33     public QuestionNode no;
34
35     private String data;
36
37     /**
38      *  Constructor, Initialize an Answer Node
39      * @param data          Answer issues to user
40      */
41     public QuestionNode(String data){
42         this(null, null, data);
43     }
44
45     /**
46      *  Constructor, Initialize a Question Node
47      * @param yes            Yes Node answer
48      * @param no             No Node answer
49      * @param data           Question to Ask
50      */
51     public QuestionNode(QuestionNode yes,
52                         QuestionNode no,
53                         String data){
54         this.yes = yes;
55         this.no = no;
56         this.data = data;
57     }
58
59     /**
60      * @return              True if the node doesn't have any child, False otherwise
61      */
62     public boolean isAnswer(){
63         return yes == null && no == null;
64     }
65
66     /**
67      * @return              Data of the node, either a question or an answer
68      */
```

```
69     public String toString(){
70         return data;
71     }
72 }//IS29
```