# CSE214

## HOMEWORK - FALL 2012

---

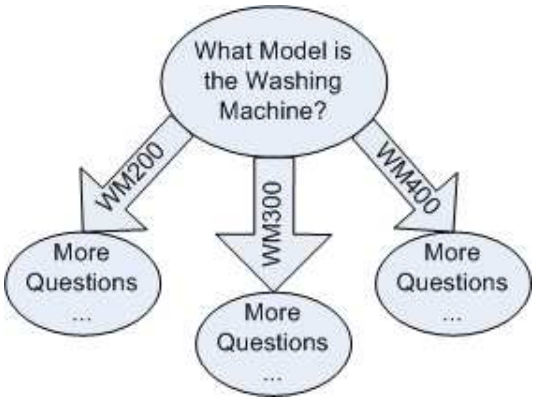
## HOMEWORK 5 - due Tuesday November 6, before 5:00pm.

**REMINDERS:**

- **Be sure your code follows the coding style for CSE214.**
- **Make sure you read the warnings about academic dishonesty. *Remember, all work you submit for homework or exams MUST be your own work.***
- **Click here to read through the steps to make sure you hand in your homework successfully. Also, login to your grading account and click "Submit Assignment" to upload and submit your assignment. Make sure that you submit the same files on both systems.**
- **You are not allowed to use Java API Tree, ArrayList, Vector, LinkedList or any other Java API Data Structure classes to implement this assignment.**
- **You may use Scanner, InputStreamReader, or any other class that you wish for keyboard input.**

**Design a program that will be used as a versatile decision-making system based on user input. There are several places where this application can be used. An example is online Technical Support for a product or group of products. A series of questions might be asked and based on the answers, a different branch of questions will be asked as follow-up-questions to determine exactly how to help the user. Another example is a telephone Technical Support, like a 1-800 number. When calling a 1-800 number, you might communicate with a machine. A series of questions are asked and as answers you typically press the numbers on the keypad, 1-9. After determining exactly what the problem is, a simple answer is usually possible.**

**In this assignment, you will use Ternary (3-ary) trees to represent the flow of data. In each node in the Ternary tree there will be a question to ask. The "answers" will be branches towards more questions in the tree. *For the normal assignment, it will be a Ternary (3-ary) tree, meaning that any node can have at most three children. The extra credit part of this assignment should handle 9-ary trees (up to 9 children).* For example:**




**1. Write a fully-documented class named `TreeNode` that stores the data in each node. It should have:**

- **Three references to other TreeNodes. Left, Middle, Right.**

  **OR**

- **An array with three references to other TreeNodes.** ***If you're doing the extra credit, the array should store potentially 9 TreeNodes.***
- `private String label`
  **The "name" of this TreeNode. It will be used when constructing the tree. More info on labels will follow.**
- `private String message`
  **The message will be displayed to the screen. It will either be a question or just a normal message. If this TreeNode has children TreeNodes, it should then traverse one level downwards to each child node to get the prompts as answers. If this TreeNode is a leaf node, then it should just display the message which will act as a solution.**
- `private String prompt`
  **The prompt will be displayed to the screen as a possible answer to a question. For example, when starting at the root of the tree, there will be a question (message) to ask, and as possible answers the root will traverse one level downwards each child TreeNode to display the prompt as a possible answer.**

**This class should have a constructor, mutator and accessor methods for each instance variable. You may also want to have a method to determine if this TreeNode is a leaf (no children).**

**2. Write a fully-documented class named `Tree` that will have a reference to the root of the tree and other useful methods described below:**

- **Constructor.**
- `public boolean addNode(String label, String prompt, String message, String parentLabel)`
  **A method to add a TreeNode to the tree. The location will be a child of parentLabel. The child node should be left justified meaning that it should first be placed in the left most TreeNode reference, then the middle, then the right. A return value of true indicates that the node was successfully added to the tree. Otherwise, the return value is false. More info on the label is in the input file format.**
  ***Note: You can use a different method signature if you need to, or define a separate method for adding each child (i.e. addNodeLeft, addNodeMiddle, addNodeRight).***
- `public TreeNode getNodeReference(String label)`
  **Returns a reference to the TreeNode that has the given label. The return value is `null` if the label is not found. This may be helpful to implement the above method. Efficiency is not important and you can use any traversal order you wish. Also, if you are more comfortable with placing this method in the TreeNode class, that is perfectly acceptable.**
- `public void preOrder()`
  **Traverses the tree in preorder, and prints the contents of the tree to the screen.**
  ***Note: This method can also be in the TreeNode class.***
- `public void beginSession()`
  **This method will be used to start the question and answer session.**

**Note: You may include additional instance variables or methods as needed.**

**3. Write a fully-documented class named `TreeDriver` which contains the main() method and it should display a menu:**

- **L (Load input file and build a tree)**
  **Prompt the user for a file name and build a *new* tree. The entire tree will be recreated each time this menu option is chosen.**
- **H (Start help session)**
  **This option will begin asking questions starting from the root of the tree; if there is no tree set up, display an error message. When displaying a question with answers, display each answer on a separate line with a number associated with it, similar to a menu. Include another option, 0 (zero) to exit the help session and return to the main menu. For example:**

```
What is the Model of the Washing Machine?
1) WM200
2) WM300
3) WM400
0) Exit Session.
```

**If the node is a leaf, then display the message and go back to the main menu. For example:**

```
You have to put in quarters to start the Washing Machine.
Thank you for using our automated help system.
(Menu...)
```

- **T (Traverse the tree in pre-order)**
  **Traverse the tree in pre-order and display the label, prompt, and message in each node on a separate line (see sample input/output).**
  **The order is root, left subtree, middle subtree, right subtree.**
- **Q (Quit)**
  **Quit the program.**

## INPUT FORMAT

- **All strings in the input file will be no longer than 60 characters *so that they will not take up more than one line.***
- **Blank lines in the input file are to be ignored.**
- **The length of the filename will not be longer than 20 characters.**

## OUTPUT FORMAT

- **Be sure your prompts and error messages for the user are clear and understandable.**
- **Be sure to have a menu after each operation.**

## INPUT FILE FORMAT

**Since this program is designed to be a versatile decision-making system based on user input, we will be using input files so that anyone with a tree of questions/answers can use this program, including but not limited to Tech Support. *It is strongly recommended that you post any sample text files you create in the Blackboard discussion board so that others can test their code.* The input file will always start with "root" on the first line. Look at the input file comments and ask any questions you may have on the discussion board.**

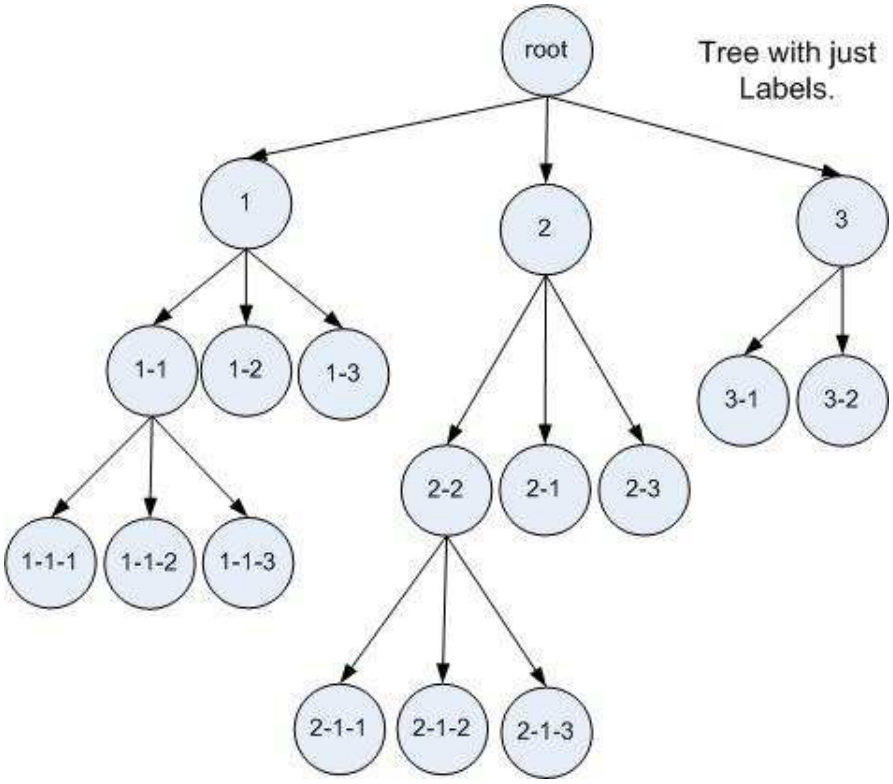
- **The format of the input file is as follows:**

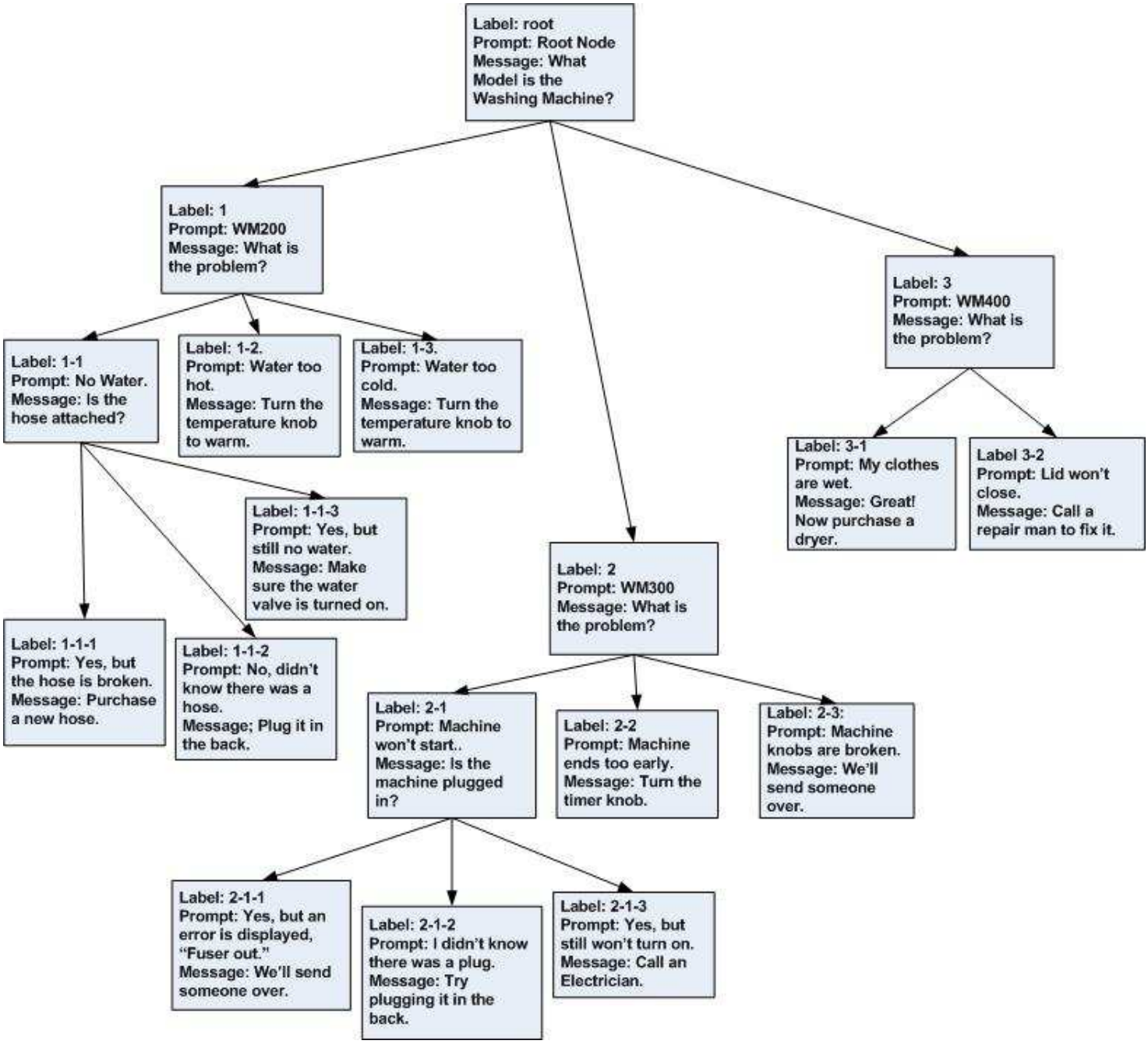
```
label, prompt, and message for the root node  (on three separate lines)
label1 <number-of-children>
label, prompt, and message for the first child of label1 (on three separate lines)
label, prompt, and message for the second child of label1 (on three separate lines)
.....
label2 <number-of-children>
label, prompt, and message for the first child of label2 (on three separate lines)
label, prompt, and message for the second child of label2 (on three separate lines)
.....
label3 <number-of-children>
label, prompt, and message for the first child of label3 (on three separate lines)
label, prompt, and message for the second child of label3 (on three separate lines)
.....
```

- **Label is a one-word string with no spaces that is used during the creation of a tree. It determines the location of a node in the tree. A picture of a tree with only their labels is shown below. Note that labels do not have to be in**

**any specific format, and they will not necessarily be made of numbers and dashes.**

- **Prompt is a string that will be printed to the screen as a choice or answer.**
  **(NOTE: The prompt for the root node should not be printed to the screen. Look at the example text file input.)**
- **Message is a string that will be displayed to the screen as either question or if the node is a leaf node, as a final message.**
- **All strings (label, prompt, message) in the input file will be no longer than 60 characters *so that they will not take up more than one line.***
- **Another possible line in the input file is the name of a previously defined label followed by a number, separated by at least one space. This number will indicate the number of children this node has. The child nodes will then be directly after this line.**
- **Any blank lines in the input file are to be ignored. They will help the creator of the input file for readability purposes.**
- **If at any point during the reading in of the text file, you encounter an error in the format, you may stop reading in the text file then report a message to the user.**
- ***For the normal assignment, a node can have 0, 1, 2, or 3 children. For EXTRA CREDIT, be ready to handle 0 through nine children.***
- **A sample input text file: Sample text file with comments , Sample text file without comments (this is the file that must be used to test your program).**
- **The visualization of the text file is the second tree below.**
- **For simplicity, we will not include comments in our test cases.**






## Extra Credit

- **Handle any number of child leaves. Note that since we may want this to be used in a telephone service, there will be anywhere from zero to nine children for any given node. 10 extra points.**
- **Display an option to go backwards in the tree (e.g. go back to previous menu). The option should be 'B'. 10 extra points.**
- **If you can come up with a creative idea regarding this programming assignment, then in your comments in the beginning of the driver class, write a brief description of what special functions your program does. The grade is determined per TA judgment. Up to 15 extra points.**

## SAMPLE INPUT/OUTPUT

**As usual, comments in green, user input in black, and computer output in blue.**

```
L - Load a Tree.
H - Begin a Help Session.
T - Traverse the Tree in preorder.
```

衰

衰

衰