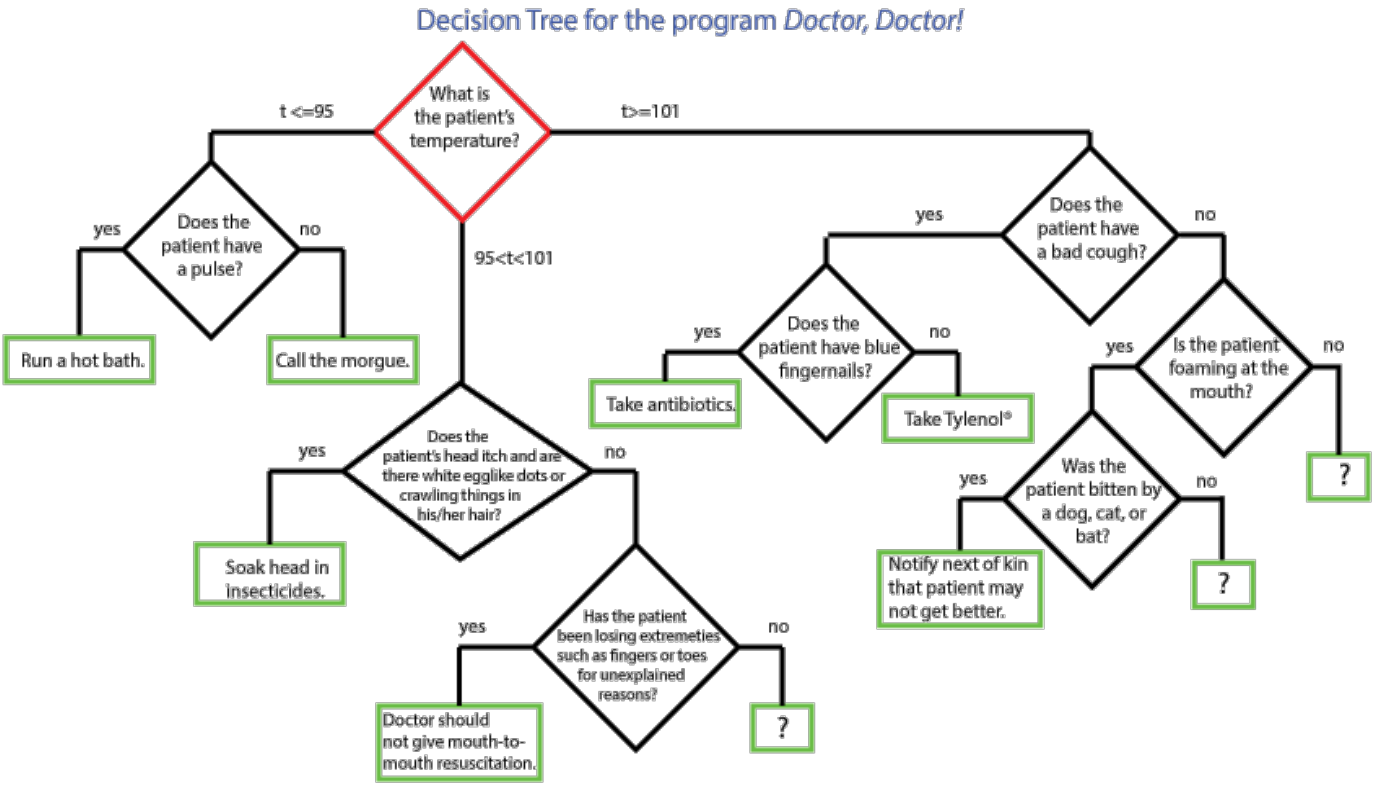
CS-141 Introduction to Programming
Writing an Expert System

Learning Objectives: Students will learn how to use nested if-else statements to implement a decision tree (the Expert System) in Java. Students also will have practice writing their own class.

Example

In lecture, you saw how to implement a medical diagnosis "expert system". It used a series of questions to arrive at a diagnosis. The structure of the rules can be displayed in a decision tree:

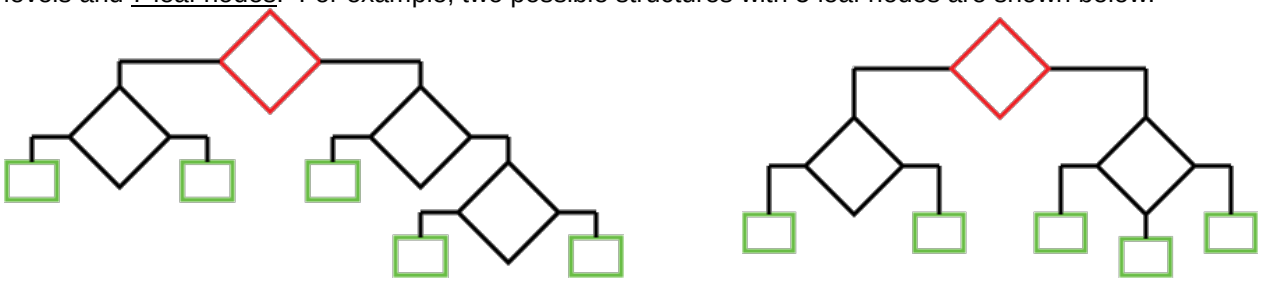


Each box is called a node. The top red node is called the root node; it is where the diagnosis begins. The green nodes at the bottom are called leaf nodes; they are where the final diagnosis can be found. Note that there are some missing leaves in the written rules. These are specified by "?" in the above tree. In these cases, it is assumed the doctor does not have a diagnosis; the program should let the user know this.

Assignment – Part 1: Software Implementation

Your assignment is as follows:

1. Pick a topic you know something about or something that would be fun to make up.
2. **Tree Design**: Carefully and neatly draw a decision tree. At a minimum, your decision tree should have at least two levels and 7 leaf nodes. For example, two possible structures with 5 leaf nodes are shown below:



3. Every decision should eventually lead to a response, i.e. there should be no missing leaf nodes (e.g. as in the "?" above). Also, try to be consistent in the structure, for example, in the tree Medical Diagnosis example above, the "yes" always appears on the left branch and the "no" on the right. This will make coding a bit easier. Be sure to complete the decision tree design before continuing.
4. **UML Design**: Create a UML Diagram of your ExpertSystem class. It won't have much in it. Follow what we did in class. The Doctor program would have a default constructor, a member variable (field) for the diagnosis variable, and a toString method. There could also be an accessor (getter) for any member variables you include. Note, you probably don't need a mutator (setter) since you don't want the user changing the value of variable input by the user.
5. **Implementation**: When you are ready to start coding, create a new Netbeans Java Application Project giving it a name appropriate to what it does (e.g. DoctorProject in the case of the medical diagnosis program). Have Netbeans create a Main Class. Once the project is created, create a second class which represents your expert system, giving it an appropriate name, e.g. Doctor.
6. Add a method that implements your decision tree. Use your UML and decision tree diagram to guide the writing of your code. Don't implement everything at once. Start with the top most if-else statement (i.e. the one right below the root) and compile & test (see part 7 below for testing code) before continuing. Maintaining proper indentation will significantly increase the readability of your code and decrease how much time you spend having to debug. If your tree has more than 2 levels, you should break it up into separate methods.
7. **Unit Testing**: In the main method (in your Main class), create an *instance* of your expert system and call the method which implements your tree. For example, for the DoctorProject in the main method we would include the code

```
Doctor doctor = new Doctor();
Doctor.diagnose();
```

8. When done, you should test every branch of your code by running it once for each possible "path" from the root to a leaf node. In the case of the Doctor program, there are 10 leaf nodes and thus the program must be run 10 times to thoroughly test the code.

Assignment – Part 2: Software Testing

This is to be done on the day the assignment is due. Be sure to being a copy of your decision tree.

1. Pick another team in the class to work with.
2. Write your name on your decision tree diagram. With the other team, exchange copies of your decision trees.
3. Write your partner's and your names on their decision tree paper and indicate that you and your partner are the "Testers".
4. Run their code at least once for each possible output (leaf node). To run their code, they can either give you a copy of their code on a thumb drive, or you can swap seats and run their code on their computer. Each time you run their code, check off the corresponding leaf in their decision tree. This will help you keep track of what you have tested.
5. Based on the results of running their program, answer the questions on the next page.
6. When you are done, staple together all of the following: both copies of the decision tree, both sets of answers to the questions, and paper copies of the code.
7. On WISE, submit an electronic copy of your 2 Java classes – Main.java and your Expert system (e.g. Doctor.java). Be sure to include the name of your partner.

CS-141 Introduction to Programming
Testing an Expert System

You and your partner: _____

The names of the team you are reviewing: _____

The area of expertise of the program you are reviewing: _____

1. Did the team's program correctly implement their decision tree? If not, what went wrong?

2. Was their program understandable and easy to use? For example, was the interface well formatted, were the questions clear, was it clear how you were supposed to answer, was the output clear? If not, what advice would you give the team for improving their program?

Is there anything that you particularly liked about their program?