First off, look at your main method and notice how the TreePrinter's constructor is invoked. Look also at the TestTreeModel class. TreePrinter should be written in such a way that it works with both the TestTreeModel and the VariableTreeModel. The thing to really notice is that these both implement the TreeModel interface (that's a hint as to what the type of TreePrinter's constructor's parameter type should be.) Because they implement TreeModel, they make available getRoot(), getChildCount(), and getChild(). TreePrinter uses these three TreeModel methods.

Next, look at the sample output from the first page of the lab handout. Notice the indentation --- a child is indented with a three space offset from its parent. Also remember that the ArrayList a in the main() method contains itself, hence the 'idref=0' line in the sample output.

The print() method that you write for TreePrinter uses recursion and a pre-order traversal to print the tree. Here's pseudo-code:

```
private void print(Object node, String indent) {
   int id = lookup(node);
   if (id >= 0)
      println "idref=" id;
   else
      add node to the lookup ArrayList;
      print an output line for node;
      increment the id counter;
      for each of node's children
         call print on the child, increasing the indent by 3 spaces;
}
```

From this code you can infer that you'll need to write a private lookup() method, you'll need an ArrayList (referred to as "lookup" above) that holds already-visited nodes, and you'll need an id counter. The ArrayList and the counter should be set up as private fields.

Look back at the main() method, and you'll realize that you need one more method. What is it, and how does it relate to main() and the TreePrinter methods describe above?