Composite Pattern

In this lab, you will create an application using the Composite pattern.

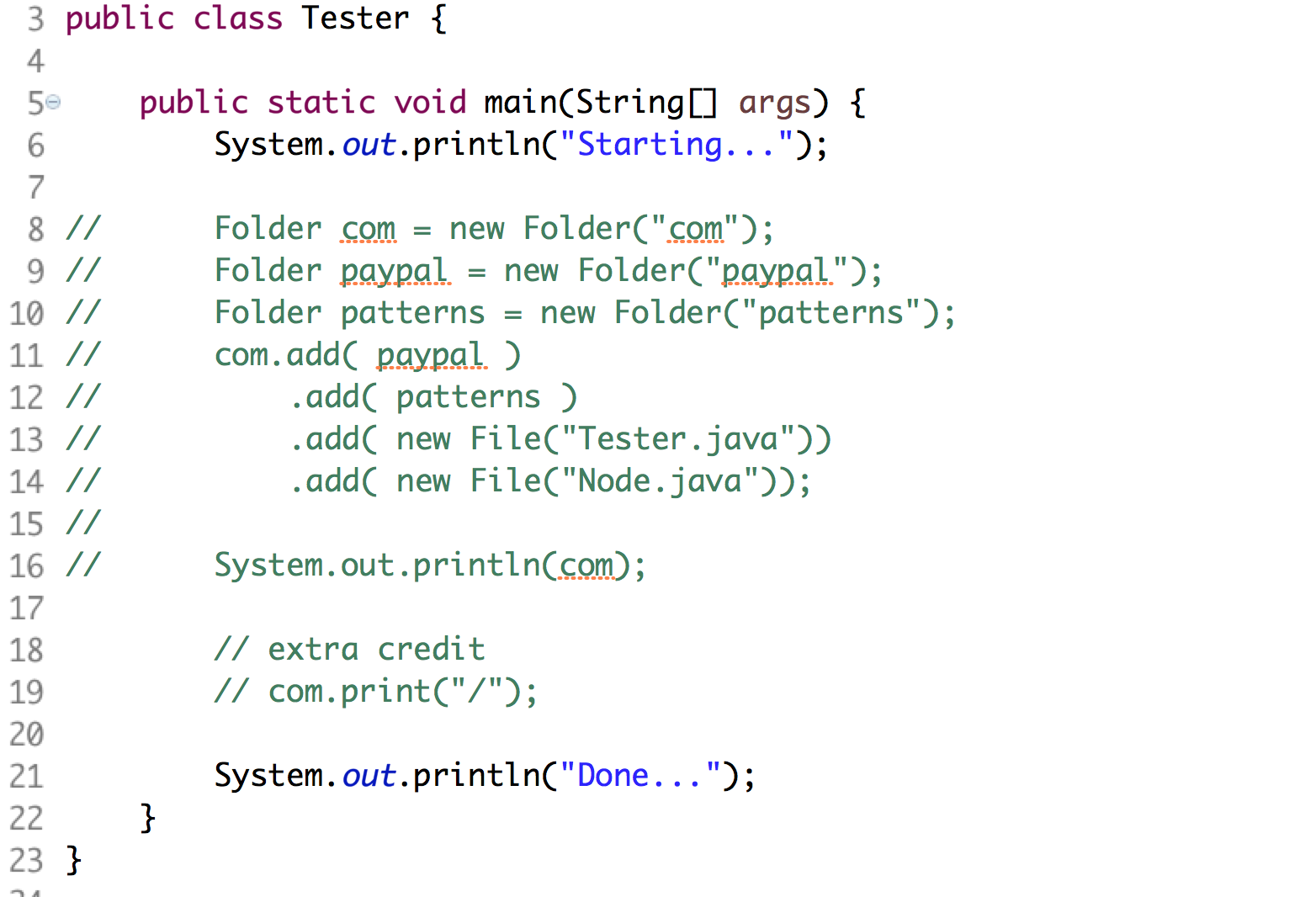
# Objectives

In this lab, you will

* Create File which is a Node
* Create a Folder which is a TreeNode
* Add Files and Folders to a Folder
* Display the result

# Exercise

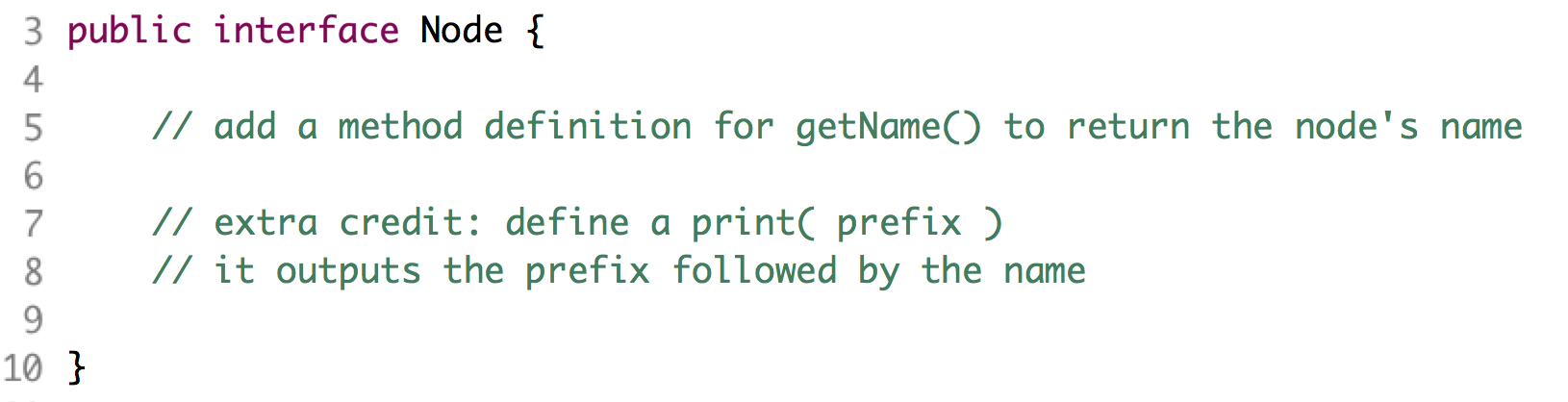
1. The Composite design pattern represents the concept of Parent -> Child -> GrandChild, etc… There are two types, a Node and a TreeNode. Everything is a Node. Every container is a TreeNode. This example models a file system with Folders and Files. The Folder is a TreeNode, and both are Nodes.
2. In Eclipse, open the package, com.paypal.patterns.Composite to see the project files.
3. Open the file, Tester.java, as shown below:



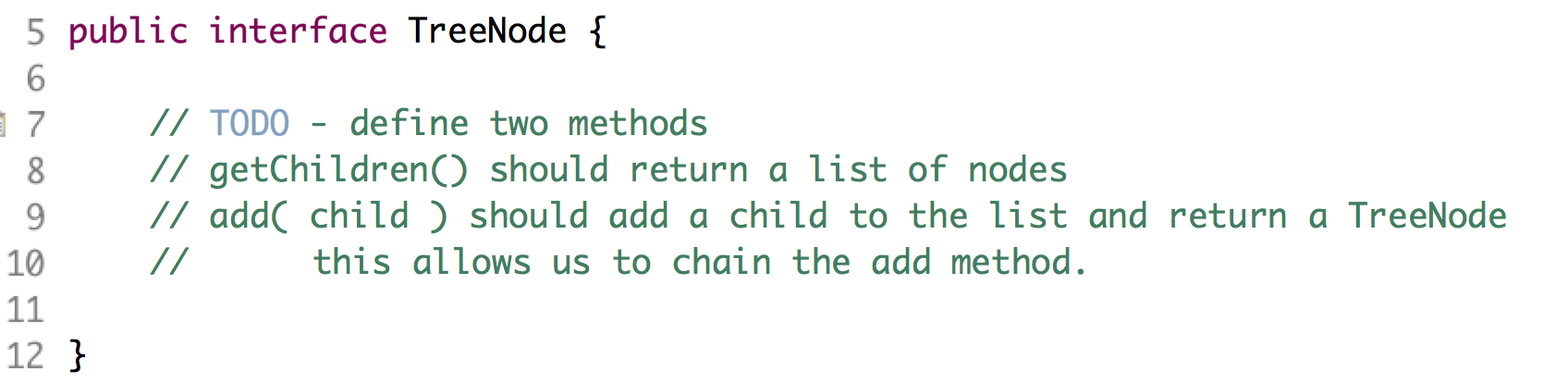
1. In lines 8-10, it creates three Folders. In lines 11-14, it adds Folders and Files to the “com” Folder. Notice the add() method returns a Folder, either the Folder added or the Parent Folder (if adding a File).
2. Lines 18-19 represent the Extra Credit problem shown below.

# Basic Types

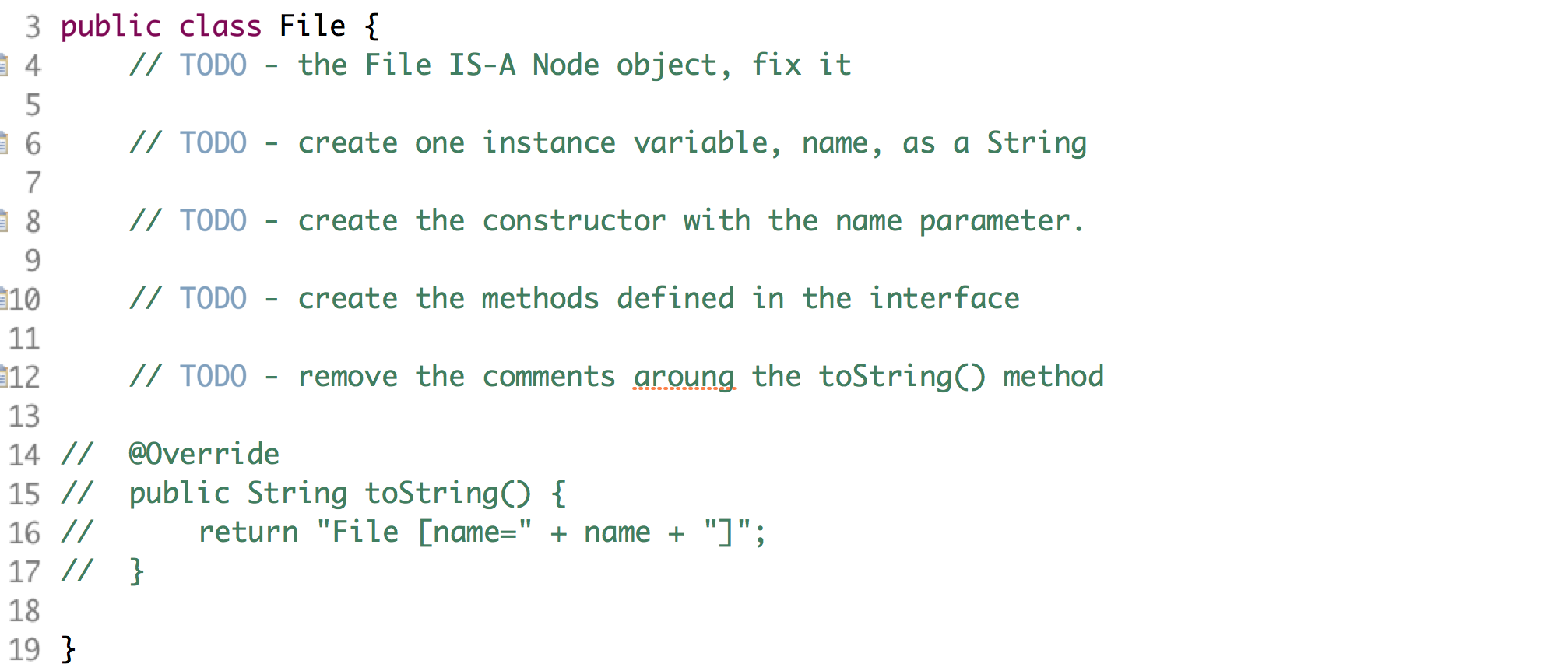
1. Open Node.java shown below:



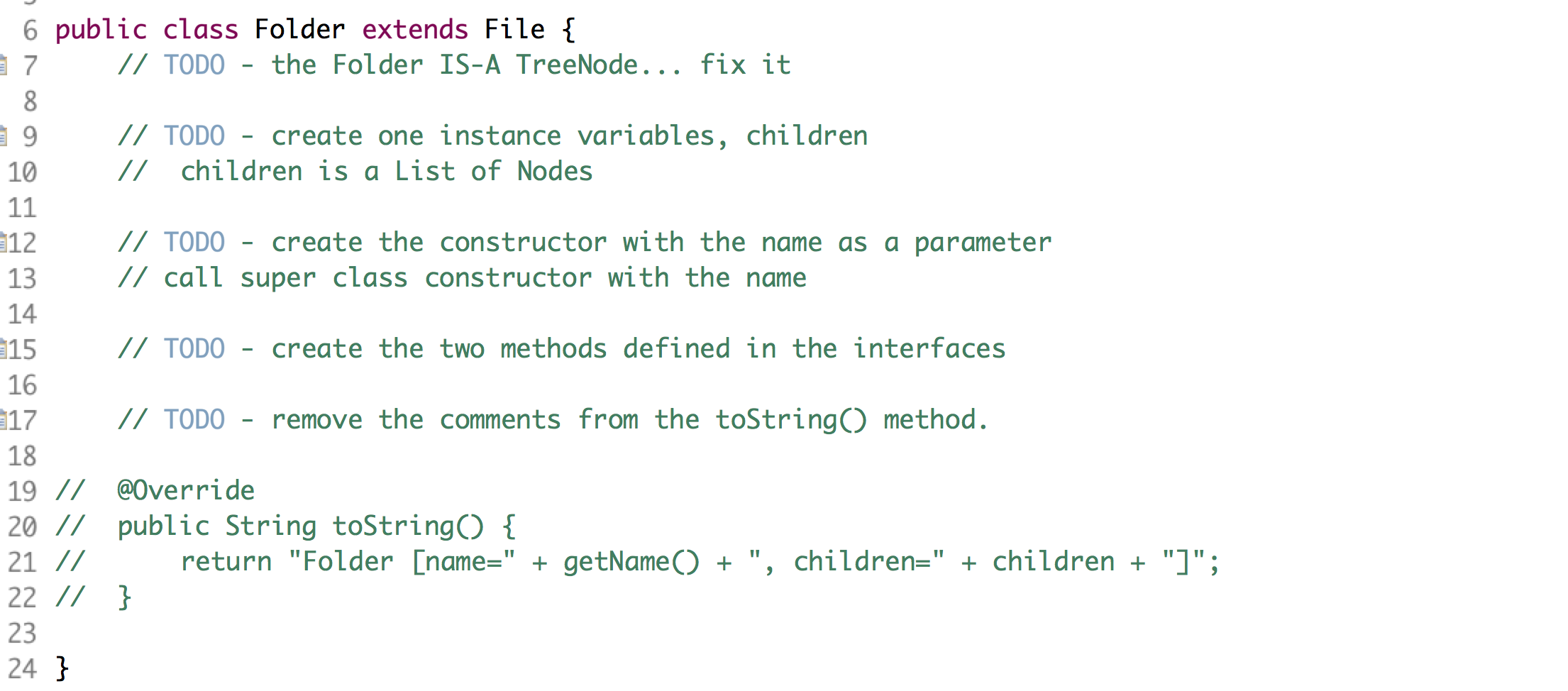
1. After line 5, create the signature of a getName() method to return the String name of the Node.
2. Open TreeNode.java shown below:



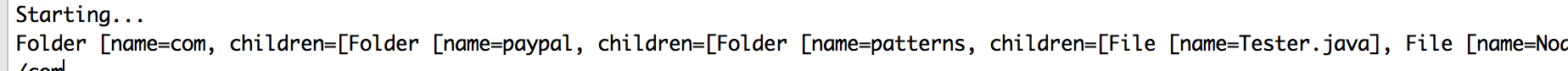
1. After line 7, create the signature of two methods, getChildren() which returns a List of Nodes, and add( Node child ) which returns a TreeNode.
2. Open File.java shown below:



1. On line 3, change the class definition to include the Interface, Node.
2. After line 6, create a private instance variable, name, as a String.
3. Follow the directions of the other TODO comments.
4. Uncomment lines 14-17 to enable the toString() method.
5. Open Folder.java shown below:

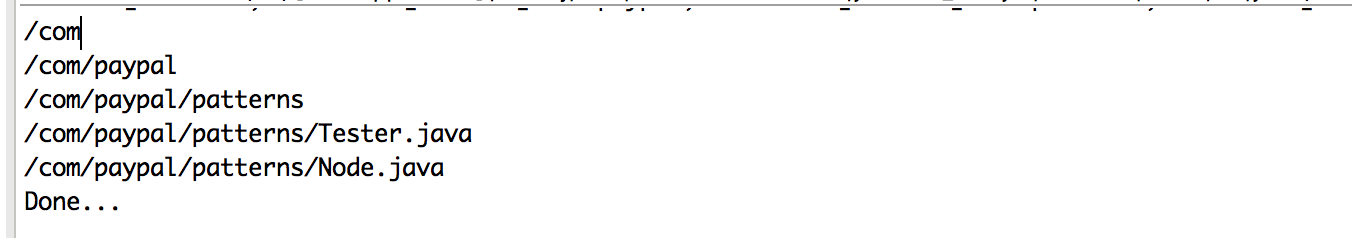


1. Notice the Folder IS-A File. It is also a TreeNode and should implement that interface.
2. Follow the directions of the other TODO comments.
3. Uncomment lines 19-22 to enable the toString() method.
4. Back in Tester.java, uncomment lines 8 - 16 and execute the file as a Java application.
5. The application should display all the files and folders created in a terrible format shown below.



# Extra Credit

1. Let’s fix the display of the names of the files and folders.
2. In the Node.java interface, add a new method, print( String prefix ) which returns a void.
3. In the File and Folder classes, create the print() method. It should output the prefix followed by the name field.
4. In the Folder.print() method. It should call the print() method on each child and pass the input prefix plus the Folder name.
5. NOTE: in the solution, I used a ‘/’ as a path separator. Use any character you wish.
6. Now, when you run the application, it should display:



Congratulations. You have completed this lab.