Visitor Pattern

In this lab, you will add the Visitor pattern to the Folder Composite.

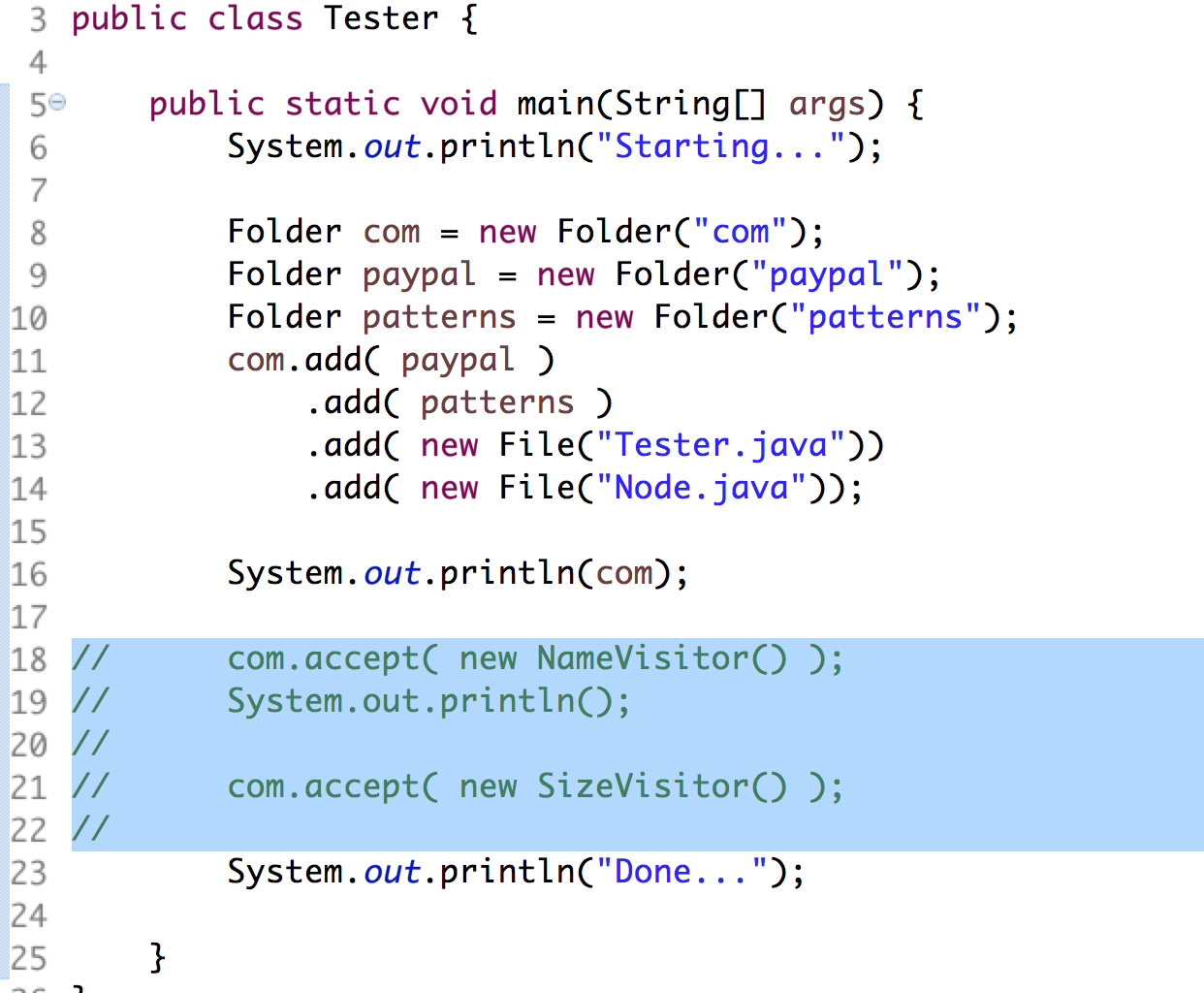
# Objectives

In this lab, you will

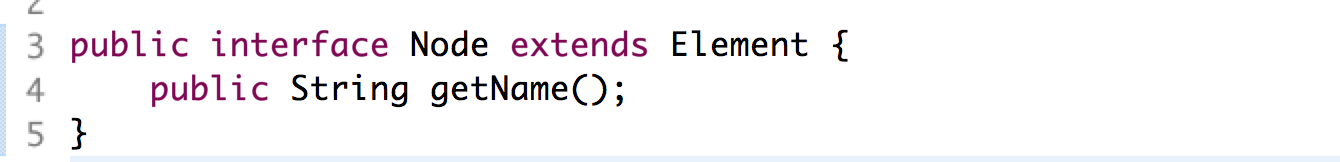
* Define the Visitor interfaces
* Add the methods to the Folder and File
* Invoke the visitor to the outermost layer
* Run the app

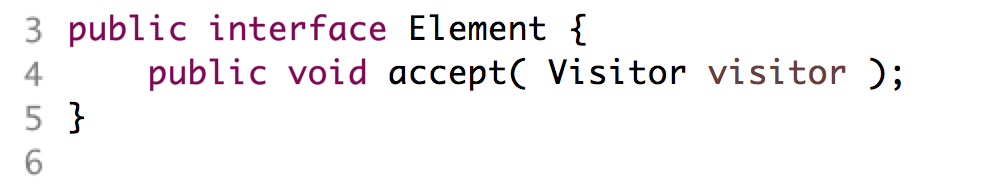
# Exercise

1. The Visitor pattern allows you to add multiple methods to an object at runtime instead of merely at compile time.
2. In this example, we will add the Visitor pattern to the Composite application previously run. We will create the same Folder and File structure as before, then, at runtime, we will invoke two different visitors using the same method, accept(), on the objects.
3. In Eclipse, in the exercises project, open the package com.paypal.patterns.Visitor to view the project files.
4. Examine the client, Tester.java, as shown below:

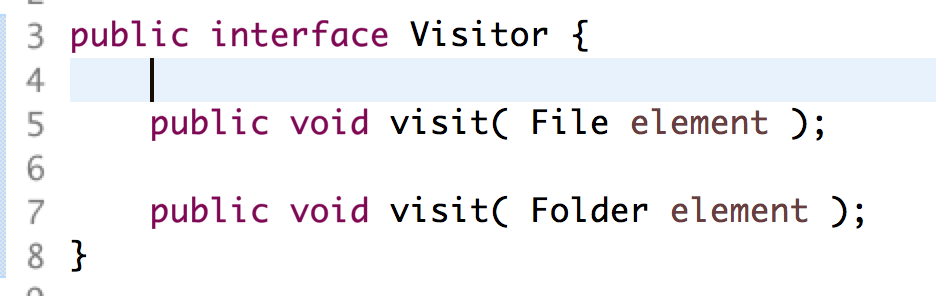


1. Lines 8-14 creates the file system structure as in the Composite solution.
2. Lines 18-21 invokes the Visitor in the outer most element in the file system.
3. We change the Node interface by adding the accept() method from the Element interface as shown below:

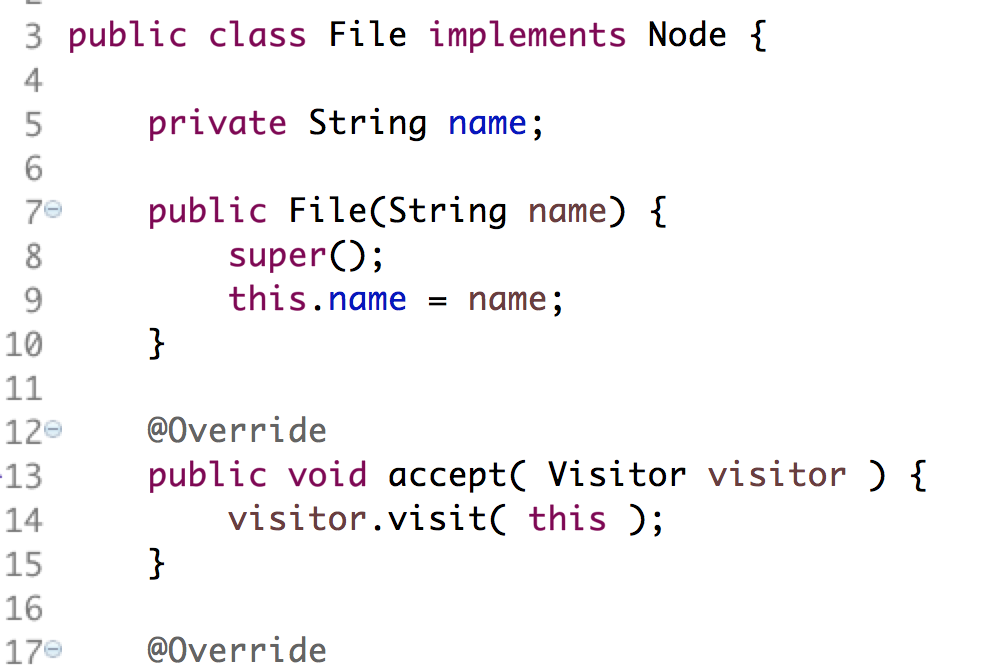




1. This ensures that all the concrete components in our model accept the Visitor object.
2. Examine the Visitor.java file shown below:



1. This defines a visit() method for EACH object with an accept(). In this case, both the File and Folder objects. Note how the visit() methods are overloaded with parameters of each element.
2. Examine the File object shown below:



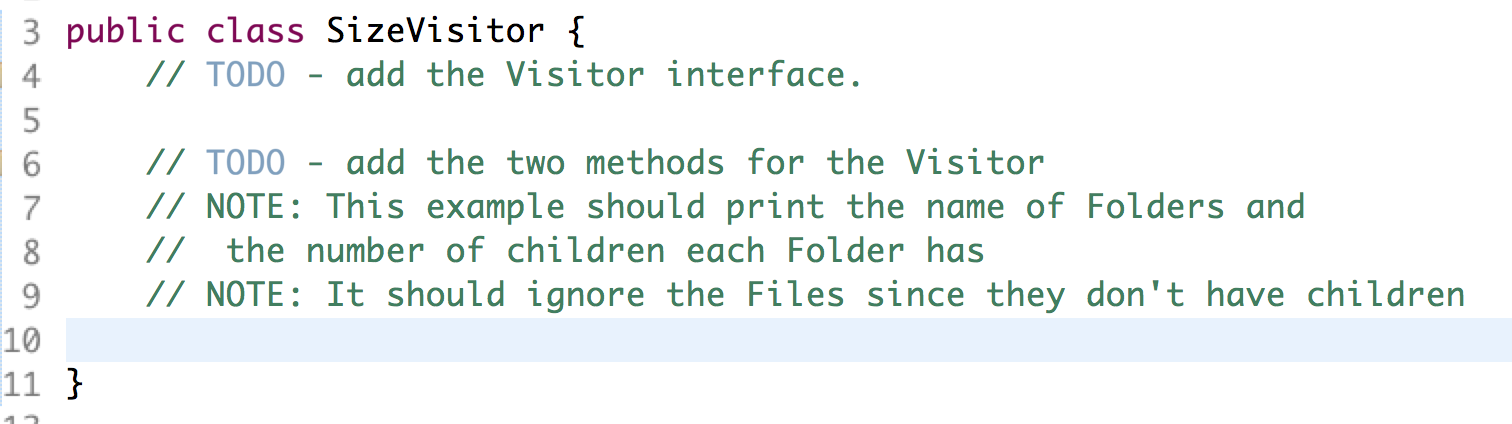
1. This will call the Visitor::visit( File f ) overloaded method.
2. Similarly, in the Folder class shown below:



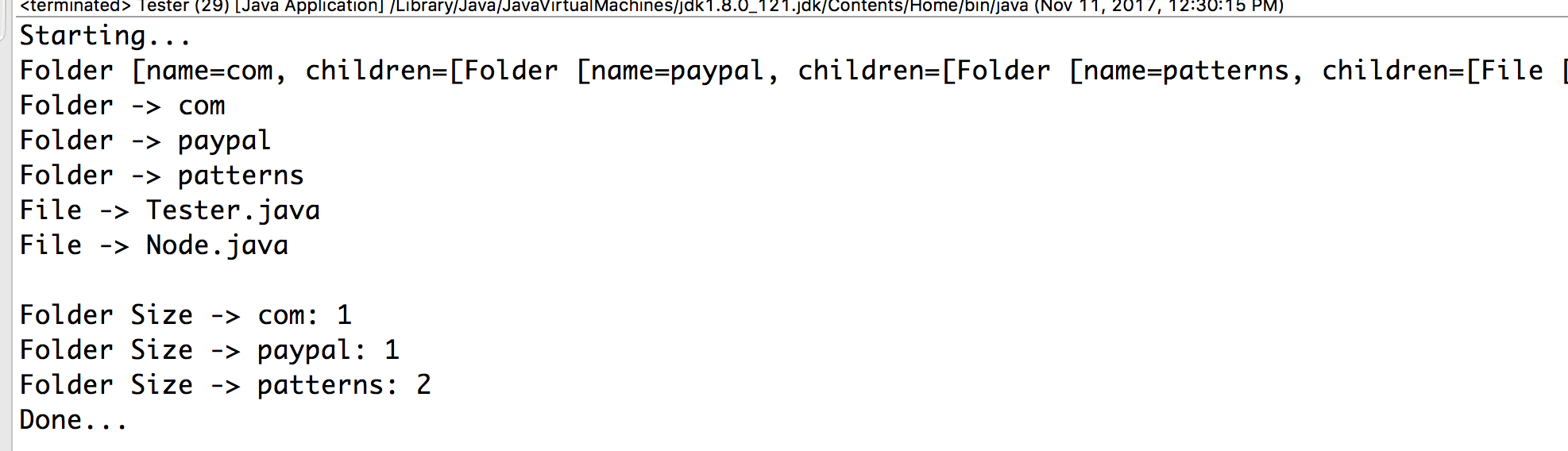
1. In lines 17-23, the Folder will first call Visitor::visit( Folder f ), then call accept() for each of its children.
2. Now that we have created the structure, let’s create some Visitors.
3. The first Visitor, NameVisitor.java, is shown below:



1. The above prints the name of each component. When invoked at the top of the tree, it will print the name of each Folder and File.
2. Our task is to create a Visitor which displays the number of children in each Folder.
3. Edit the SizeVisitor.java as shown:

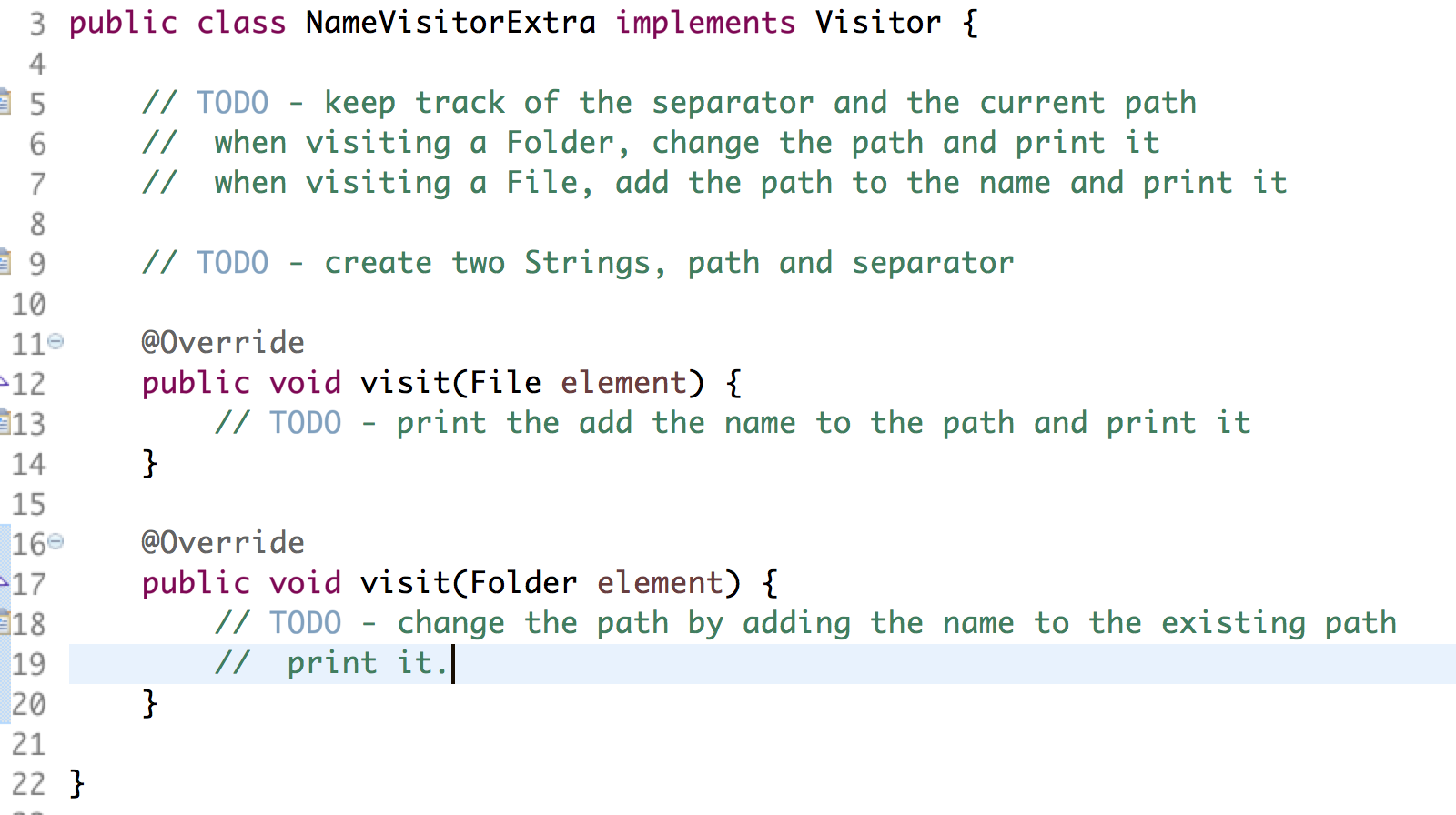


1. Uncomment the lines in Tester.java and execute the file as a Java application. You should see:

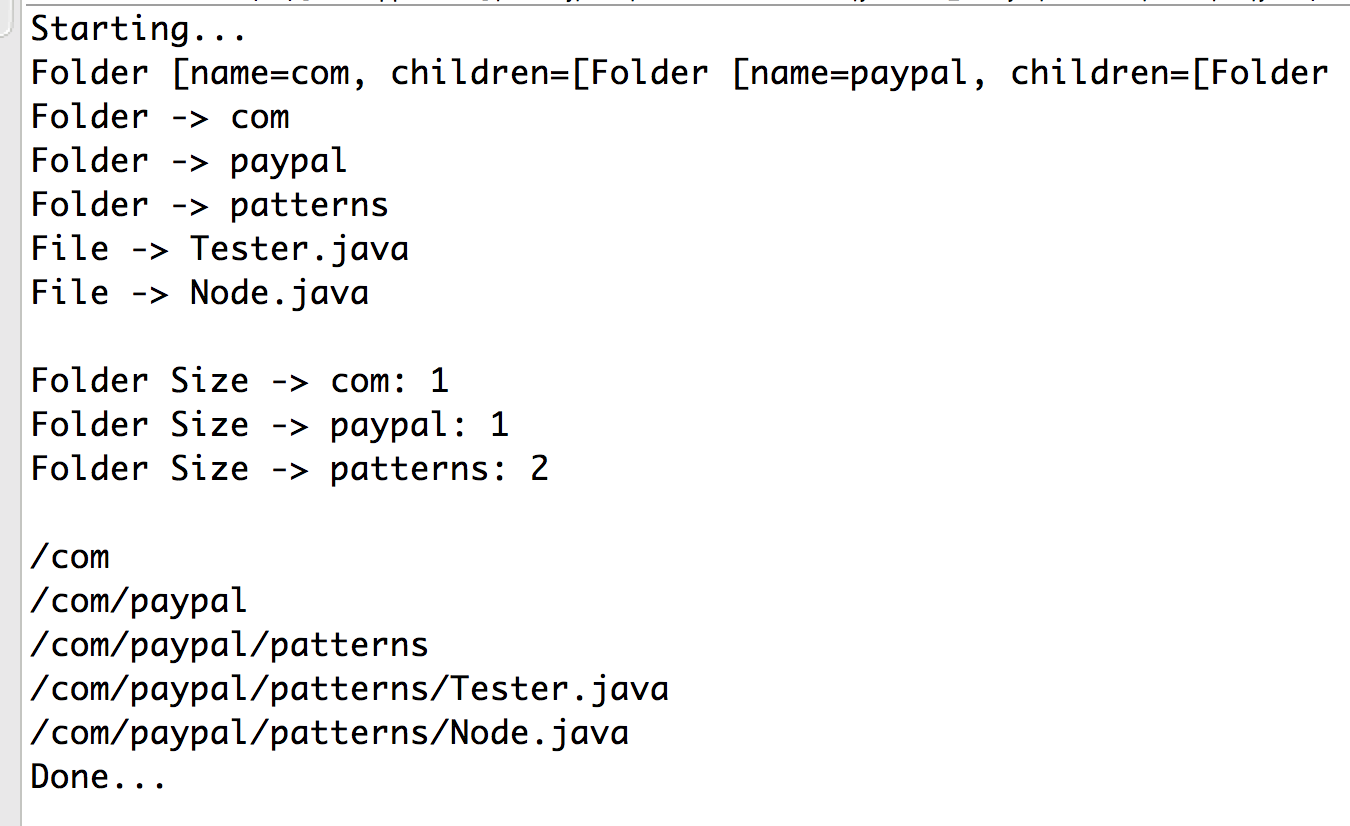


# Extra Credit

1. This output of the names is really ugly. Let’s add a Visitor that keeps track of the path and displays a better format.
2. Edit the file, NameVisitorExtra.java, as shown below:



1. Edit the Tester.java and invoke the new Visitor. The output should be:



Congratulations. You have completed this lab.