Paul Kummer

Dr. Hanku Lee

Compilers CSIS 455-01

09/11/2021

Lab 02: Examining Visitors Continued

Purpose of Lab

The purpose of this lab is to help clarify the concepts of the previous lab. It shows how a visitor is created and can have information about the visitor retrieved. More importantly, this lab is suppose to demonstrate how using visitors an abstract syntax tree can be created and traversed by calling visit and accept methods and also using interfaces for visitors to interact with other objects.

Example Code

**Executable Code**

Runs the java code that uses visitors to make a computer with parts. The details of the new computer are hidden in the visitor and are accessed with “computer.accept”.

VisitorPatternDemo.java

**Text

Description automatically generated**

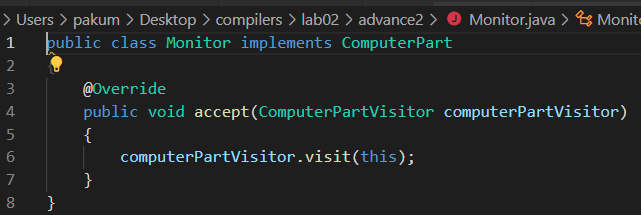
**Computer Components**

The code for the following classes is for making components of a computer. The computer is made up of a mouse, keyboard, monitor, and desktop. The computer is the root of the abstract syntax tree.

Mouse.java

Text

Description automatically generated

Keyboard.java

Monitor.java

Text

Description automatically generated

Desktop.java

Text

Description automatically generated

**Desktop Components**

Within the computer there is the Desktop. This desktop contains its own components of a motherboard and a power supply.

MotherBoard.java

Text

Description automatically generated

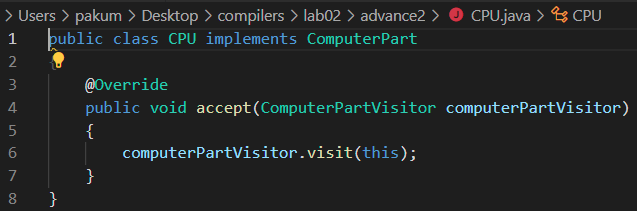
PowerSupply.java

Text

Description automatically generated

**MotherBoard Components**

Creating more dept to the abstract syntax tree, the motherboard has its own components. Each component of the motherboard is also a component of a desktop and computer. The components are a cpu, hdd, ssd, gpu, and mem.

CPU.java

HDD.java

Text

Description automatically generated

SSD.java

Text

Description automatically generated

GraphicCard.java

Text

Description automatically generated

Memory.java

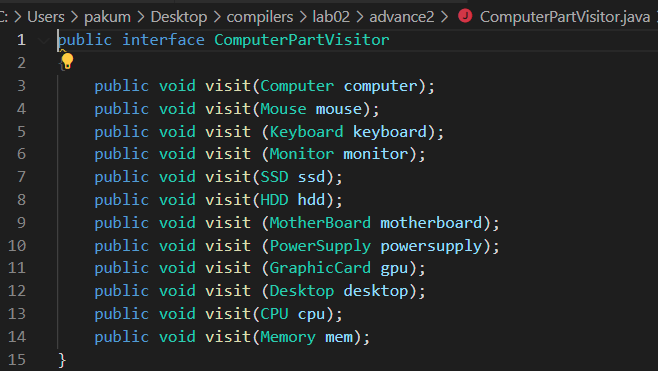
Text

Description automatically generated

**Using Visitors**

For the abstract syntax tree to work, each of the components is encapsulated as a visitor. This visitor is passed into other objects where the methods of visit and accept can be called to get information from the visitor. To make this work, an interface with the visitor must be used for different objects.

ComputerPartVisitor.java

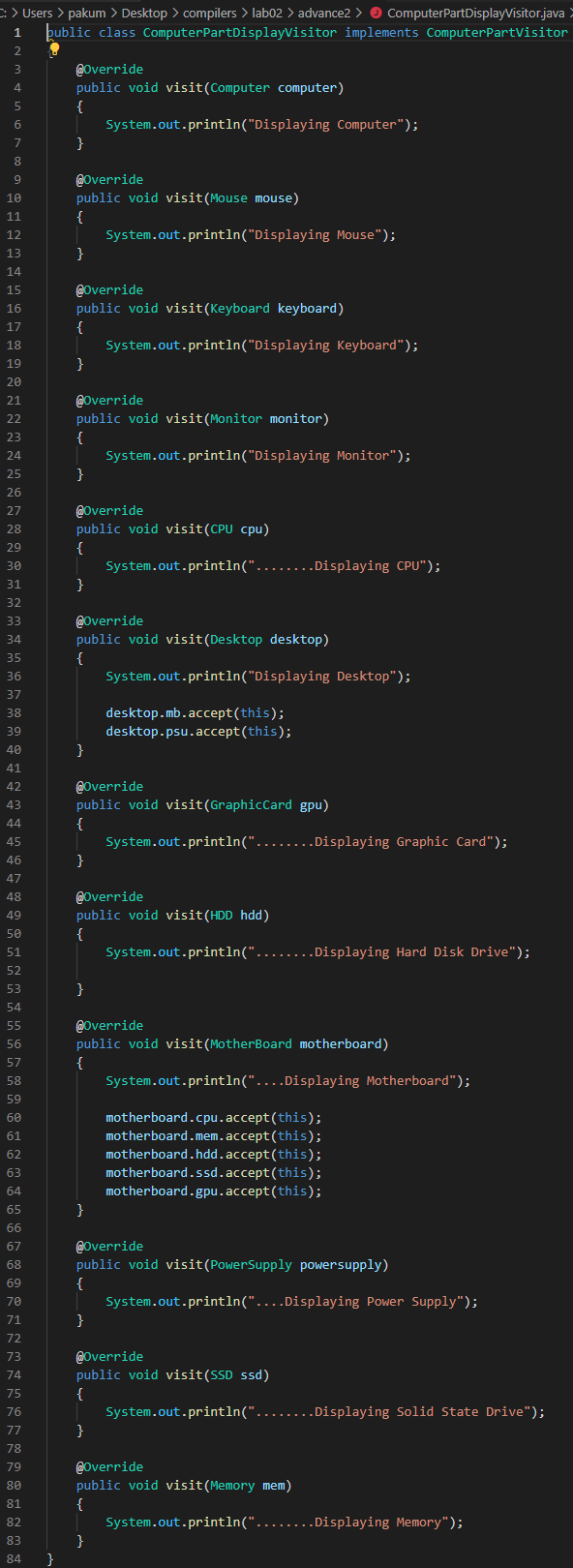
****

ComputerPart.java

Graphical user interface, text, website

Description automatically generated

ComputerPartDisplayVisitor.java



**Code Execution**

After the java code is compiled, the VisitorPatternDemo can be run. When it runs it will show all the parts of the new computer and roughly show the depth of the AST.

**Text

Description automatically generated**

Conclusion

This lab helped solidify some of the concepts of the first lab. It showed how depth can be created in an AST by creating new objects within another object. The root will have no knowledge of what is in the objects farther down the tree, but it can access the parts by visiting and accepting visitors. This helps keep objects contained to whatever is category they fit into. Additionally, the functionality of a visitor is hidden within the object that is currently the visitor. This allows better modularity.

When the code is run, the depth of the tree is shown by visually showing if a component is part of the computer or if it is a component of a component. This also shows that a visitor can traverse a tree to reach all the nodes.