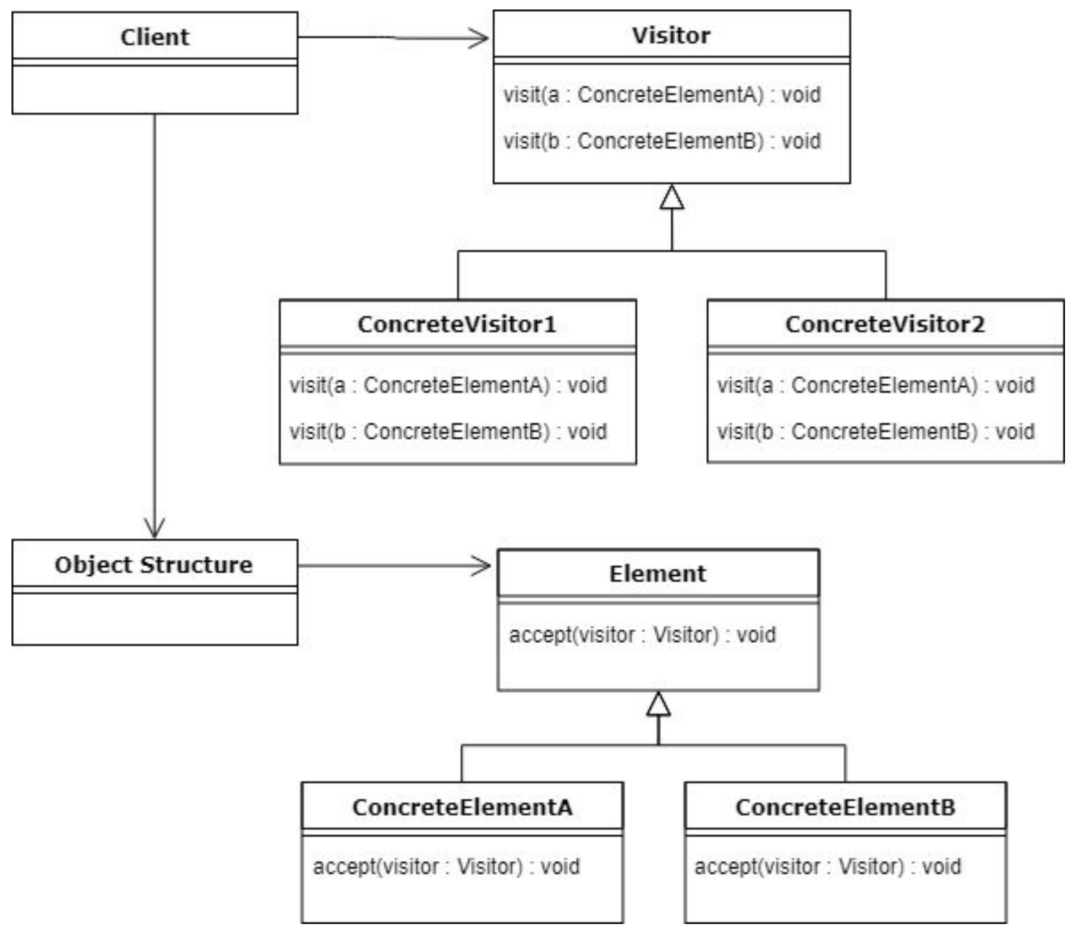
Visitor Pattern

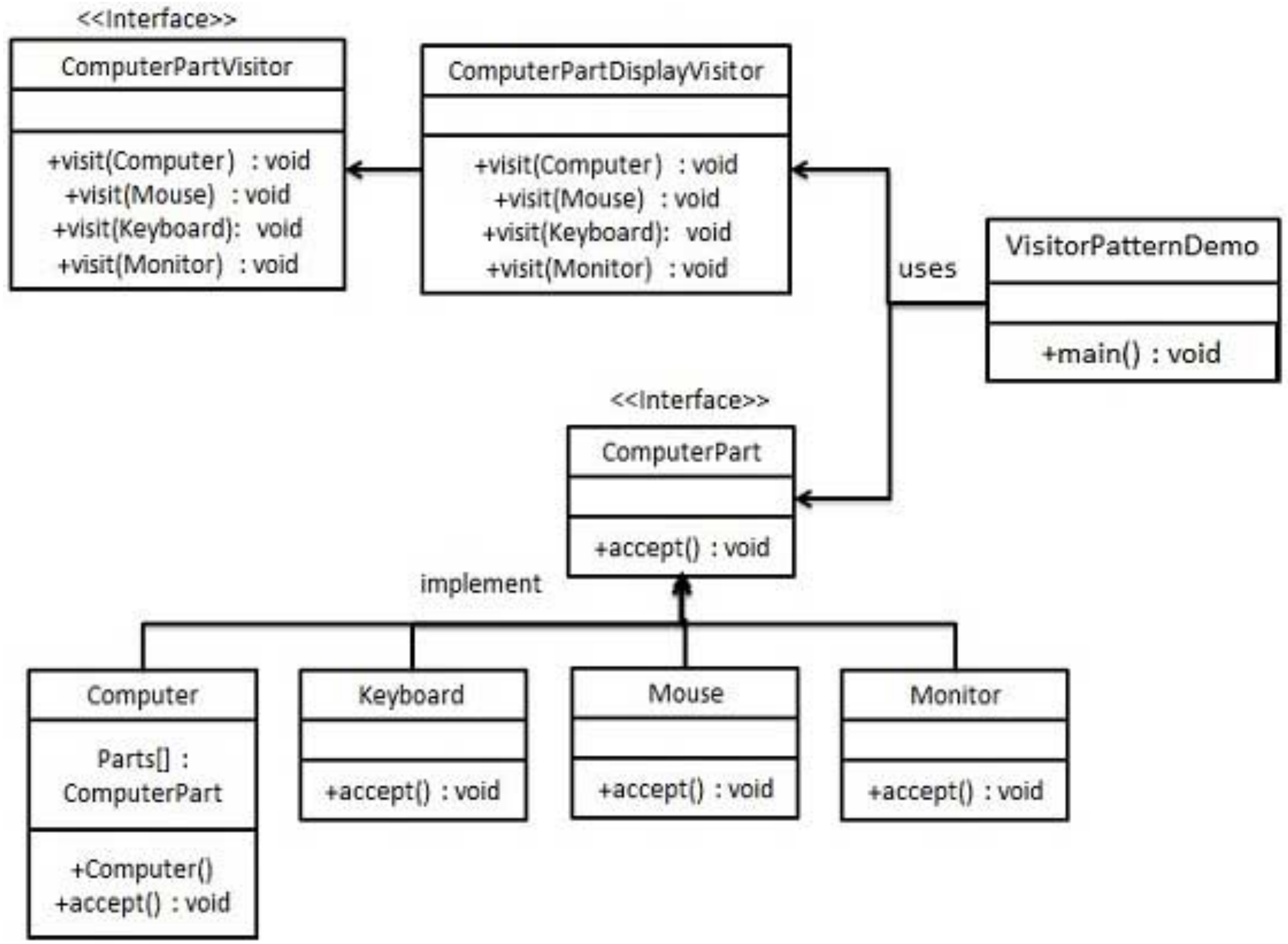
Visitor Pattern

- Use a visitor class which changes the executing algorithm of an element class.
- Place the new behavior into a separate class called *visitor*, instead of trying to integrate it into existing classes.
- This pattern comes under behavior pattern category.

Visitor Pattern

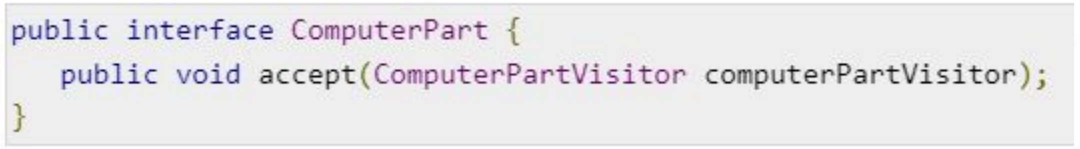

Client
Visitor
visit(a : ConcreteElementA) : void
visit(b : ConcreteElementB) : void
ConcreteVisitor1
visit(a : ConcreteElementA) : void
visit(b : ConcreteElementB) : void
ConcreteVisitor2
visit(a : ConcreteElementA) : void
visit(b : ConcreteElementB) : void
Object Structure
Element
accept(visitor : Visitor) : void
ConcreteElementA
accept(visitor : Visitor) : void
ConcreteElementB
accept(visitor : Visitor) : void

Visitor Pattern



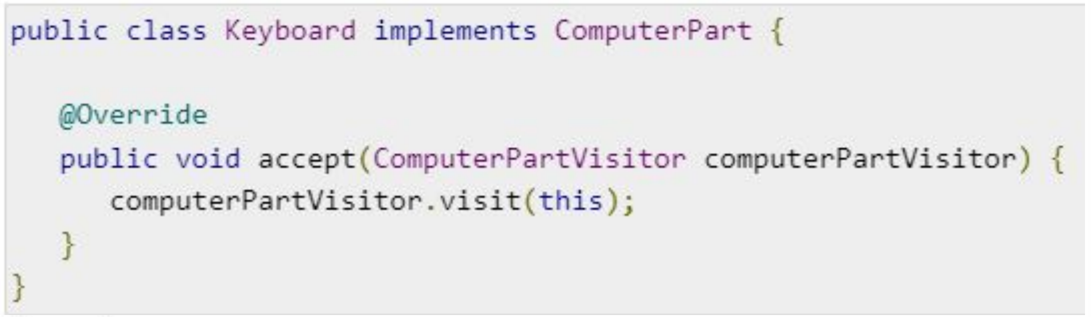
Visitor Pattern

ComputerPart.java



```java
public interface ComputerPart {
   public void accept(ComputerPartVisitor computerPartVisitor);
}
```

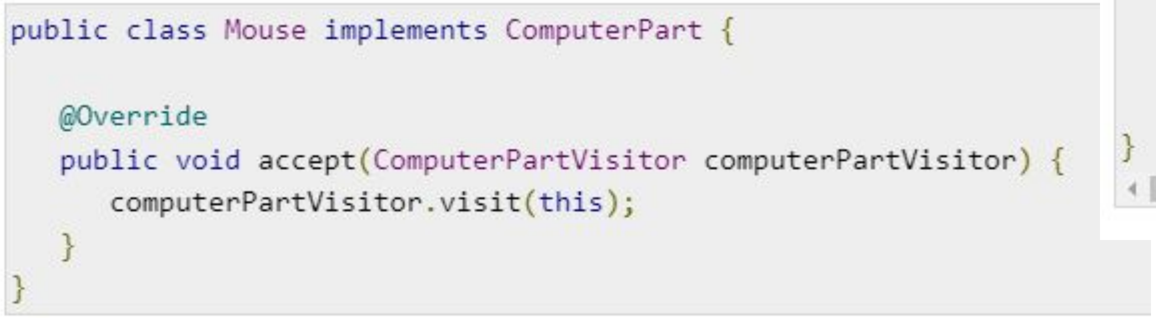
Keyboard.java



```java
public class Keyboard implements ComputerPart {

   @Override
   public void accept(ComputerPartVisitor computerPartVisitor) {
      computerPartVisitor.visit(this);
   }
}
```

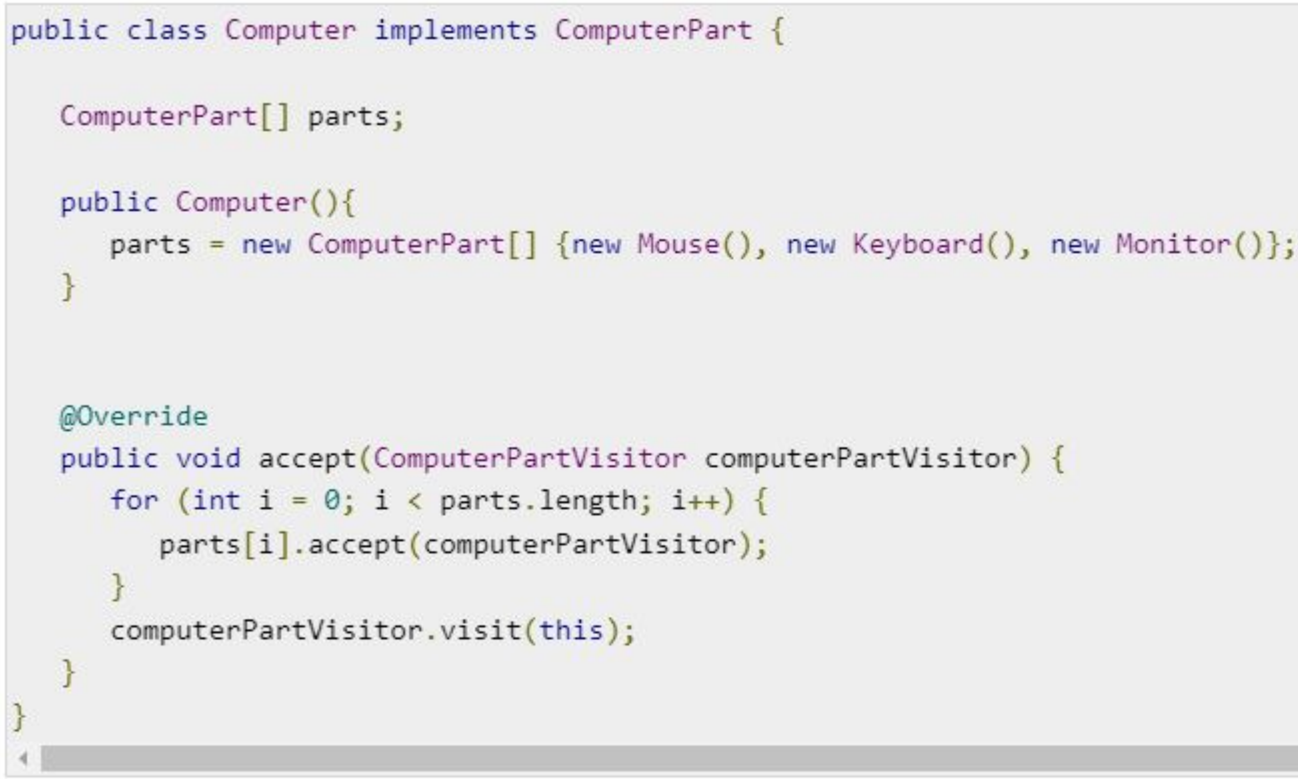
Mouse.java



```java
public class Mouse implements ComputerPart {

   @Override
   public void accept(ComputerPartVisitor computerPartVisitor) {
      computerPartVisitor.visit(this);
   }
}
```

Computer.java



```java
public class Computer implements ComputerPart {

   ComputerPart[] parts;

   public Computer(){
      parts = new ComputerPart[] {new Mouse(), new Keyboard(), new Monitor()};
   }


   @Override
   public void accept(ComputerPartVisitor computerPartVisitor) {
      for (int i = 0; i < parts.length; i++) {
         parts[i].accept(computerPartVisitor);
      }
      computerPartVisitor.visit(this);
   }
}
```

Visitor Pattern

ComputerPartVisitor.java

```java
public interface ComputerPartVisitor {
	public void visit(Computer computer);
	public void visit(Mouse mouse);
	public void visit(Keyboard keyboard);
	public void visit(Monitor monitor);
```

ComputerPartDisplayVisitor.java

```java
public class ComputerPartDisplayVisitor implements ComputerPartVisitor {

   @Override
   public void visit(Computer computer) {
      System.out.println("Displaying Computer.");
   }

   @Override
   public void visit(Mouse mouse) {
      System.out.println("Displaying Mouse.");
   }

   @Override
   public void visit(Keyboard keyboard) {
      System.out.println("Displaying Keyboard.");
   }

   @Override
   public void visit(Monitor monitor) {
      System.out.println("Displaying Monitor.");
   }
}
```

VisitorPatternDemo.java

```java
public class VisitorPatternDemo {
   public static void main(String[] args) {

      ComputerPart computer = new Computer();
      computer.accept(new ComputerPartDisplayVisitor());
   }
}
```

Visitor Pattern

- Difference between strategy and Visitor?
- Learn the concept “Double Dispatch”.
- How visitor pattern is related to Composite?