

Chain of Responsibility Pattern

```
package com.journaldev.design.chainofresponsibility;

import java.util.Scanner;

public class ATMDispenseChain {
    3 usages
    private DispenseChain c1;
    1 usage
    public ATMDispenseChain() {
        this.c1 = new
            Dollar500Dispenser();

        DispenseChain c2 = new Dollar200Dispenser();
        DispenseChain c3 = new Dollar100Dispenser();

        c1.setNextChain(c2);
        c2.setNextChain(c3);
    }
}
```

Run: ATMDispenseChain

Enter amount to dispense
100
Dispensing 10 50\$ note
Dispensing 1 20\$ note
Dispensing 1 10\$ note
Enter amount to dispense
120
Dispensing 2 50\$ note
Enter amount to dispense
10
Dispensing 2 50\$ note
Dispensing 1 20\$ note
Enter amount to dispense
Amount should be in multiple of 10s.

Command Pattern

```
public class CommandPatternDemo {
    public static void main(String[] args) {
        Stock abcStock = new Stock();

        BuyStock buyStockOrder = new BuyStock(abcStock);
        SellStock sellStockOrder = new SellStock(abcStock);

        Broker broker = new Broker();
        broker.takeOrder(buyStockOrder);
        broker.takeOrder(sellStockOrder);

        broker.placeOrders();
    }
}
```

Run: CommandPatternDemo

Stock [Name: ABC, Quantity: 10] bought
Stock [Name: ABC, Quantity: 10] sold
Process finished with exit code 0

Iterator Pattern

```
1 public class IteratorPatternDemo {
2
3     public static void main(String[] args) {
4         NameRepository namesRepository = new NameRepository();
5
6         for (Iterator iter = namesRepository.getIterator(); iter.hasNext(); ) {
7             String name = (String) iter.next();
8             System.out.println("Name : " + name);
9         }
10    }
11 }
12
```

Run: IteratorPatternDemo

Process finished with exit code 0

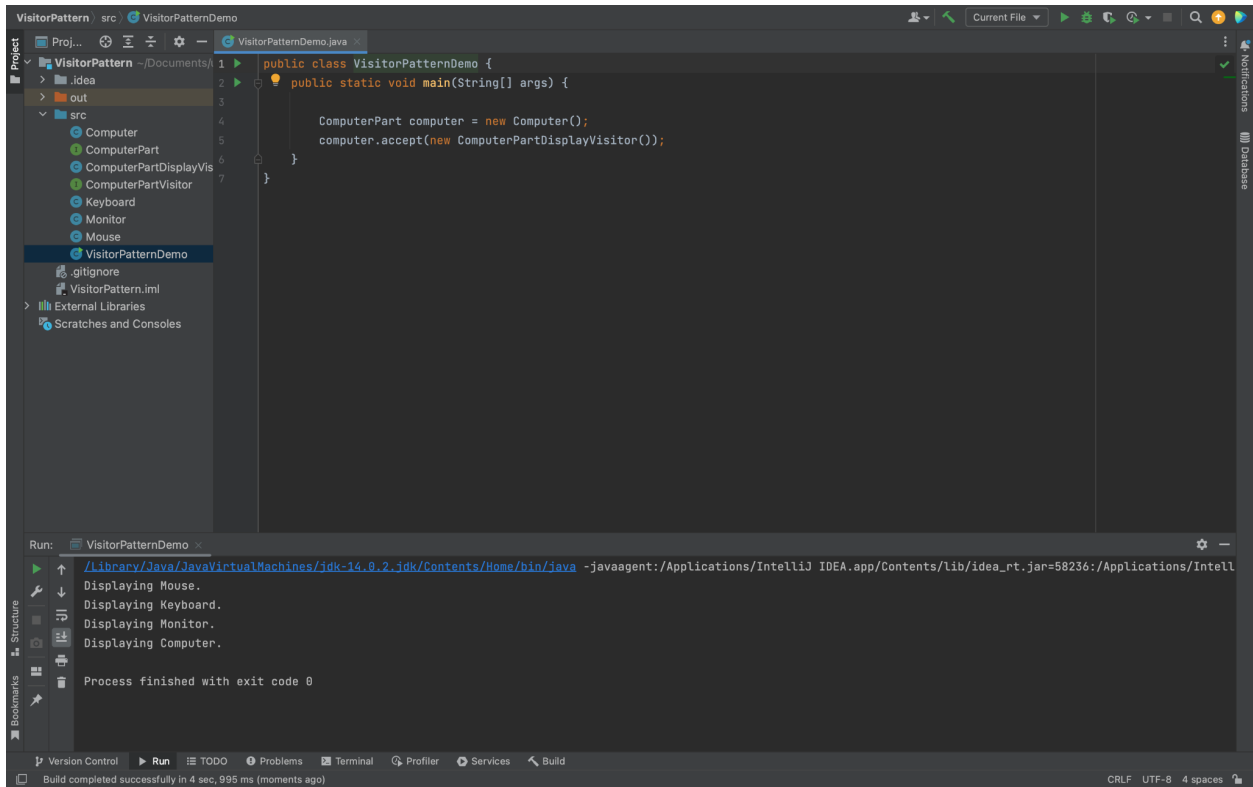
Observer Pattern

```
1 public class ObserverPatternDemo {
2     public static void main(String[] args) {
3         Subject subject = new Subject();
4
5         new HexaObserver(subject);
6         new OctalObserver(subject);
7         new BinaryObserver(subject);
8
9         System.out.println("First state change: 15");
10        subject.setState(15);
11        System.out.println("Second state change: 10");
12        subject.setState(10);
13    }
14 }
```

Run: ObserverPatternDemo

Process finished with exit code 0

Visitor Pattern



```
1 public class VisitorPatternDemo {
2     public static void main(String[] args) {
3
4         ComputerPart computer = new Computer();
5         computer.accept(new ComputerPartDisplayVisitor());
6     }
7 }
```

Run: VisitorPatternDemo

```
/Library/Java/JavaVirtualMachines/jdk-14.0.2.jdk/Contents/Home/bin/java -javaagent:/Applications/IntelliJ IDEA.app/Contents/Lib/idea_rt.jar=58236:/Applications/IntelliJ
Displaying Mouse.
Displaying Keyboard.
Displaying Monitor.
Displaying Computer.
Process finished with exit code 0
```

Build completed successfully in 4 sec, 995 ms (moments ago)