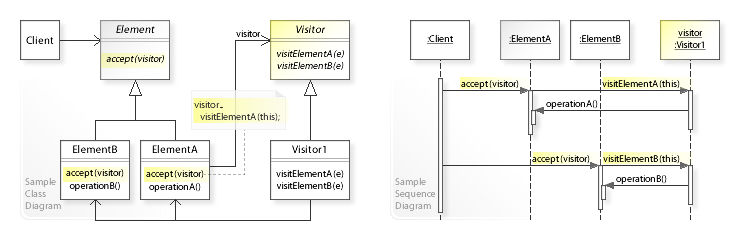
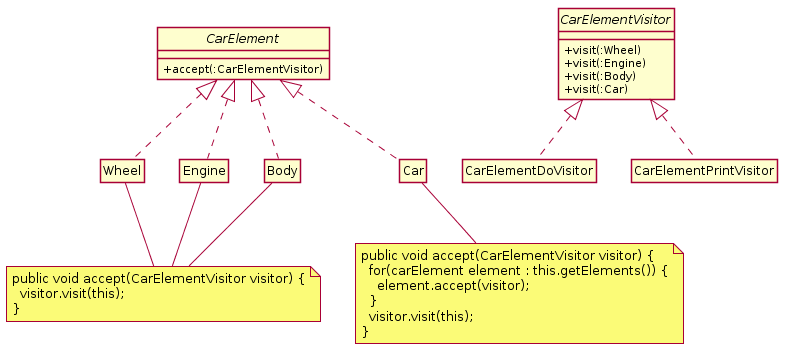
Visitor Pattern

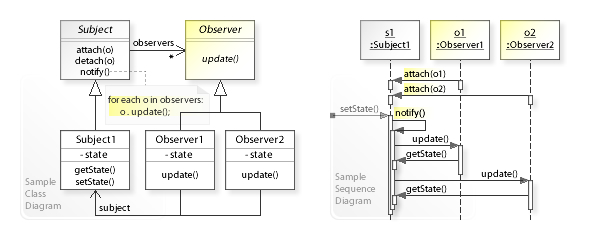
In [object-oriented programming](https://en.wikipedia.org/wiki/Object-oriented_programming) and [software engineering](https://en.wikipedia.org/wiki/Software_engineering), the visitor [design pattern](https://en.wikipedia.org/wiki/Software_design_pattern) is a way of separating an [algorithm](https://en.wikipedia.org/wiki/Algorithm) from an [object](https://en.wikipedia.org/wiki/Object_(computer_science)) structure on which it operates. A practical result of this separation is the ability to add new operations to existing object structures without modifying the structures. It is one way to follow the [open/closed principle](https://en.wikipedia.org/wiki/Open/closed_principle).





Observer Pattern

The observer pattern is a [software design pattern](https://en.wikipedia.org/wiki/Design_pattern_(computer_science)) in which an [object](https://en.wikipedia.org/wiki/Object_(computer_science)#Objects_in_object-oriented_programming), named the subject, maintains a list of its dependents, called observers, and notifies them automatically of any state changes, usually by calling one of their [methods](https://en.wikipedia.org/wiki/Method_(computer_science)).



import java.util.List;

import java.util.ArrayList;

import java.util.Scanner;

class EventSource {

public interface Observer {

void update(String event);

}

private final List<Observer> observers = new ArrayList<>();

private void notifyObservers(String event) {

observers.forEach(observer -> observer.update(event));

}

public void addObserver(Observer observer) {

observers.add(observer);

}

public void scanSystemIn() {

Scanner scanner = new Scanner(System.in);

while (scanner.hasNextLine()) {

String line = scanner.nextLine();

notifyObservers(line);

}

}

}

public class ObserverDemo {

public static void main(String[] args) {

System.out.println("Enter Text: ");

EventSource eventSource = new EventSource();

eventSource.addObserver(event -> {

System.out.println("Received response: " + event);

});

eventSource.scanSystemIn();

}

}