CSC 7203 : Advanced Object Oriented Development

J Paul Gibson, D311

paul.gibson@telecom-sudparis.eu

http://www-public.it-sudparis.eu/~gibson/Teaching/CSC7203/

Visitor Pattern

…/~gibson/Teaching/CSC77203-AdvancedOO-L3-Visitor.pdf

ALAIN TERZIAN
JEAN
RENO
LEMERCIER
LES VISITEURS
ILS NE SONT PAS NÉS D'HIER !
1123
1993
POIRÉ
CHRISTIAN CLAVIER . JEAN RENO . VALERIE LEMERCIER
LES VISITEURS
JEAN-MARIE POIRE
MARIE-ANNE CHAZEL

The Visitor Pattern

See - http://sourcemaking.com/design_patterns/visitor

- **Intent**
  - Represent an operation to be performed on the elements of an object structure. Visitor lets you define a new operation without changing the classes of the elements on which it operates.
  - The classic technique for recovering lost type information.
  - Do the right thing based on the type of two objects.
  - **Double dispatch**

- **Problem**

  Many distinct and unrelated operations need to be performed on node objects in a heterogeneous aggregate structure. You want to avoid "polluting" the node classes with these operations. And, you don't want to have to query the type of each node and cast the pointer to the correct type before performing the desired operation.

The Visitor Pattern

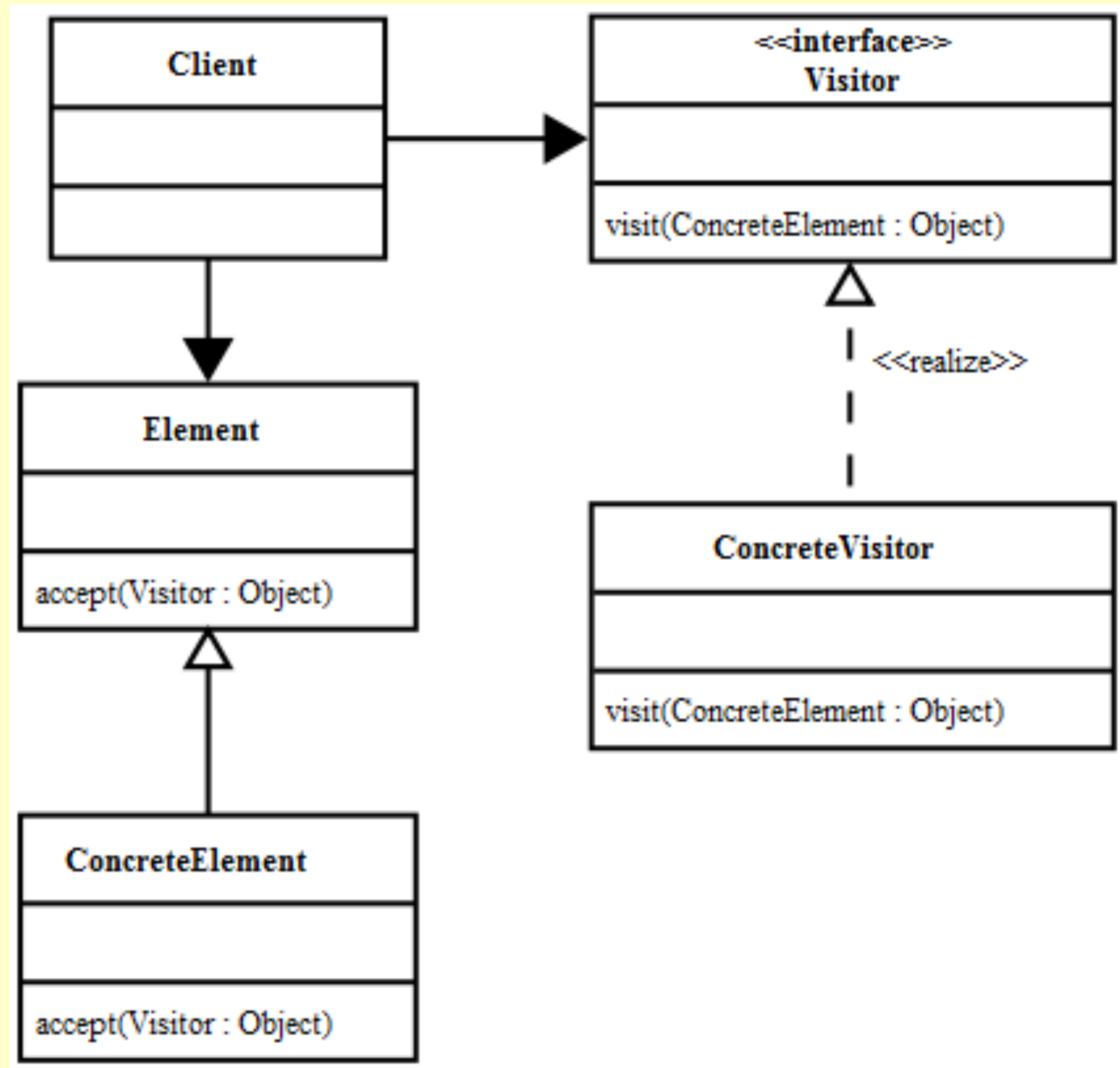
See - http://sourcemaking.com/design_patterns/visitor

Relation to other patterns

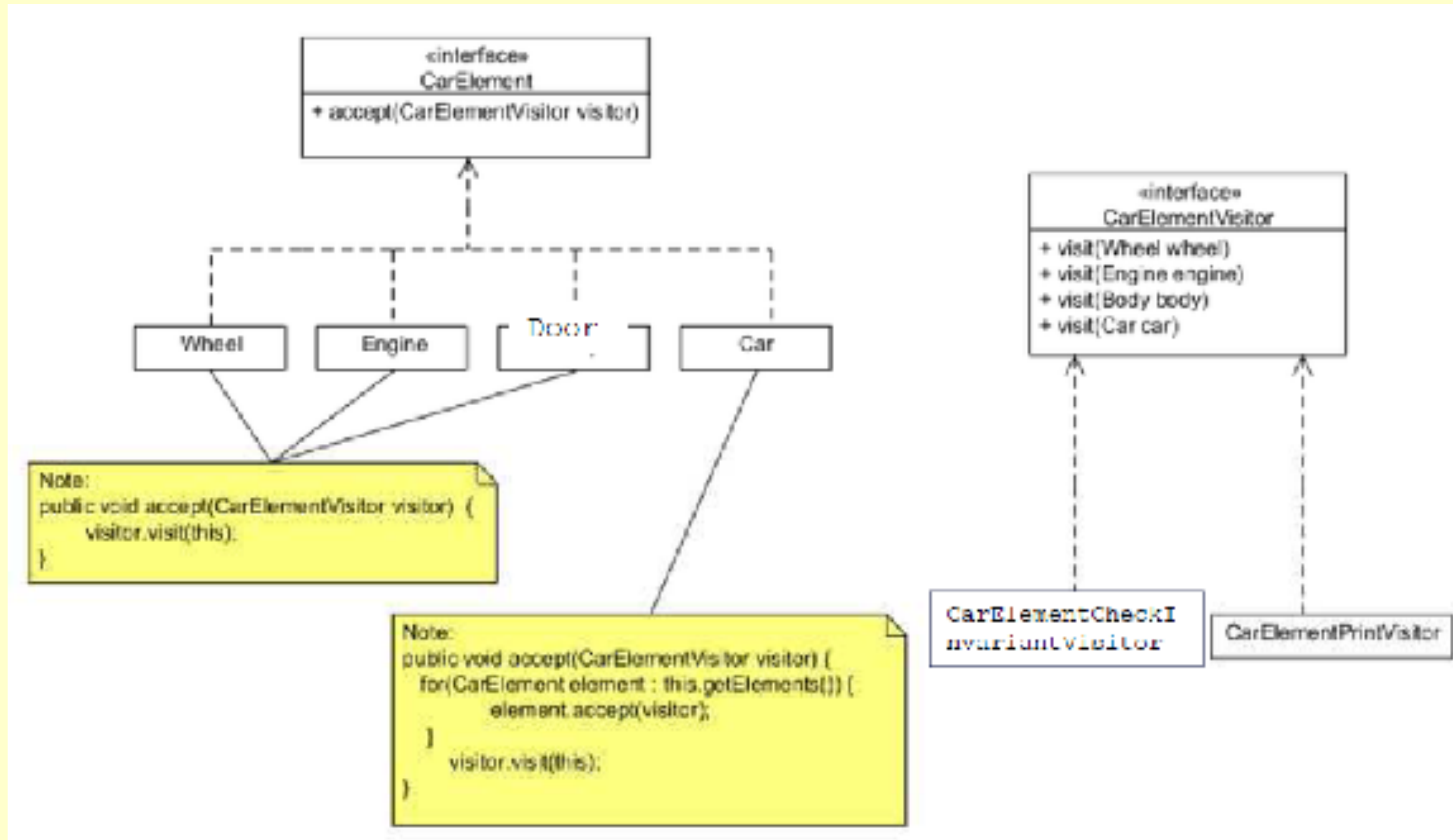
•The abstract syntax tree of Interpreter is a Composite (therefore Iterator and Visitor are also applicable).
•Iterator can traverse a Composite. Visitor can apply an operation over a Composite.
•The Visitor pattern is like a more powerful Command pattern because the visitor may initiate whatever is appropriate for the kind of object it encounters.
•The Visitor pattern is the classic technique for recovering lost type information without resorting to dynamic casts.

NOTE: Visitor is not good for the situation where "visited" classes are not stable. Every time a new Composite hierarchy derived class is added, every Visitor derived class must be amended.

UML Class Diagram Visitor

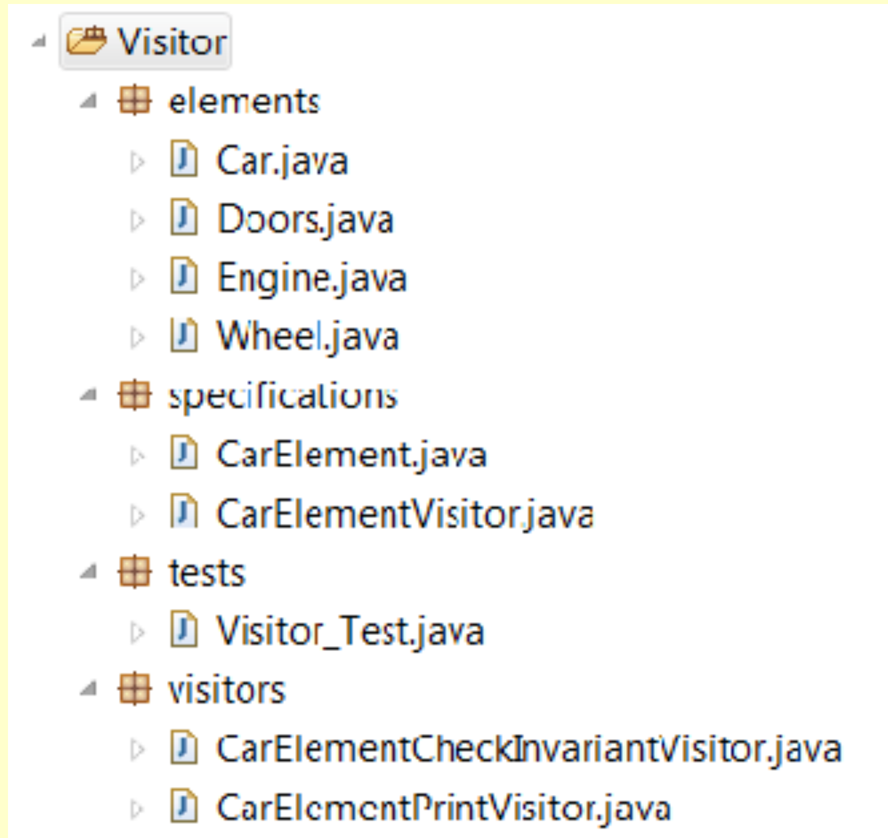


UML Class Diagram Visitor Example - Car



Download the visitor example from

~gibson/Teaching/CSC7203/Code/Visitor.zip



An example *visitable*

```
class Car implements CarElement{
    CarElement[] elements;

    public CarElement[] getElements() {
        return elements.clone(); // Return a copy of the array of references.
    }

    public Car() {
        this.elements = new CarElement[]
          { new Wheel("front left"), new Wheel("front right"),
            new Wheel("back left") , new Wheel("back right"),
            new Doors(), new Engine(8) };
    }

    public String toString(){ return "\n  *** A Car *** \n"; }

    public boolean invariant (){ return (elements!=null && elements.length>0);}

    public void accept(CarElementVisitor visitor) {
        visitor.visit(this);
        for(CarElement element : this.getElements()) {
            element.accept(visitor);
        }

    }
}
```

An example visitor

```
class CarElementPrintVisitor implements CarElementVisitor {

    public void visit(Wheel wheel) {
        System.out.println(wheel);
    }

    public void visit(Engine engine) {
        System.out.println(engine);
    }

    public void visit(Doors doors) {
        System.out.println(doors);
    }

    public void visit(Car car) {
        System.out.println(car);
    }
}
```

Testing the example

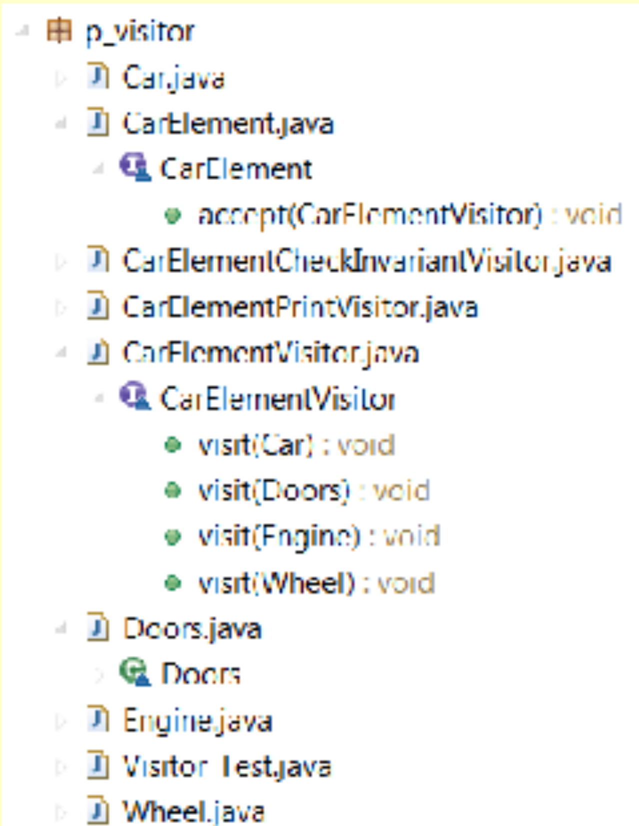
```
public class Visitor_Test {
    static public void main(String[] args){
        Car car = new Car();
        car.accept(new CarElementPrintVisitor());
        car.accept(new CarElementCheckInvariantVisitor());
    }
}
```

```
 *** A Car ***

front left is not turning
front right is not turning
back left is not turning
back right is not turning
LeftDoorLocked is true and RightDoorLocked is true
Engine speed is 0 / 8
```

QUESTION: what is **`CarElementCheckInvariantVisitor`** doing?

UML Class Diagram Visitor Example - Car



TO DO: Add a visitor (`breakInvariantVisitor`) which changes the state of each component of the car so that their invariants are broken.

Update the test class to check that this visitor is working as required