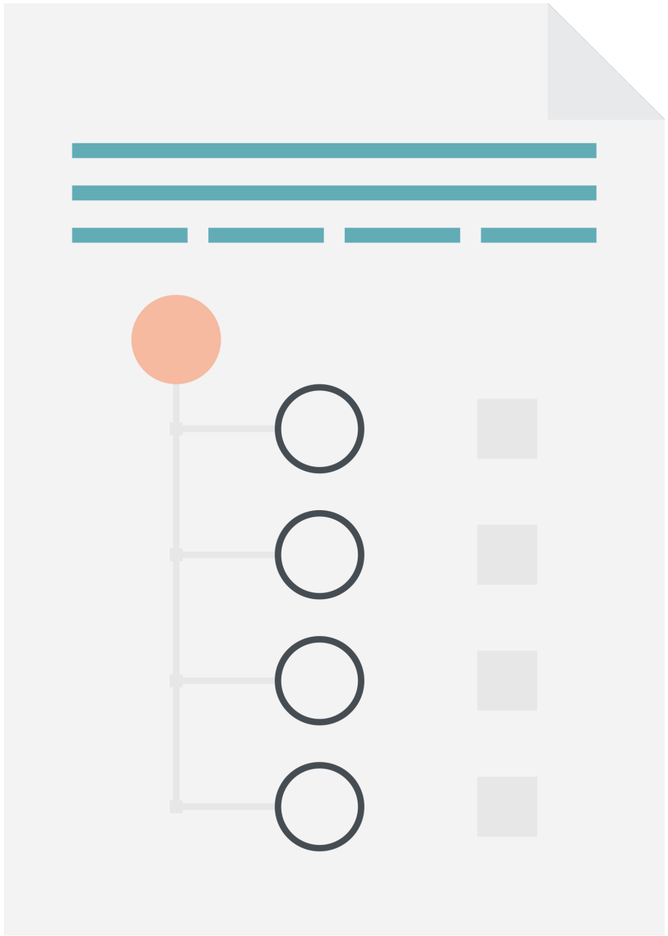
# Visitor Pattern


Bryan Hansen

twitter: bh5k | http://www.linkedin.com/in/hansenbryan

# Concepts

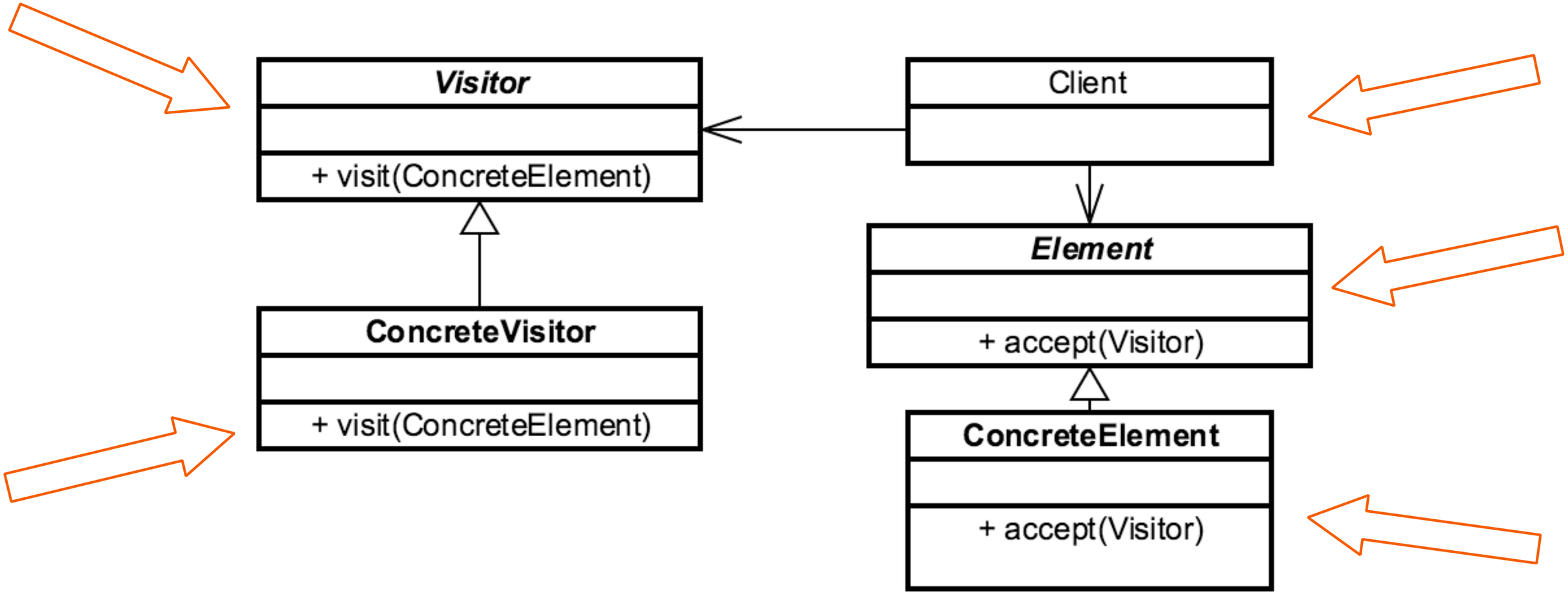
- Separate Algorithm from Object
- Adding new features
- Maintain Open/Closed principle
- Visitor changes
- Examples:
  - java.lang.model.element.Element
  - java.lang.model.element.ElementVisitor



# Design

- Interface based
- Visitor and Element
- Elements have visit method
- Visitor knows every Element
- Visitor, ConcreteVisitor, Element, ConcreteElement

# UML


Visitor
+ visit(ConcreteElement)
Client
Element
+ accept(Visitor)
ConcreteVisitor
+ visit(ConcreteElement)
ConcreteElement
+ accept(Visitor)

# Everyday Example - Order

```
public class VisitorDemo {

   public static void main(String[] args) {
      PartsOrder order = new PartsOrder();
      order.addPart(new Wheel());
      order.addPart(new Fender());
      order.addPart(new Oil());

      order.accept(new AtvPartsShippingVisitor());
   }
}
```

# Exercise Visitor

Code Without

Visitor, Element, ConcreteVisitor, ConcreteElement

# Pitfalls

- Plan for adaptability
- Indirection somewhat confusing
- Adapter pattern

# Contrast

## Visitor

- Interfaced based
- Externalized changes
- Multiple visitors

## Iterator

- Interfaced based / Anonymous
- Encapsulates
- Singular

# Visitor Summary

- Expect changes
- Minor complexity
- Externalizes change