CS 635 Advanced Object-Oriented Design & Programming
Spring Semester, 2010
Doc 18 Chain of Responsibility & Builder
8 Apr 2010

References

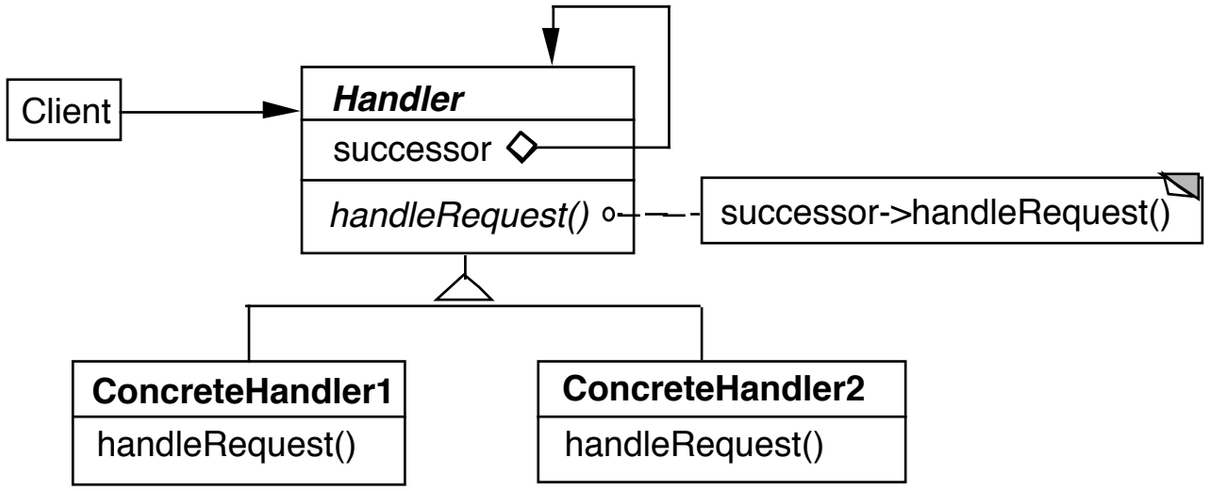
Wikipedia, http://en.wikipedia.org/wiki/Chain_of_responsibility_pattern

Design Patterns: Elements of Reusable Object-Oriented Software, Gamma, Helm, Johnson, Vlissides, Addison-Wesley, 1995, pp. 97-106

The Design Patterns Smalltalk Companion, Alpert, Brown, Woolf, 1998, pp. 47-62

SAX
    http://www.saxproject.org/
    http://en.wikipedia.org/wiki/Simple_API_for_XML

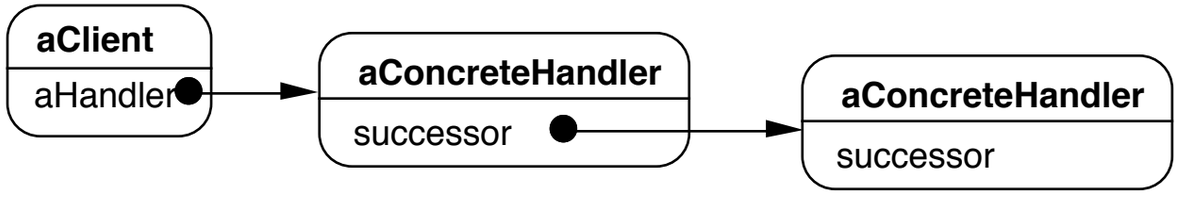
Chain of Responsibility



Dynamically create chain of handlers

Multiple handlers may be able to handle a request

Only one handler actually handles the request



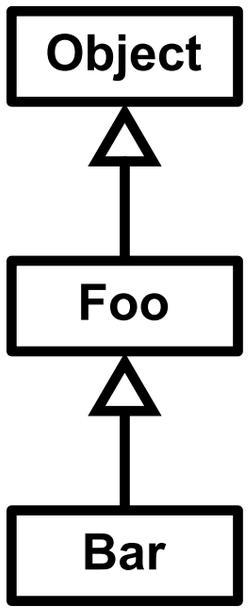
Consequences

Reduced coupling

Added flexibility in assigning responsibilities to objects

Not guaranteed that request will be handled

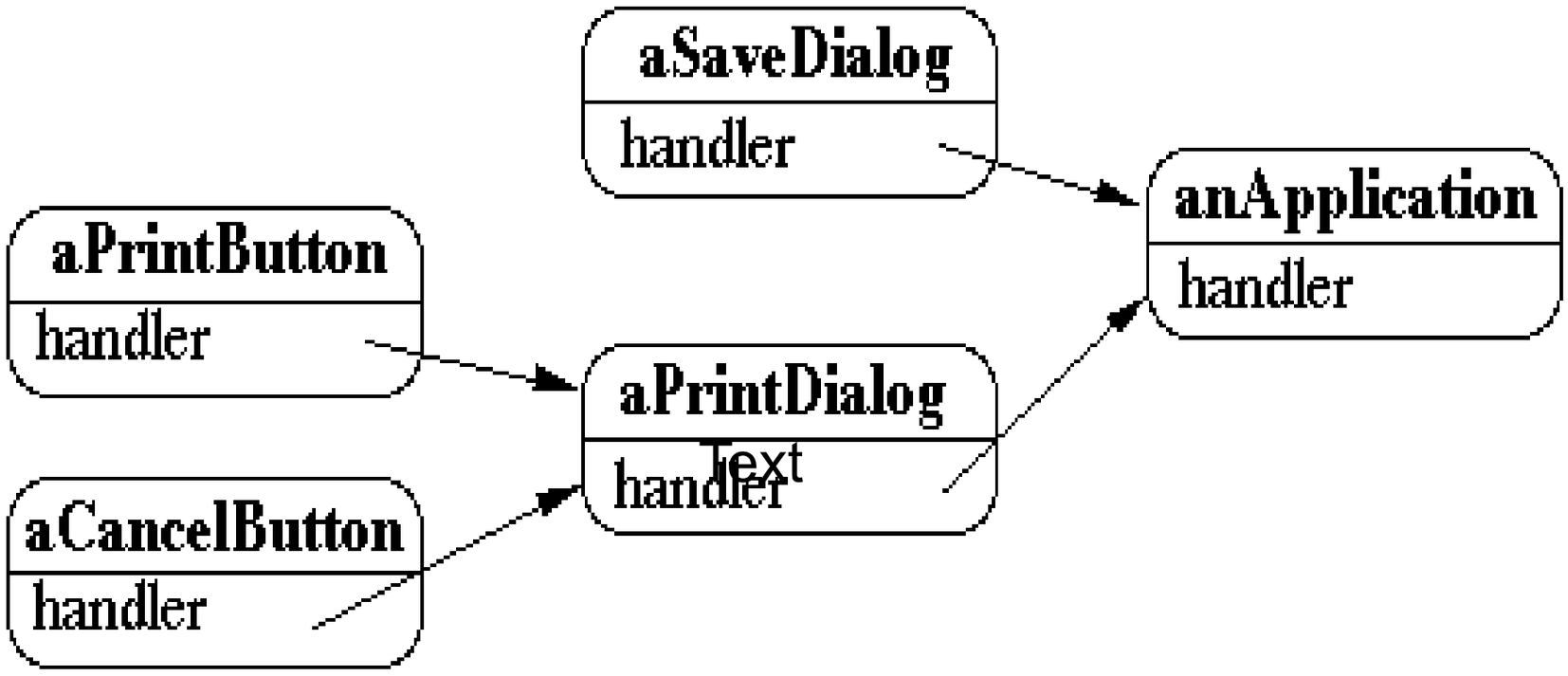
Finding Methods




```
test = new Bar();
test.toString();
```

Context Help System

User clicks on component for help



Tree of handlers

From specific to general

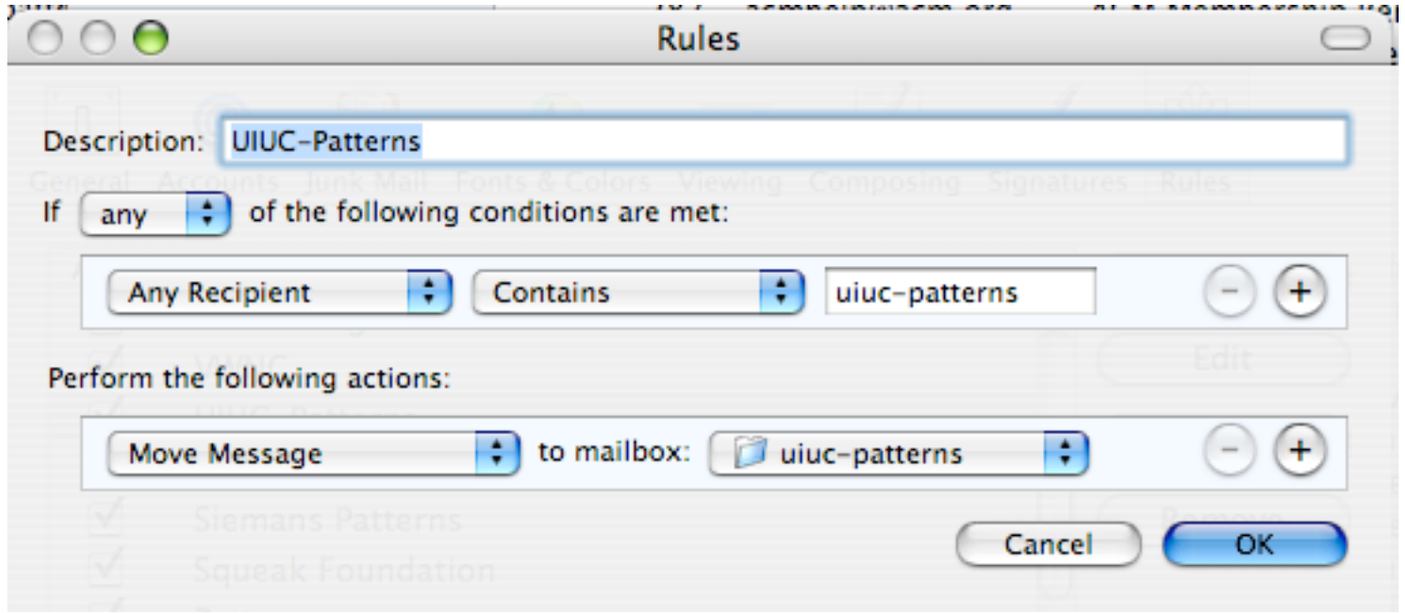
Email Filters in Mail Client

User creates a set of rules

- delete
- move
- modify

Chain the rules

First rule that applies handles the mail



Other Examples

Java 1.0 AWT action(Event)

http://wiki.cs.uiuc.edu/PatternStories/JavaAWT

javax.servlet.Filter

http://tomcat.apache.org/tomcat-4.1-doc/servletapi/javax/servlet/Filter.html

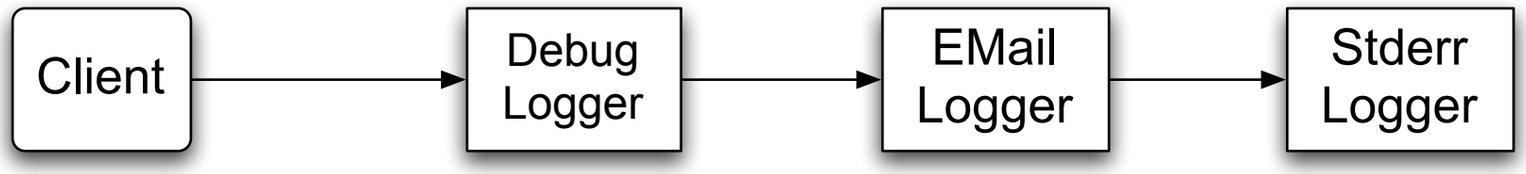
Microsoft Windows global keyboard events

http://www.javaworld.com/javaworld/jw-08-2004/jw-0816-chain.html

Apache Commons Chain

http://commons.apache.org/chain/

Logger Example




```
class ChainOfResponsibilityExample {
   public static void main(String[] args) {
      // building the chain of responsibility
      Logger l = new DebugLogger(Logger.DEBUG).setNext(
                  new EMailLogger(Logger.ERR).setNext(
                   new StderrLogger(Logger.NOTICE) ) );

      l.message("Entering function x.", Logger.DEBUG);    // handled by DebugLogger
      l.message("Step1 completed.", Logger.NOTICE);       // handled by Debug- and
StderrLogger
      l.message("An error has occurred.", Logger.ERR);    // handled by all three Logger
   }
}
```

First Attempt

```
abstract class Logger {
   public static int ERR = 3;
   public static int NOTICE = 5;
   public static int DEBUG = 7;
   protected int mask;

   protected Logger next;
   public Logger setNext(Logger l) {
      next = l;
      return this; }

   abstract public void message(String msg, int priority);
}

class DebugLogger extends Logger {
   public DebugLogger(int mask) {
      this.mask = mask; }

   public void message(String msg, int priority) {
      if (priority <= mask) debug log here
      if (next != null) next.message(msg, priority);
   }
}
```

```
class EMailLogger extends Logger {
   public EMailLogger(int mask) { this.mask = mask; }

   public void message(String msg, int priority) {
      if (priority <= mask) send email here;
      if (next != null) next.message(msg, priority);
   }
```

Improved Logger

```
abstract class Logger {
   public static int ERR = 3;
   public static int NOTICE = 5;
   public static int DEBUG = 7;
   protected int mask;

   protected Logger next;
   public Logger setNext(Logger l) {
          next = l;
         return this; }

   public void message(String msg, int priority) {
      if (priority <= mask) log(msg);
      if (next != null) next.message(msg, priority);
   }

   abstract void log(String message);
}

class StderrLogger extends Logger {
   public StderrLogger(int mask) { this.mask = mask; }

   void message(String msg, int priority) { send to err}
}
```

```
class EMailLogger extends Logger {
   public EMailLogger(int mask) { this.mask = mask; }

    void message(String msg, int priority) { email here}
   }
}

class DebugLogger extends Logger {
   public DebugLogger(int mask) { this.mask = mask; }

   void message(String msg, int priority) { debug stuff}
   }
}
```

Is this the Chain of Responsibility?

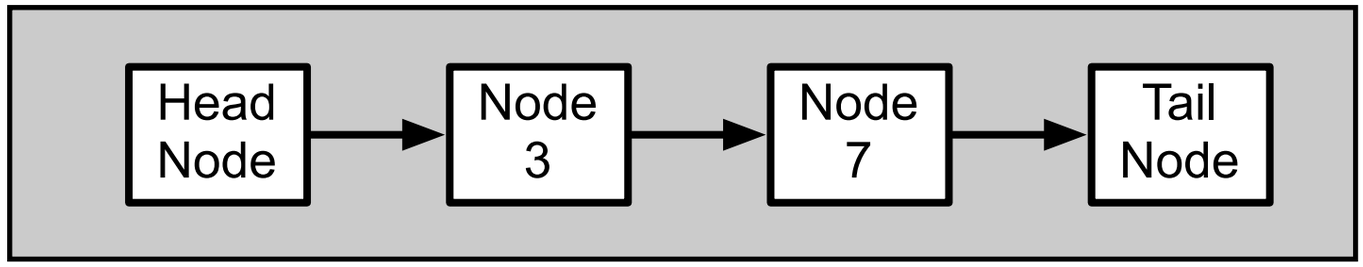
Object-Oriented Recursion

A method polymorphically sends its message to a different receiver

Eventually a method is called that performs the task

The recursion then unwinds back to the original message send

Linked List toString



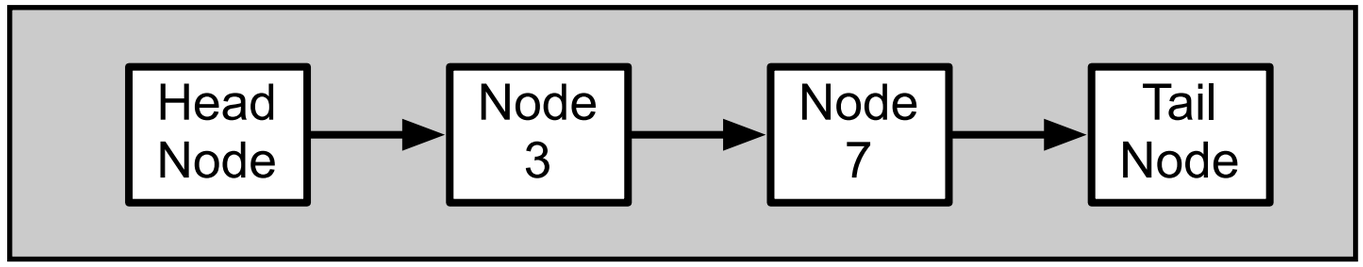
(3 7)

```
class HeadNode {
        public String toString() {
                return "(" + next.toString();
        }
}
```

```
class TailNode {
        public String toString() {
                return " )";
        }
}
```

```
class Node {
        public String toString() {
                return " " + element + next.toString();
        }
}
```

Linked List add




```
class HeadNode {
        public void add(int value) {
                next.add(value);
        }
}
```

```
class TailNode {
        public void add(int value) {
                prependNode(value);
        }
}
```

```
class Node {
        public void add(int value) {
                if (element > value)
                        prependNode(value);
                else
                        next.add(value);
        }
}
```

OO Recursion

Decorator

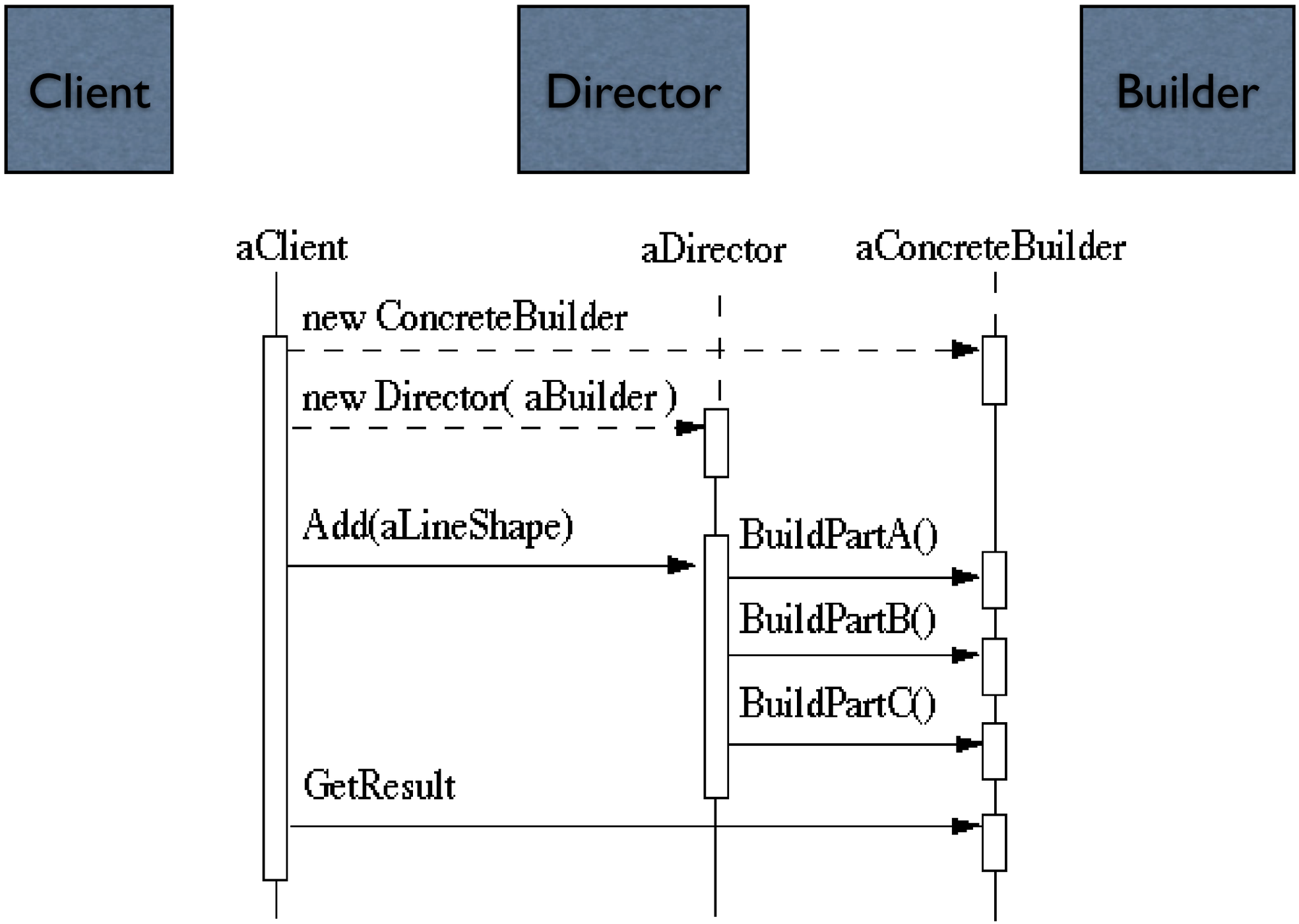
Chain of Responsibility

Builder

Separate construction of a complex object from its representation

So same construction process can create different representations

Builder


Client
Director
Builder
aClient
aDirector
aConcreteBuilder
new ConcreteBuilder
new Director(aBuilder)
Add(aLineShape)
BuildPartA()
BuildPartB()
BuildPartC()
GetResult

RTF Converter

A word processing document has complex structure

How to convert Rich Text Format (RTF) to

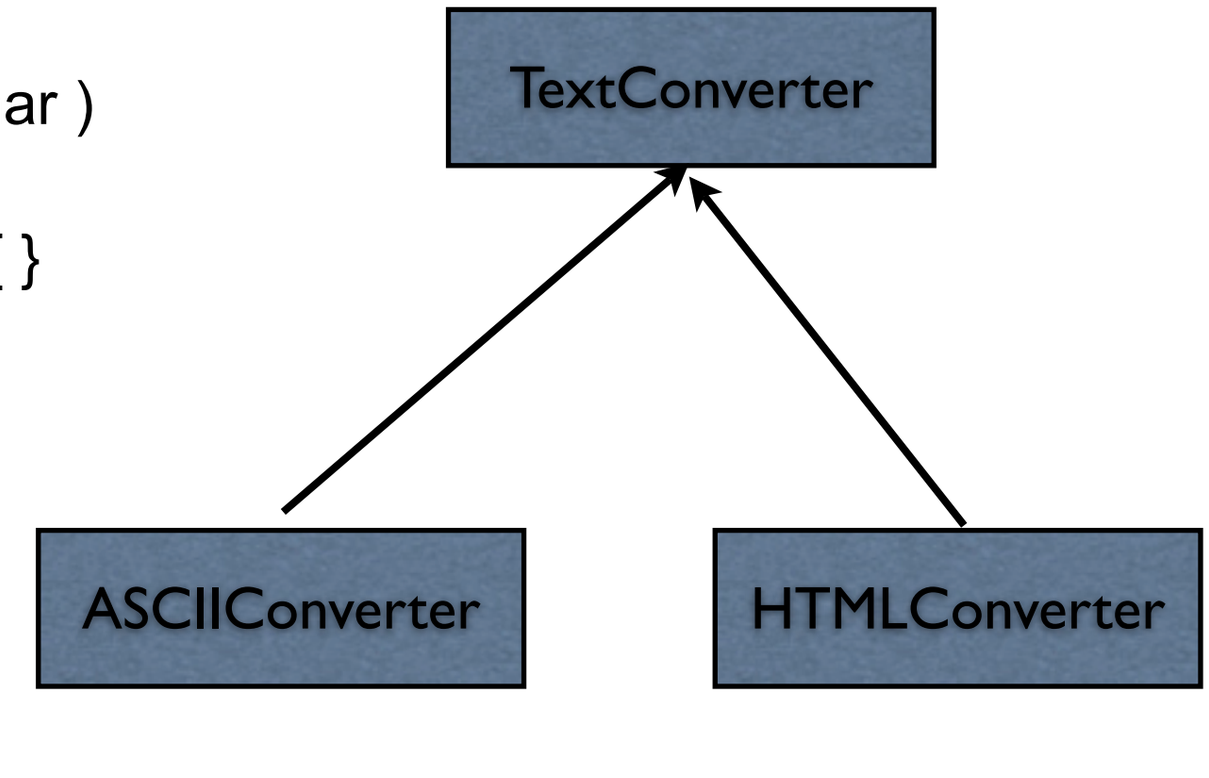
- TeX
- html
- PDF
- etc.

Pseudo Solution

```
class RTF_Reader {
   TextConverter builder;
   String RTF_Text;

   public RTF_Reader( TextConverter aBuilder, String RTFtoConvert ){
      builder = aBuilder;
      RTF_Text = RTFtoConvert;
      }

   public void parseRTF(){
      RTFTokenizer rtf = new RTFTokenizer( RTF_Text );

      while ( rtf.hasMoreTokens() ){
         RTFToken next = rtf.nextToken();

         switch ( next.type() ){
            case CHAR:    builder.character( next.char() ); break;
            case FONT:    builder.font( next.font() ); break;
            case PARA:    builder.newParagraph( ); break;
            etc.
         }
      }
   }
}
```

Builder Classes

```
abstract class TextConverter {
   public void character( char nextChar )
{  }
   public void font( Font newFont )   { }
   public void newParagraph() {}
}
```



Sample Program

```
main(){
   ASCII_Converter simplerText = new ASCII_Converter();
   String rtfText;

   // read a file of rtf into rtfText

   RTF_Reader myReader =
      new RTF_Reader( simplerText, rtfText );

   myReader.parseRTF();

   String myProduct = simplerText.getText();
}
```

The Hard Part

The builder interface

XML Example

```
<?xml version="1.0" encoding="UTF-8"?>
<RootElement param="value">
    <FirstElement>
        Some Text
    </FirstElement>
    <SecondElement param2="something">
        Pre-Text <Inline>Inlined text</Inline> Post-text.
    </SecondElement>
</RootElement>
```

SAX - Builder Pattern

XMLReader

ContentHandler

ContentHandler Interface

void characters(char[] ch, int start, int length)

Receive notification of character data.

void endDocument()

Receive notification of the end of a document.

void endElement(java.lang.String uri, java.lang.String localName, java.lang.String qName)

Receive notification of the end of an element.

void endPrefixMapping(java.lang.String prefix)

End the scope of a prefix-URI mapping.

void ignorableWhitespace(char[] ch, int start, int length)

Receive notification of ignorable whitespace in element content.

void processingInstruction(java.lang.String target, java.lang.String data)

Receive notification of a processing instruction.

void setDocumentLocator(Locator locator)

Receive an object for locating the origin of SAX document events.

void skippedEntity(java.lang.String name)

Receive notification of a skipped entity.

void startDocument()

Receive notification of the beginning of a document.

void startElement(java.lang.String uri, java.lang.String localName, java.lang.String qName, Attributes atts)

Receive notification of the beginning of an element.

void startPrefixMapping(java.lang.String prefix, java.lang.String uri)

Begin the scope of a prefix-URI Namespace mapping.

Simple API XML (SAX)

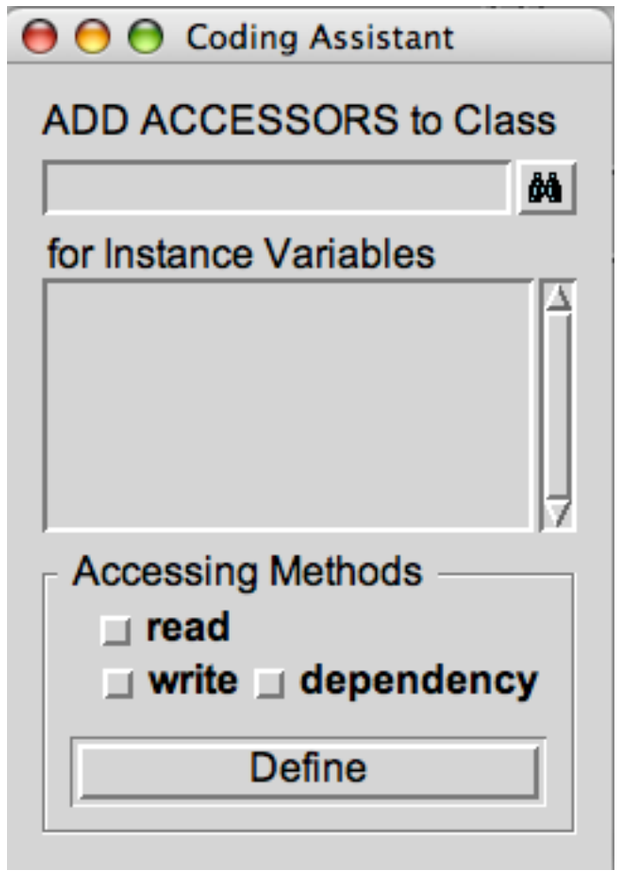
```
public static void main (String args[]) throws Exception {
  XMLReader director = XMLReaderFactory.createXMLReader();
  ContentHandler builder = new MySAXApp();
  director.setContentHandler(builder);
  director.setErrorHandler(builder);

  FileReader source = new FileReader("Foo.xml");
  director.parse(new InputSource(source));
  handler.getResult();
  }
```

Examples - VW Smalltalk

ClassBuilder
MenuBuilder
UIBuilder

UIBuilder

```
#(#{UI.FullSpec}
    #window:
    #(#{UI.WindowSpec}
      #label: #(#{Kernel.UserMessage} #key: #CodingAssistant
            #defaultString: 'Coding Assistant' #catalogID: #UIPainter)
      #min: #(#{Core.Point} 242 320 )
      #max: #(#{Core.Point} 242 320 )
      #bounds: #(#{Graphics.Rectangle} 279 140 521 460 ) )
    #component:
    #(#{UI.SpecCollection}
      #collection: #(
        #(#{UI.LabelSpec}
          #layout: #(#{Graphics.LayoutOrigin} 14 0 12 0 )
          #label: #(#{Kernel.UserMessage} #key: #ADDACCESSORSToClass
            #defaultString: 'ADD ACCESSORS to Class' #catalogID: #UIPainter) )
        #(#{UI.LabelSpec}
          #layout: #(#{Graphics.LayoutOrigin} 16 0 65 0 )
          #label: #(#{Kernel.UserMessage} #key: #forInstanceVariables
              #defaultString: 'for Instance Variables' #catalogID: #UIPainter) )
```



Strategy
vs
Builder