Assignment 3
public class ASS3Test {
    boolean debug = false;
    URLConnection urlConnection;

    @Before
    public void init() {
        try {
            if (debug) {
                this.urlConnection = mock(URLConnection.class);
                when(urlConnection.getLastModified()).thenReturn(1533389966554L);
            } else {
                URL url = new URL("http://www.google.com");
                this.urlConnection = url.openConnection();
            }
        }
    } catch (Exception e) {
        System.out.println(e.getMessage());
    }
    }
}
Student Test Class - Continued

```java
@Test
public void test() {
    try {
        WebUpdater wd = new WebUpdater(urlConnection);
        wd.addObserver(new Observer1);
        wd.addObserver(new Observer2);
        wd.checkModifiedDate();
    } catch (Exception e) {
        System.out.println(e.getMessage());
    }
}
```
Student Question

is init() a factory method?
Which is better?

URL url = new URL("http://www.google.com");
urlConnection = url.openConnection();
WebUpdater wd = new WebUpdater(urlConnection);

Verses

WebUpdater wd = new WebUpdater("http://www.google.com");
@Test
public void test() {
    try {
        WebUpdater wd = new WebUpdater(urlConnection);
        wd.addObserver(new Observer1());
        wd.addObserver(new Observer2());
        wd.checkModifiedDate();
    } catch (Exception e) {
        System.out.println(e.getMessage());
    }
}

How is this a test?
Prototype
Prototype

Specify the kinds of objects to create using a prototypical instance, and create new objects by copying this prototype

Applicability

Use the Prototype pattern when

A system should be independent of how its products are created, composed, and represented; and

When the classes to instantiate are specified at run-time; or

To avoid building a class hierarchy of factories that parallels the class hierarchy of products; or

When instances of a class can have one of only a few different combinations of state.
Insurance Example

Insurance agents start with a standard policy and customize it

Two basic strategies:

Copy the original and edit the copy

Store only the differences between original and the customize version in a decorator
Copying Issues

Shallow Copy Verse Deep Copy

Original Objects

Shallow Copy
Shallow Copy Verse Deep Copy

Original Objects

Deep Copy

Deeper Copy
class Door {
   public:
      Door();
      Door( const Door&);
      virtual Door* clone() const;

      virtual void Initialize( Room*, Room* );
      // stuff not shown

   private:
      Room* room1;
      Room* room2;
   }

Door::Door ( const Door& other ) //Copy constructor {
   room1 = other.room1;
   room2 = other.room2;
}

Door* Door::clone() const {
   return new Door( *this );
}
Cloning Issues - Java Clone

**Shallow Copy**

class Door implements Cloneable {
    private Room room1;
    private Room room2;

    public Object clone() throws CloneNotSupportedException {
        return super.clone();
    }
}

**Deep Copy**

public class Door implements Cloneable {
    private Room room1;
    private Room room2;

    public Object clone() throws CloneNotSupportedException {
        Door thisCloned = (Door) super.clone();
        thisCloned.room1 = (Room) room1.clone();
        thisCloned.room2 = (Room) room2.clone();
        return thisCloned;
    }
}
Prototype-based Languages

No classes

Behaviour reuse (inheritance)
  Cloning existing objects which serve as prototypes

Some Prototype-based languages

Self
JavaScript
Squeak (eToys)
Perl with Class::Prototyped module
Chain of Responsibility
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

Microsoft Windows global keyboard events

Apache Commons Chain
http://commons.apache.org/chain/
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
    }
}
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);
    }
}
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.
class HeadNode {
    public String toString() {
        return "(" + next.toString();
    }
}

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

class TailNode {
    public String toString() {
        return ")";
    }
}
```java
class HeadNode {
    public void add(int value) {
        next.add(value);
    }
}

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

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

Decorator  Chain of Responsibility