Tuesday 10/14/14 4156

review

ch. 6 command pattern

client creates command object

client calls setCommand() to store the command object at invoker

later client asks invoker to execute command

command may be used once or many times

encapsulates a request
can parameterize other objects
with different request
queue or log requests - store + load
support undoable operations
- Undo Command

No Command null object

macro - array of command objects

Tuesday 10/14/14 4156

ch. 7 adapter + facade patterns

adapter - intermediary between code that expects one interface & service that provides a different interface

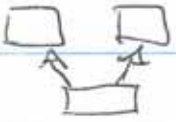
Client makes request to adapter calling a method on target interface

adapter translates request into one or more calls on adaptee using adaptee interface

client receives result of call without knowing anything about the translation

adapter pattern converts the interface of a class into another interface the client expects, allows classes to work together even tho they have incompatible interfaces (object adapter)

also class adapter, but needs multiple inheritance - inherits from both sides



Tuesday 10/14/14 4156

facade [redacted] - hides complexity of one or more classes behind a single clean facade

used primarily for complex subsystem where most clients do not need direct access to the underlying subsystem classes

difference is to simplify vs. alter an interface

facade pattern provides unified interface to a set of interfaces for a subsystem
intent is a higher-level interface that makes subsystem easier to use

design principle - principle of least knowledge
- limits interactions a class has w/ other classes & # of other classes

lots of dependencies → fragile
(aka Law of Demeter)

Tuesday 10/14/14 4156

ch. 8 template method pattern

defines the skeleton steps of an algorithm & defers implementations of one or more steps to subclasses

steps can be abstract methods
subclass MUST supply implementation

or can be "hooks" - empty or default implementation that subclass need not provide

tells subclasses "don't call us, we'll call you"

often used for frameworks

different from strategy which composes with another object to implement entire algorithm

Tuesday 10/14/14 4154

ch 9 Collections

Iterator Pattern - hasNext()
next()

provides a way to access the elements of an aggregate object sequentially w/o exposing underlying rep

places task of traversal on iterator object, not on aggregate

make no assumptions about ordering

composite pattern - allows to compose into tree structures to represent part-whole hierarchies, lets clients treat individual objects & compound objects uniformly
leaves & nodes in tree

apply operations to whole or part

Component interface - add, remove, getChild plus operations
some operations only make sense for composite or leaf

default impl. throws exception

Tuesday 10/14/14 4156

iterate over composite

internal - each composite uses iterator on its children & calls operation for each child recursively

external - so a client can walk over tree, need to maintain position for hasNext & next

use stack to hold iterator object for each level - code in book

need null Iterator for nodes with no children - hasNext always returns false (cf. null value & conditional check)

other composite issues

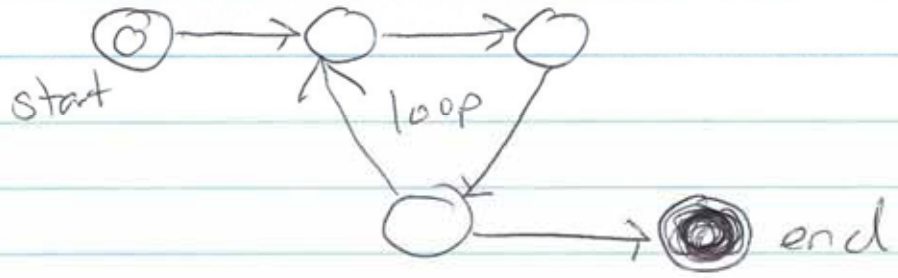
children might or might not reference parents
need specific orderings of children
cache frequent operations rt keep repeating traversal

point is uniformity for client - doesn't need to check node vs. leaf

Tuesday 10/14/14 4156

ch. 10 state pattern

FSA



instance variable represents current state
each operation has conditional to check state to see what to do
- but if a new state is added, need to change all the operations

instead new state subclass for each state, delegate each operation to current state

state interface defines all operations

each state "knows" what state it is
in so no conditional - does its own variant of the operation
- which might set state instance variable to a new state object

- closed for modification
- open for extension

so easy to add new states

Tuesday 10/14/14 4156

state pattern - allows an object to alter its behavior when internal state changes — object appears to change its class (via the composition + delegation to its internal State variable)

similar to strategy pattern, but there chosen of algorithm determined by context w.r.t state transition with a setter to the context object

State objects can be declared static & shared among instances of the context (if no internal state specific to a context)

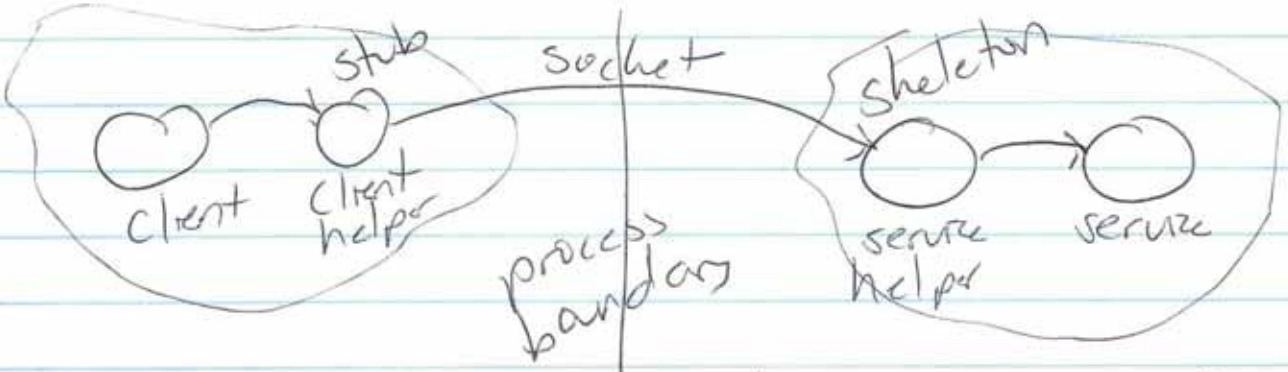
Tuesday 10/14/14 4156

ch. 11 proxy pattern

proxy - controls & manages access to another object

remote proxy - local representative of a remote object
sends operations to the real object & returns results
handles all low level network ops

Java RMI, etc.



registry for where services found & access to local proxies
arguments & return types must be serializable

proxy pattern - provides surrogate or placeholder to another object to control access to it

Tuesday 10/14/14 4/56

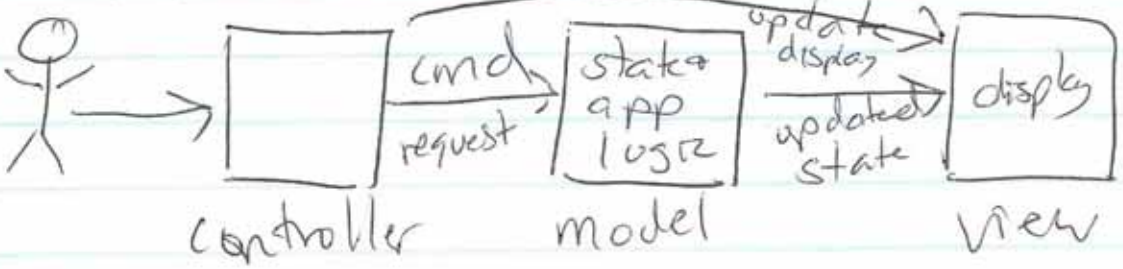
proxy must impl. same interface as object it controls & holds reference to that object

Similar to decorator but diff. purpose - add behavior vs. control access

many variations
protection access control
virtual
caching
reference counting
firewall network resources
synchronization safe multithreading
copy on write
complexity - like facade

ch. 12 compound patterns
standard set of patterns working together

MVC - Model - View - controller



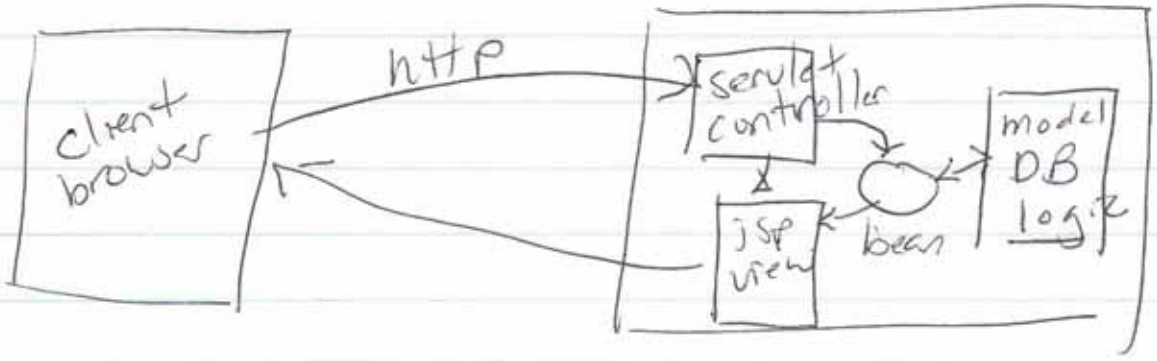
Tuesday 10/14/14 4156

model - uses Observer to keep view & controller updated on state changes
not dependent on view or controller

view/controller - Strategy
can swap another
view object configured with controller (strategy) object

view - composite UI

MVC for Web "Model 2" adaptation



bean convention
all properties private getter/setter
public no-argument constructor
implement serializable interface

Tuesday 10/14/14 4156

ch. 13 (last chapter)

finally define "design pattern"
- solution to a problem in a context

context - recurring design situation

problem - goal intending to achieve & also constraints

solution - general design that resolves goal wrt constraints

example
context - collection of objects
problem - need to step thru w/o exposing collection impl.
solution - encapsulate iteration in another class - iterator

patterns catalogs
Gang of Four book
domain specific catalogs
examples in diff. prog lang

discovered, not invented - rule of three

Tuesday 10/14/14 4156

creational - object instantiation
decouple client from objects
it needs to instantiate

structural - compose classes or objects
into larger structures

behavioral - how classes & objects
interact to distribute responsibility

creational - singleton, abstract factory,
factory method, others

structural - decorator, composite, proxy,
facade, adapter, others

behavioral - template method, command,
iterator, observer, state,
strategy, others

class vs. object patterns
inheritance composition

template method ~~most~~ others
factory method
adaptor, interpreter