N Design Patterns You Might Actually Use

Aja Hammerly
5 Design Patterns
You Might Actually Use

Aja Hammerly
Aja Hammerly
@thagomizer_rb
http://github.com/thagomizer
http://www.thagomizer.com

SUSTANTIAL
Motivations
MAGIC SHOW
Singleton?
Factory?
Something Else?

https://www.flickr.com/photos/laughingsquid/7658733440/
again, looks like strategies
Design Patterns
Common Vocabulary
Kool-Aid
INSTANT SOFT DRINK MIX • ADD SUGAR

RASPBERRY IMITATION FLAVOR

DIRECTIONS: Empty contents into large pitcher. Add 1 cup sugar. Add COLD water and ice to make 2 quarts.

Fumaric acid, sugar, artificial flavor and color, calcium carbonate, dioctyl sodium sulfosuccinate (DSS).

GENERAL FOODS CORP., WHITE PLAINS, N.Y., U.S.A.

NET WT. 4.5 GM.

https://www.flickr.com/photos/roadsidepictures/4488040607
Setting Expectations

• ~140 Slides in 30 minutes
• Code Medium
• Advanced Beginner - Intermediate
Patterns

- Template
- Strategy
- Composite
- Decorator
- Command
Template
What Is It?
Outline For Similar Tasks
Examples
Active Record Models
TPS REPORT
(TOILET PAPER SUPPLY)

96 ROLLS P/BOX
4 BOXES \( \rightarrow \) 384 ROLLS

1000 SHEETS P/ROLL

4 SHEETS P/ROLL BEGINNING OF ROLL
2 SHEETS P/ROLL END OF ROLL

384 ROLLS AVAILABLE SHEETS

USAGE

32 PERSONS (AVERAGE)
12 SHEETS P/WIPE
2 WIPES P/DAY \( \times 2 \text{ for all} \)

\( (32 \cdot 12 \cdot 2) = 768 \text{ SHEETS P/DAY} \)

CAPACITY OF CURRENT INVENTORY

379,392 \( \div \) 768 = 494 DAYS

PENDING
DIETARY
CHANGES

https://www.flickr.com/photos/elorg/2584274950/
Recognizing It
class PastDueAccountsReport

    def extract
        @data = Accounts.past_due
    end

    def aggregate
        @data = @data.group_by { |a| a.owner }
    end

    def email
        Email.new("accounting@adequate.hq", @data)
    end

    def generate
        extract
        aggregate
        email
    end
end
class NewUsersReport

  def extract
    @data = Users.new_in_last_24_hours
  end

  def aggregate
    @data = @data.group_by { |u| u.department }
  end

  def email
    Email.new("reports@adequate.hq", @data)
  end

  def generate
    extract
    aggregate
    email
  end
end
class NewUsersReport

  def extract
    @data = Users.new_in_last_24_hours
  end

  def aggregate
    @data = @data.group_by { |u| u.department }
  end

  def email
    Email.new("reports@adequate.hq", @data)
  end

  def generate
    extract
    aggregate
    email
  end
end
class ReportTemplate
  def extract; raise "NYI"; end
  def aggregate; raise "NYI"; end
  def email; raise "NYI"; end

  def generate
    extract
    aggregate
    email
  end
end
class ReportTemplate
  def extract; raise "NYI"; end
  def aggregate; raise "NYI"; end
  def email; raise "NYI"; end

  def generate
    extract
    aggregate
    email
  end
end
class ReportTemplate
  def extract; raise "NYI"; end
  def aggregate; raise "NYI"; end
  def email; raise "NYI"; end
  def generate
    extract
    aggregate
    email
  end
end
class NewUsersReport < ReportTemplate

  def extract
    @data = Users.new_in_last_24_hours
  end

  def aggregate
    @data = @data.group_by { |u| u.department }
  end

  def email
    Email.new("reports@adequate.hq", @data)
  end
end
class PastDueAccountsReport < ReportTemplate

  def extract
    @data = Accounts.past_due
  end

  def aggregate
    @data = @data.group_by { |a| a.owner }
  end

  def email
    Email.new("accounting@adequate.hq", @data)
  end
end
Share Common Code
Isolate Differences
Easy To Add New Versions
Force Common Algorithm
Read multiple files

Inheritance

Inheritance
Report A

- Extract 1
- Transform 1
- Load 1

Report B

- Extract 1
- Transform 2
- Load 2

Report C

- Extract 2
- Transform 3
- Load 2
Strategy
What Is It?
Polymorphic Algorithm
Polymorphic

what?
Change The Algorithm At Runtime
blah blah algorithm?
Extract Algorithm
Examples
Storage Strategy

http://upload.wikimedia.org/wikipedia/commons/8/85/Komórki_1_m_self_storage.jpg
Stock Market

Sales Tax

http://www.flickr.com/photos/phillip/345829246/
Recognizing It
def handle(a, b, c):
    # step 1
    normalize_inputs a, b, c

    # step 2
    reticulate_splines

    # step 3
    if a == 'x'
        results = x_request a, b
    elsif a == 'y'
        results y_request a
    elsif a == 'z' and b == '123'
        results z_request a, b, c
    end

    # step 4
    present results
end
def handle(a, b, c)
    # step 1
    normalize_inputs a, b, c

    # step 2
    reticulate_splines

    # step 3
    if a == 'x'
        results = x_request a, b
    elsif a == 'y'
        results y_request a
    elsif a == 'z' and b == '123'
        results z_request a, b, c
    end

    # step 4
    present results
end
def initialize
  @strategies = {}
  @strategies['x'] = XStrategy.new
  @strategies['y'] = YStrategy.new
  @strategies['z'] = ZStrategy.new
end

def handle a, b, c
  # step 1
  normalize_inputs a, b, c

  # step 2
  reticulate_splines

  # step 3
  @strategies[a].run a, b, c

  # step 4
  present results
end
def initialize
    @strategies = {}
    @strategies['x'] = XStrategy.new
    @strategies['y'] = YStrategy.new
    @strategies['z'] = ZStrategy.new
end

def handle a, b, c
    # step 1
    normalize_inputs a, b, c

    # step 2
    reticulate_splines

    # step 3
    @strategies[a].run a, b, c

    # step 4
    present results
end
def initialize()
    @strategies = {}
    @strategies['x'] = XStrategy.new
    @strategies['y'] = YStrategy.new
    @strategies['z'] = ZStrategy.new
    @strategies['aa'] = AAStrategy.new
end

def handle(a, b, c)
    # step 1
    normalize_inputs a, b, c

    # step 2
    reticulate_splines

    # step 3
    @strategies[a].run(a, b, c)

    # step 4
    present results
end
def add_strategy(key, strategy)
  @strategies[key] = strategy
end
add_strategy 'aa', AAStategy.new
Share Common Code

Isolate Differences

Easily Add New Strategies

Force Common Interface

Avoid Inheritance
Read multiple files
Harder to follow in source
Can be lots of data
Must have the same interface
Composite
What Is It?
Common Interface
For One or Many
Common Interface?
Let Consumers Ignore Quantity
Vocabulary
Component

operation()

Leaf

operation()

Composite

@subcomponents[]

operation()
Examples
Task Management
<!DOCTYPE HTML PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd>

<html>
<head>
  <meta name="TITLE" content="/HTML5_de_Erick_Dimas.jpg"/>
  <meta name="KEYWORDS" content="/HTML5_de_Erick_Dimas.jpg"/>
  <meta name="DESCRIPTION" content="/HTML5_de_Erick_Dimas.jpg"/>
</head>
<body bgcolor="#ffffff">

<html>
</body>
</html>
Test Scores

90% Real

http://farm3.static.flickr.com/2177/2230010178_40c2741290.jpg
Recognizing It
def report_scores(x)
  case x
  when Student
    x.score
  when Classroom
    x.students.map(&:score).avg
  when School
    x.classes.map(&:students).flatten.map(&:score).avg
  end
end
def report_scores x
    case x
        when Student
            x.score
        when Classroom
            x.students.map(&:score).avg
        when School
            x.classes.map(&:students).flatten.map(&:score).avg
    end
end
def report_scores x
  case x
    when Student
      x.score
    when Classroom
      x.students.map(:score).avg
    when School
      x.classes.map(:students).flatten.map(:score).avg
  end
end
def report_scores x
  case x
  when Student
    x.score
  when Classroom
    x.students.map(&:score).avg
  when School
    x.classes.map(&:students).flatten.map(&:score).avg
  end
end
Composite
class Composite
  def score
    # Some implementation
  end
end
class Student
  attr_accessor :score
end
class Classroom
  attr_accessor :students

  def score
    students.map(&:score).avg
  end
end
class School
  attr_accessor :classes
  def score
    classes.map(&:score).avg
  end
end
def report_scores(x)
    x.score
end
Common Interface
Arbitrary Depth
Insert In the Middle
No Messy Ifs
Harder to follow in source
Must remember hierarchy
Decorator
What Is It?
Extend an Instance of an Object
an instance of food?

http://laura-c-f.deviantart.com/art/Confused-Dog-286725830
Wrapper Adding Functionality
My Object
Examples
GUI Elements
Calculating Discounts

http://therockinstallion.deviantart.com/art/Sparky-BOGO-2-390047802
RPG Characters
Recognizing It
class Character
  attr_accessor :health, :damage, :xp

  def alive?
    health > 0
  end

  def attack enemy
    enemy.health -= self.damage
  end
end
POW!!
require 'forwardable'

class DragonMightDecorator
  extend Forwardable

  def_delegators :@character, :health, :xp

  def initialize(character)
    @character = character
  end

  def damage
    @character.damage * 1.5
  end
end
require 'forwardable'

class DragonMightDecorator
  extend Forwardable

  def_delegators :@character, :health, :xp

  def initialize(character)
    @character = character
  end

  def damage
    @character.damage * 1.5
  end
end
require 'forwardable'

class DragonMightDecorator
  extend Forwardable

  def_delegators :@character, :health, :xp

  def initialize(character)
    @character = character
  end

  def damage
    @character.damage * 1.5
  end
end
require 'forwardable'

class DragonMightDecorator
  extend Forwardable

  def_delegators :@character, :health, :xp

  def initialize(character)
    @character = character
  end

  def damage
    @character.damage * 1.5
  end
end
> c = Character.new
> c.health = 100; c.damage = 10;

> d = DragonMightDecorator.new(c)
> d.damage
15.0

> d.health
100
Change Behavior Dynamically

Customize an Instance

Support Nesting
Offload Pain To Consumers

Multiple Files

Many Decorators -> Unpredictable
Command
What Is It?
Objectified Closure
What Is Closure?
Object Containing Algorithm & Context
Examples
Delayed Job
Drawing Applications

http://upload.wikimedia.org/wikipedia/commons/1/1e/Tuxpaint-magic-tools.png
Active Record
Migrations

100% Real
Recognizing It
Undo
class CreateRecipes < ActiveRecord::Migration
  def self.up
    create_table :recipes do |t|
      t.column  :name, :string
      t.timestamps
    end
  end

  def self.down
    drop_table :recipes
  end
end
class CreateRecipes < ActiveRecord::Migration
  def self.up
    create_table :recipes do |t|
      t.column :name, :string
      t.timestamps
    end
  end

  def self.down
    drop_table :recipes
  end
end
Delay
Delegation
Undo & Redo
Offload / Queue / Distribute / Delegate
Simple To Understand
Encourages Good Design
Overused
YAGNI
So What?
You Are Using Patterns
Common Vocabulary
Cheat Sheet
Several Objects That Are Mostly The Same?
Template
Process The Same Except *Step 3*?
Strategy
Hierarchical Objects?
Composite
Sometimes Need Enhancements?
Decorator
Undo or Delay or Delegate?
Command
Design Patterns

Elements of Reusable Object-Oriented Software

Erich Gamma
Richard Helm
Ralph Johnson
John Vlissides

Foreword by Grady Booch
Design Patterns
Elements of Reusable Object-Oriented Software
Erich Gamma
Richard Helm
Ralph Johnson
John Vlissides

Foreword by Grady Booch

Gang Of Four
Design Patterns In Ruby
Pattern Index

The book *Pattern Languages Of Program Design* http://hillside.net/patterns/books/Details/070.htm is an eclectic collection of patterns. Many of them are accessible on line.

See CategoryPattern index.

The GangOfFour wrote up 23 DesignPatterns. These patterns elaborate on their work:

- ExtensionObjects
- HiddenManagers

They are also interested in UsesOfGofPatterns.

Some pattern languages that are useful for large business applications.

- CrossingChasms
- RelationalDatabaseAccessLayer
- ErrorHandling
- DecouplingObjects (DecouplingOfObjectOrientedSystems)
- FrameworkConstruction

Other sets of patterns (often pattern languages).

- UserInterfacePatterns
- MessagingPattern
- SystemOfNames
- TransactionsAndAccounts
- DebuggingPatternLanguage
- TestingPatterns
- OrganizationalPatterns
- FunctionalPatternSystemForObjectOrientedDesign
- A pattern language for ObjectBasedProgramming in a procedural language.
- PatternsForEffectiveMeetings
- ComponentDesignPatterns
Thanks