User Interfaces

Lecture25

The Exam & Revision

Hamza Bennani
hamza@hamzabennani.com

October 9, 2018

Exam Structure

- Similar to resent years' exam.
- Classes and Objects, Memory Management, Inheritance and Polymorphism, Object Interconnections and design patterns, Usability, User Interface Design, MVC, Cocoa Applications

Exam Instructions

- Date???
- Three hours to complete the exam
- Answer all questions
- No supplementary material or calculators
- No books, notes, or other written materials
- 8 questions
- Worth 60% of your final grade.

Exam Advice

- Read questions carefully - check what is being asked
  - Listing or defining terms
  - Explaining or discussing concepts
  - Comparing advantages/disadvantages
  - . . .
- Check how much questions are worth
  - Rule of thumb is 1 mark per point made
  - Basic recall might be worth less

Exam Advice

- Use examples to illustrate ideas
- OK to write notes or bullet points
  - Better to write the key points clearly
- Use diagrams where appropriate
- Label your answers with question numbers
  - Indicate if parts of questions are on different pages

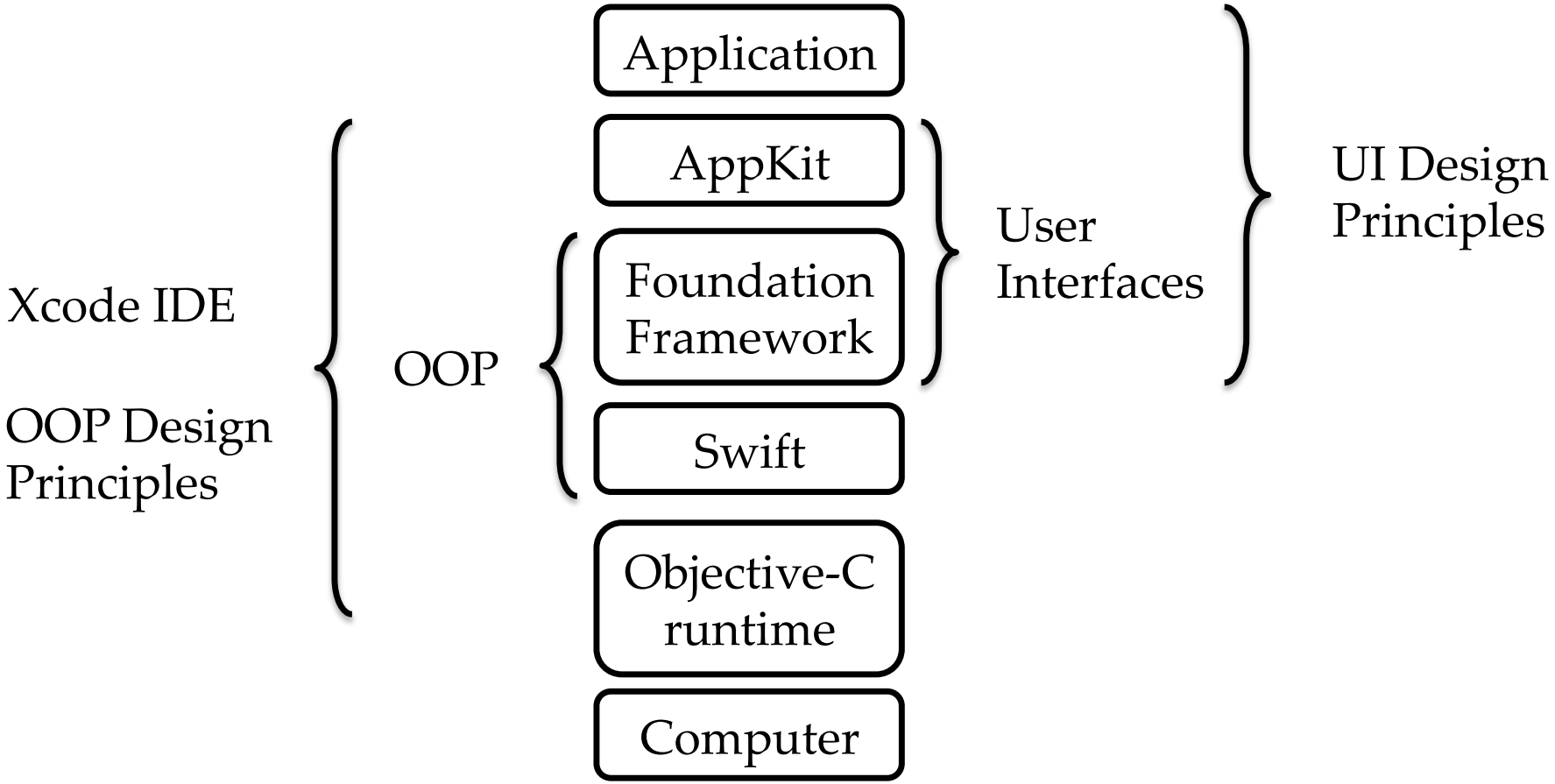
Example Questions

Look last years exam.

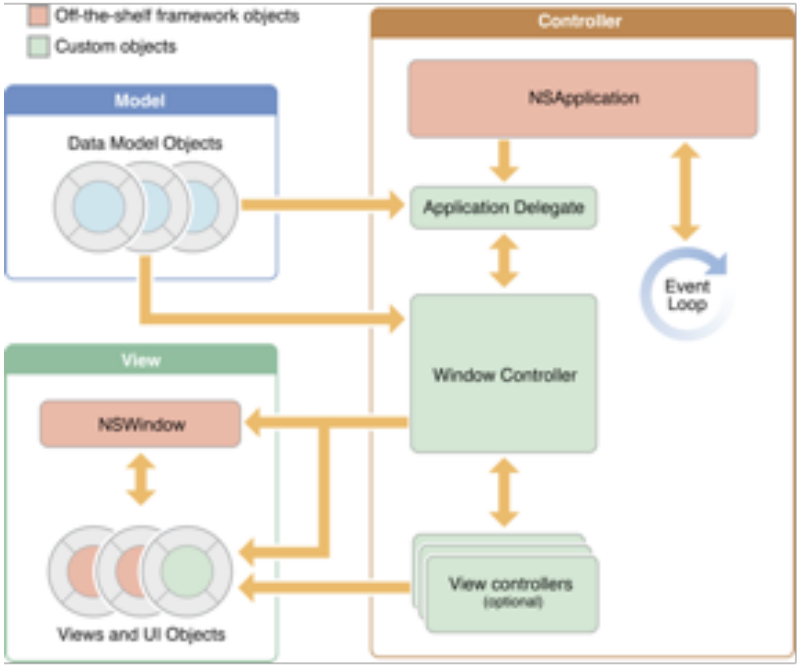
Course Summary
OO

- OO and Swift
- Classes and objects
- Inheritance
- Polymorphism
- Memory management
- Other key topics . . .
- Lecture 13: OO Review

Course Summary
UI Review

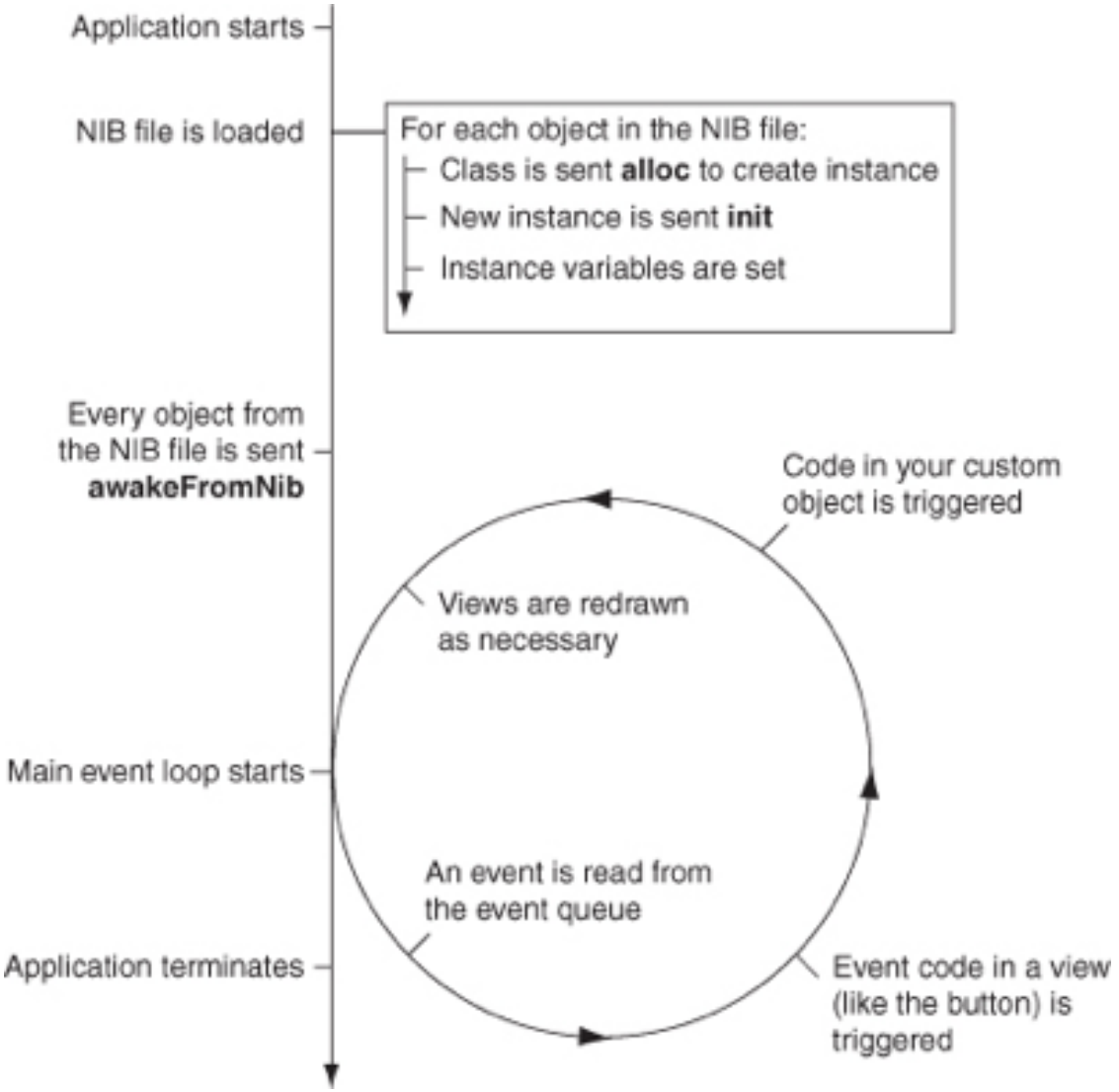


Cocoa Application
Types



Cocoa Application

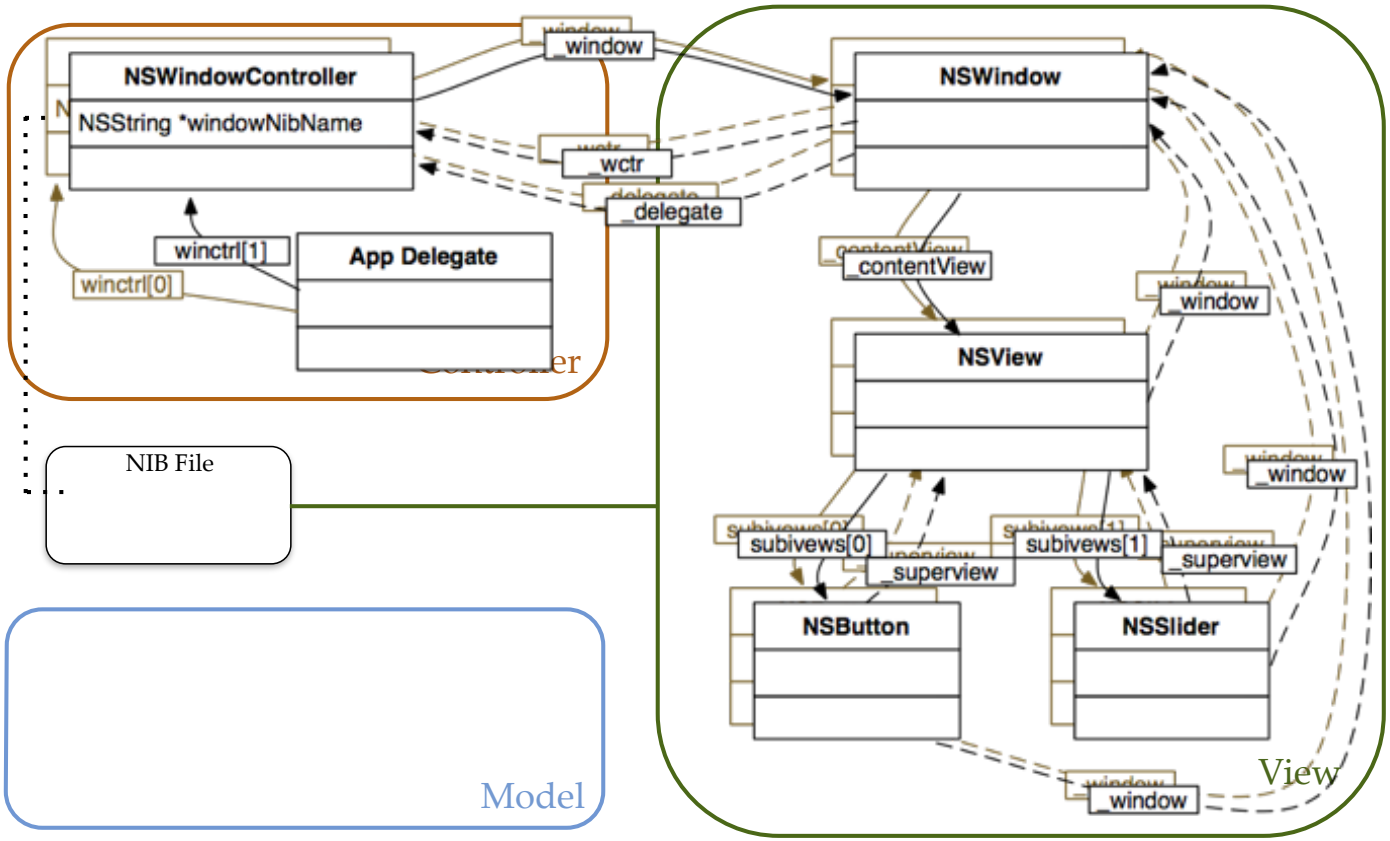
Life cycle (Bundle, Event loop, XIB/NIB or storyboard, NSApplication, App. delegate, Outlets, Target/action)



Cocoa Views and Windows

- Windows
  - Window delegate
  - Window controller
  - File's owner
  - Panels
  - Dialogs
- Views
  - Content view
  - View hierarchy
  - View geometry
  - Frame
  - Bounds

Cocoa Views and Windows

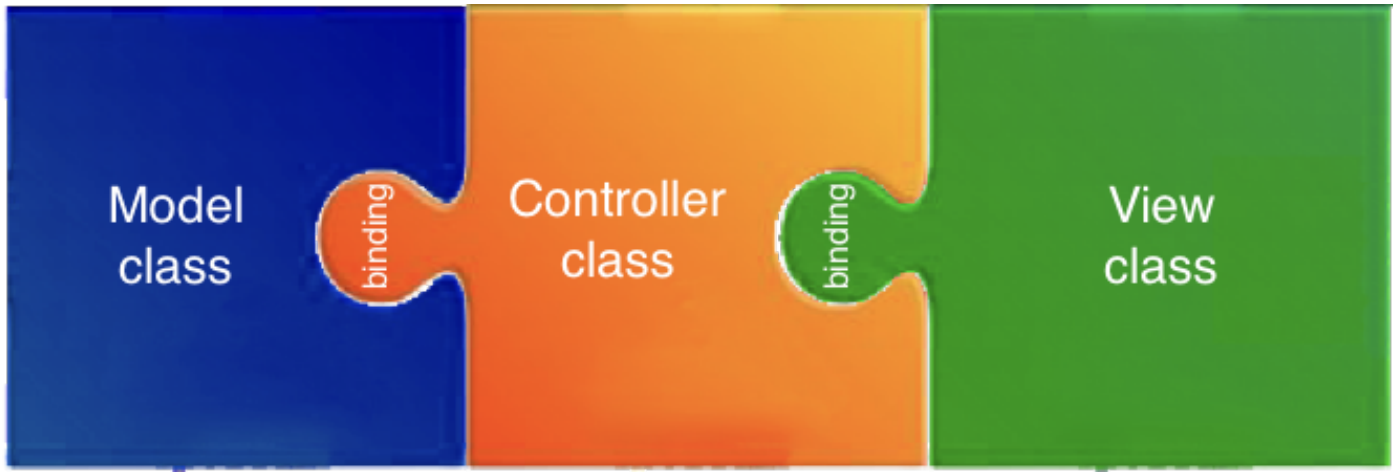


Cocoa Events

- **NSResponder** - abstract class that handles events
- **NSEvent** - class describing events
- **NSControl** - class that inherits from NSResponder and implements target action mechanism
- Responder chain
  - First responder
  - Initial responder
  - First responder proxy (within Xcode)

Cocoa Bindings

- **KVC** - access to object setters and getters using string with variable name
- **KVO** - special notification system for objects to track the changes in value of another object?s instance variable
- **NSController**
  - **NSObjectController**
  - **NSArrayController**

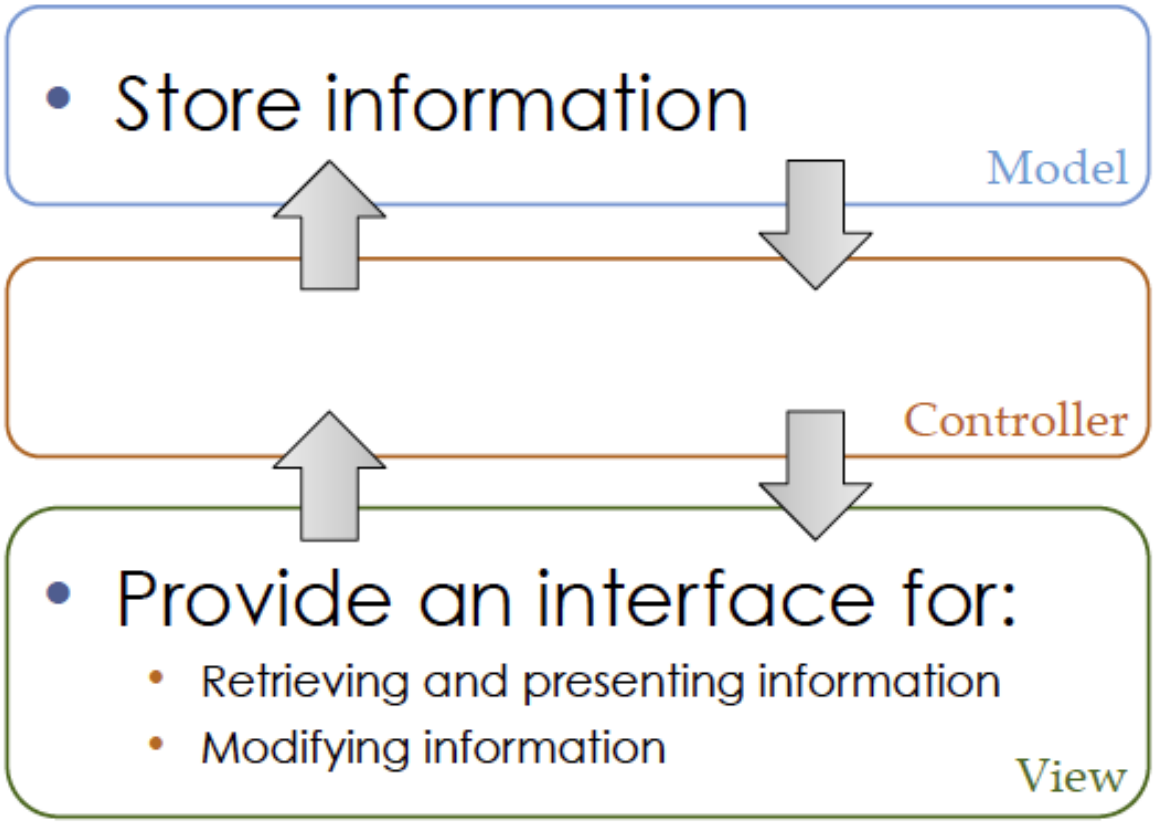


Cocoa Other

- **UndoManager** - For managing undo and redo stack
- **Notification** - For storing messages
- **NotificationCenter** - For registration of observes and dispatching of messages
- **UserDefaults** - For storing user preferences

MVC in Applications

Generally, applications perform the following functions:



The nature and type of information changes rarely

The role of the controller is to decouple the model from the view

The way we interact with information changes more often

UI Design Patterns
Solutions to common UI problems

- Even if you don't do any user research at all, your applications should incorporate many of these patterns
- Safe exploration
- Instant gratification
- Satisficing
- Changes in midstream
- Deferred choices
- Incremental construction
- Habituation
- Spatial memory
- Prospective memory
- Streamlined repetition
- Support keyboard-only operation

Content Organisation Patterns

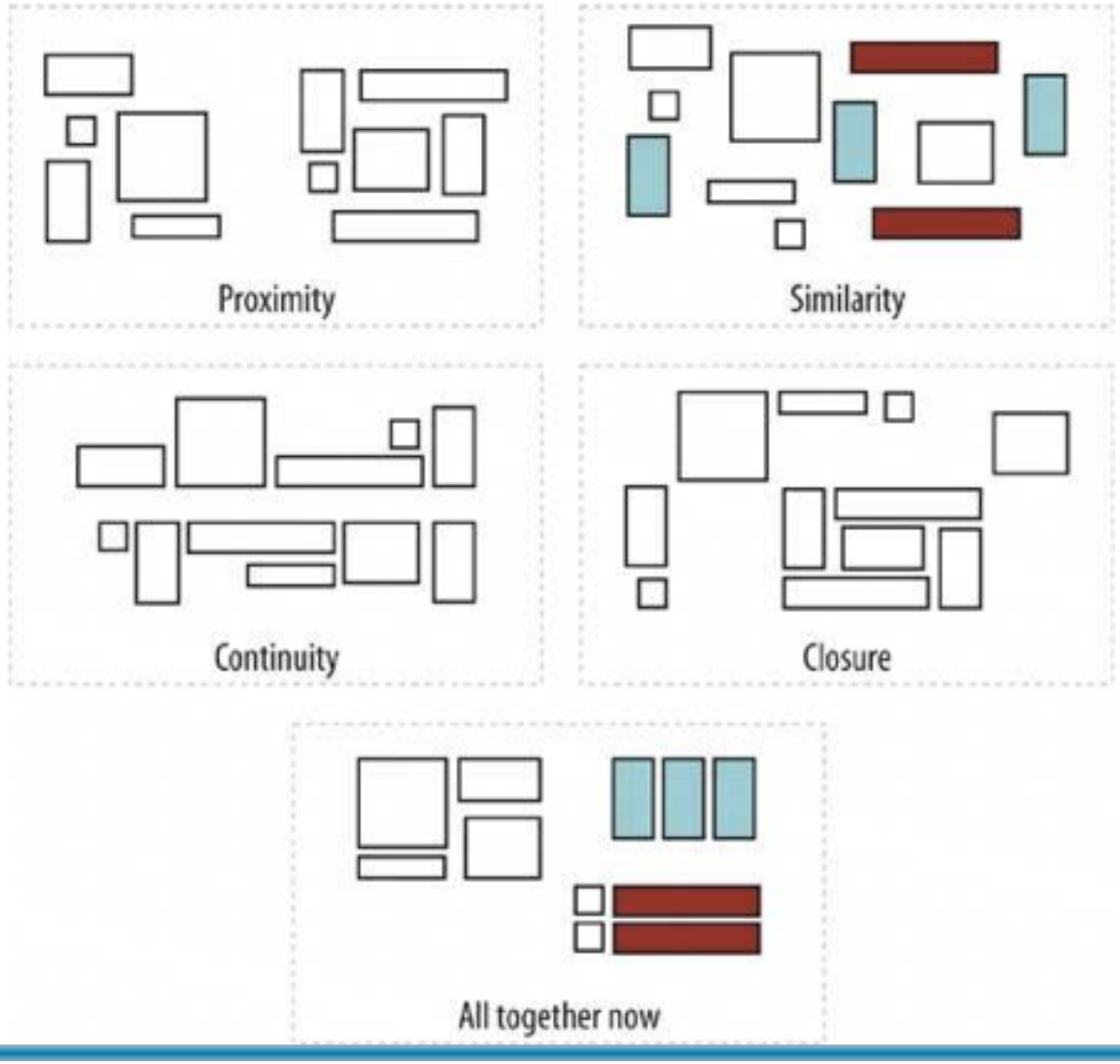
How you organise your application data and actions will help you determine how to organise your windows

- **Two-Panel Selector**, used in Mail, Finder
- **Canvas + Palette**, used in Illustrator, Photoshop
- **One-Window Drill-down**, often used on smartphones
- **Alternative Views**, used in Word, PowerPoint, Finder
- **Wizards**, for software installation, printer setup, internet purchases
- **Extras on Demand**, to hide complexity when not needed
- **Multi-Level Help**, different ways to access help

Navigation Patterns

- **Clear Entry Points** - few options on startup
- **Global Navigation** - repeated motif/toolbar on each window
- **Hub and Spoke** - organises discrete tasks from central hub
- **Pyramid** - use back/next/up to organise documents
- **Modal Panel** - dialog box that must be resolved now
- **Sequence Map** - “you are here” indicator in a sequence
- **Breadcrumbs** - for hierarchies, show parent pages
- **Annotated Scrollbar** - provide information in scrollbar
- **Colour-coded Sections** - use colour to identify locations
- **Animated Transition** - show next location by animation
- **Escape Hatch** - cancel current action

Gestalt Principles



UI Layout Design Patterns

- **Visual Framework** - consistency between windows/dialogs in your application
- **Centre Stage** - make the most important window large
- **Titled Sections** - delineate categories with obvious titles
- **Card Stack** - use tabs to organise information
- **Closable Panels** - tabs that can dynamically resize
- **Movable Panels** - let user move panels around the window
- **Right/Left Alignment** - align columns to provide vertical lines
- **Diagonal Balance** - balance from top-left to bottom-right
- **Responsive Disclosure** - show only what needs to be shown
- **Responsive Enabling** - allow only relevant responses
- **Liquid Layout** - change window contents during resize