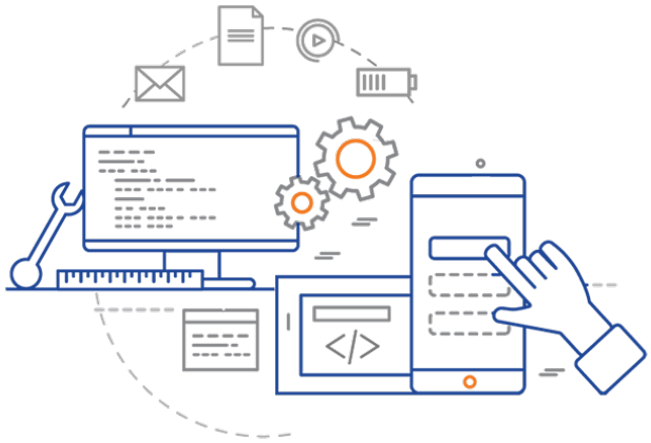

CS 398: Application Development

Week 04 Lecture: Analysis & Design 4

Design patterns

This Week

Mon (today)

- Lectures: UX Design & Prototyping; Software design principles; SOLID principles.
- Activities: Sketching/prototyping

Wed

- Lectures: Design Patterns
- Activities: Consider which patterns are relevant
  - Identify 2 patterns that you plan to use.
  - Mention them in your presentation (2-3 sentences listing them and describing their relevance).
  - Add them to your system diagram.

Fri

- Lectures: software design video <— NOT testable, optional.
- Activities: Design review
  - 10 min presentation, followed by 5 min Q&A session.
  - Everyone must be present and everyone must present *something*.
  - See the website for your time. One of the course staff will call you on your team channel at that time.

Review

Design Patterns

A design pattern is a generalizable software solution to a common problem.

Using a known design pattern provides you with a template for a well-designed solution, which may be superior to a home-grown solution. It also gives you common-ground for design discussions.

Patterns originated with Christopher Alexander, an architect, in 1977. Design patterns in software gained popularity with the book Design Patterns: Elements of Reusable Object-Oriented Software [Gamma 1994].

Creational Patterns

Creational Patterns control the dynamic creation of objects.

Pattern	Description
Abstract Factory	Provide an interface for creating families of related or dependent objects without specifying their concrete classes.
★ Builder	Separate the construction of a complex object from its representation, allowing the same construction process to create various representations.
Prototype	Specify the kinds of objects to create using a prototypical instance, and create new objects from the ‘skeleton’ of an existing object, thus boosting performance and keeping memory footprints to a minimum.
★ Singleton	Ensure a class has only one instance, and provide a global point of access to it.

Structural Patterns

Structural Patterns are about organizing classes to form new structures.

Pattern	Description
★ Adapter, Wrapper	Convert the interface of a class into another interface clients expect. An adapter lets classes work together that could not otherwise because of incompatible interfaces.
Bridge	Decouple an abstraction from its implementation allowing the two to vary independently.
★ Composite	Compose objects into tree structures to represent part-whole hierarchies. Composite lets clients treat individual objects and compositions of objects uniformly.
Decorator	Attach additional responsibilities to an object dynamically keeping the same interface. Decorators provide a flexible alternative to subclassing for extending functionality
Proxy	Provide a surrogate or placeholder for another object to control access to it.

Behavioural Patterns

Behavioural Patterns are about identifying common communication patterns between objects.

	Pattern	Description
	Command	Encapsulate a request as an object, thereby allowing for the parameterization of clients with different requests, and the queuing or logging of requests. It also allows for the support of undoable operations.
	Iterator	Provide a way to access the elements of an aggregate object sequentially without exposing its underlying representation.
	Memento	Without violating encapsulation, capture and externalize an object's internal state allowing the object to be restored to this state later.
	Observer	Define a one-to-many dependency between objects where a state change in one object results in all its dependents being notified and updated automatically.
	Strategy	Define a family of algorithms, encapsulate each one, and make them interchangeable. Strategy lets the algorithm vary independently from clients that use it.
	Visitor	Represent an operation to be performed on the elements of an object structure. Visitor lets a new operation be defined without changing the classes of the elements on which it operates.

Activities

TODO Today

Planning

1. Create project plan

Requirements

1. Pick users, (optional) create personas
2. Interview people that fall into your role
3. Identify requirements, (affinity diagram)
4. Document requirements in GitLab

Week 4 quiz is due Friday by 11:59 PM.

Design Review is Friday!
See the online schedule for your time slot.
10 min presentation.

Analysis & Design

1. Determine technical impact
2. Choose architectural style
3. System diagram
4. UI Mockup (Low-fidelity Prototype)
5. Design Patterns
   - Pick 2 to use in your design, and add them to your design review.