Design Pattern

- Patterns give a shared vocabulary
- Patterns recur in many application
- Adapter
    - **Converts the interface** of a class into another interface clients expect
    - Use an adapter where you need to use an existing class and its interface is not the one you have

- Facade
    - The pattern provides a unified interface to a set of interfaces in a subsystem
    - High-level interface that makes the subsystem easier to use
    - One advantage of facade - decoupling
    - Loose coupling

- Singleton
    - The pattern ensures you have at most one instance of a class in your application
    - A "singleton" is a class that manages an instance of **itself**
    - Lazy initialization:
        - the instance is only created when the static method is first called
    - “**public static** synchronized Singleton getInstance()” (synchronized keyword prevents two threads from entering the method at the same time)

- Iterator (tested in quiz 4)
    - The pattern allows a way to access the elements of an aggregate object **sequentially** without showing its underlying representation

- Template method (tested in quiz 4)
    - Define the skeleton of an algorithm in an operation, deferring some steps to sub-classes
    - Let subclasses redefine certain steps of an algorithm without changing the algorithm’s structure
    - Abstract methods

- Composite (tested in quiz 4)
    - Compose objects into tree structures to represent part-whole hierarchies
    - Let clients treat individual objects and compositions of objects uniformly

- State
    - Allows an object to alter its behavior when its internal state changes its class
    - The object will appear to change its class
- Proxy
    - Provides a surrogate placeholder for another object to control access to it

- Observer
    - One-to-many dependency between objects so that when object changes state, all its dependents are notified and updated automatically

- Strategy
    - Define a family of algorithms, encapsulate each one and makes them interchangeable.

- Factory
    - Defines an interface for creating an object but lets subclasses decide which class to instantiate

- Decorator
    - Attach additional responsibilities to an object dynamically

- Command
    - Encapsulates a request to an object letting you parameterize clients with different requests
    - Developer Integration Testing sometimes is also called - DIT

    - The MVC is a **compound** pattern
        - Consisting of
            - Observer
                - Listen to “button” class
                - Observers are dependent on the objects
                - Observers like to be **notified** when **updates** happen
            - Composite
            - Strategy
        - The model of MVC makes use of the **observer** pattern
        - The View in MVC uses **composite** pattern to implement the user interface, which usually consists of nested components like panels, frames and buttons.

    - the Controller in MVC uses **strategy** pattern because the View can use different implementations of the controller to get different behavior.

- **"Good enough" design is better than "perfect design".**

Guest Lecture

- What makes a successful software engineer
    - Managing complexity
    - Requirement, planning, integration, testing, design
    - SWE > Programmer
- Software engineering is all about **managing complexity**
- Programming is about **getting code to work**

- Avoid “clever” code
- Design principles matter
    - Abstraction
    - Encapsulation
    - Strong Cohesion
    - Loose Coupling
    - Modular Design
    - Object-oriented
    - Inheritance

    - The most successful app from our guest speaker was based on a jigsaw puzzle game

    - Advertise your app in another APP
    - Two main things to evaluate a project
        - **MVC**
        - **Layer**

    - Over-engineering
        - There is a **cost** to SW engineering processes
        - Make sure the cost is not higher than it yields

    - Personal Relations
        - Your attitude is critically important
        - Best SW Engineer ever + bad attitude = **dead career**
            - Constantly negative
            - Overly sarcastic / snarky
        - Nobody will want to work with you
            - Including people who make hiring or promotion decisions
        - Never respond to email while angry
            - Wait for 24 hours

        - Write your cathartic response but do not send
        - People remember negative interactions for a long time

- Personal Relation cont..
    - The best colleagues..
        - Humble
        - Make time for you
        - Don’t make you feel dumb
        - Say “hi” to people :)

- Work Ethic
    - A good work ethic is necessary to career success
    - A bad work ethic is just as toxic as a bad attitude
    - Be proactive. Break up your work into small milestones to keep from getting behind
    - Be curious
        - Read up on new technologies languages, etc.

- Enterprise Systems
    - Not all software is **consumer-based**
    - Business (Enterprise)
    - Turnkey systems
    - Databases, order management, account services, customer services
    - Other Devs may be your customer

- Your Career
    - You should enjoy what you do
    - Do not be afraid to change
    - You do not have to be a SW engineer

- Come out of your shell
    - More personal interaction as your career advances
    - Meetings, code reviews, work with manager designers, QA
    - You will be expected to participate
    - Presentations
    - Building software is collaborative

        - Do not stress. It is a skill like any other that can be improved with practice

- Non-Standard hours
    - Releases & client deployments
        - Weekend “rollouts”
        - “On-call”
        - Shifts
    - Production outages
        - Emergency work

- Time Management
    - Set aside work time
        - Blick time on your calendar
    - Plan your day first
        - Look at calendar
        - Do not dive right into work
    - Handling interruptions
        - Keep context switching low
        - Respond to emails at set time per day
        - Acceptable response time is 24 hours