COMP354

Software Engineering

Lecture 14

Design Patterns

Design Patterns

Micro-architectures that resolve issues of design

offer tentative solutions in *issue-driven design*

Facade — simplifies interface to a subsystem

hides internal structure of subsystem

Example: most subsystems

Observer — decouples depender from dependee

allows multiple, dynamic dependency

provides protocol for change notification

Example: visual widgets in Java Swing are updated when notified

Composite — makes a compound object look like a single object

especially useful for recursive compound object

Example: document trees (as in XML)

Mediator — manages complicated dependencies

when no one dependee is responsible for change notification

Design Patterns Descriptions

1.The motivation or context that this pattern applies to.

2.Prerequisites that should be satisfied before deciding to use a pattern.

3.A description of the program structure that the pattern will define.

4.A list of the participants needed to complete a pattern.

5.Consequences of using the pattern...both positive and negative.

6.Examples!

Design Patterns — Gang of Four Catalog — Creational

The Abstract Factory is a type of design pattern that is used for creating groups of dependent instances of classes without specifying their concrete classes.

The Singleton Design Pattern makes sure that at any point of time, there is one and only one instance of a class present and also provides a global point of access to the object.

The Factory Design Pattern delegates the work of object creation to derived classes of the interface.

The Builder Design Pattern separates the creation of an instance of a class from its representation so that the same creation pattern can be used for creating different representations of the object.

The Prototype Design Pattern creates new instances of classes by copying its prototype. Micro-architectures that resolve issues of design

Design Patterns — Gang of Four Catalog — Structural

The Bridge Design Pattern separates an abstract interface from its implementation so that both the interface and its implementation can change without any dependency between each other.

The Decorator Design Pattern is used when additional functionality needs to be added to objects dynamically.This pattern provides an alternative to subclassing for extending an object's functionality.

The Flyweight Design Pattern is used to share a large numbers of small objects efficiently.

The Adapter Design Pattern is a type of design pattern that is used for converting the interface of a class into an interface that its clients expect to see. This pattern allows incompatible interfaces to work together.

The Composite Design Pattern is used to compose objects so that they can be represented in part-whole hierarchies in tree-structures. This pattern allows clients to treat individual objects equally.

The Facade Design Pattern is used to provide a high-level interface that makes the subsystem easier to use. It helps create a unified interface to a set of interfaces in the subsystem.

The Proxy Design Pattern is used is used when it is required to use another object as a substitute to control access to this object.

Design Patterns — Gang of Four Catalog — Behavioral

The Command Design Pattern helps to encapsulate each request as an object. This helps in queing or logging requests and helps perform difficult operations.

The Iterator Design Pattern is used to access elements of an aggregated object sequentially without exposing its representation.

The Memento Design Pattern helps design how to capture an object's internal state so that an object can be restored to that state later.

The Mediator Design Pattern is used to design encapsulation for interaction among objects. It helps design a loose coupling by preventing objects from referring to each other directly. This allows to vary their interactions independently.

The Interpreter Design Pattern helps interpret sentences in any given language, provided that the grammar for the language is provided.

The Chain of Responsibility Design Pattern avoids tying the sender of a request tightly to its receiver. It helps design chains of receiving objects and pass the request along the chain until an object handles the request.

The Observer Design Pattern is used to design and define a one-to-many relationship between objects so that when one object changes state, all its dependents are notified and updated automatically.