Jim Fawcett
CSE687 – Object Oriented Design
Spring 2010

DESIGN SUMMARY

Design Summary

- Principles
  - Liskov Substitution
  - Open Closed
  - Dependency Inversion
  - Interface Segregation
  - Least knowledge
- Design Techniques
  - Encapsulate – Single Responsibility
  - Parameterize – Separate library design from application design
  - Hook – Install base class as parent for application code
  - Isolate – provide both interface and object factory
- Design Process
  - Distinguish between application side decomposition and solution side re-composition

Liskov Substitution

- Clients typed to use base pointers or references can use derived pointers or references with no knowledge of the derived details.
- Support Liskov Substitution by:
  - Providing virtual base functions
  - Virtual base destructor
  - Avoid use of dynamic_cast
  - Don’t overload virtual functions or across class scopes

Open Closed Principle

- Reusable software entities should be open for extension but closed for modification.
- Support Open Closed Principle by
  - Using template parameterization.
  - Providing Hook base classes
- Example:
  - XmlDocument prototype

Dependency Inversion

- Software clients and servers should not depend on each others details. They should both depend on the server's abstraction.
- Support Dependency Inversion by:
  - Providing interface with protocol language that supports all server operations.
  - Provide class factory that instantiates server objects on the client's behalf.
- Example:
  - Parser – uses Rules derived from IRules and created by Builder

Interface Segregation

- Don’t make clients depend on interface methods they don’t need.
- Support Interface Segregation by:
  - Segregating interfaces by functionality
    - Each interface supports a specific model
  - Classes implement just those interfaces they need to support their requirements.

Least Knowledge

- Client callers know only the calling interface, and none of the service implementation.
- Service responders know nothing of the caller beyond the contents of the request.
- To support Least Knowledge:
  - Apply dependency inversion
  - Pass messages

Encapsulation

- A class should manage completely its own data and resources.
  - Clients should have no access to its internal implementation.
  - This prevents clients from putting class instance into invalid state.
- Enforce Encapsulation with:
  - Private access control of all private member functions and data.
  - Expose only encapsulated parts.
  - Use no global data.

Parameterize

- Distinguish between application design and library design.
  - Parameterize reusable library classes with class and member template arguments.
  - Make library code more flexible be including template functions that use compiler type inference to accept a variety of argument types.
- Example:
  - Tracer class from MT3Q1 and MT3Q1b, Sp2010

Hook

- Install base class as hook for application code
  - Hook provides a base protocol used by a library class.
  - Applications derive from the hook and register classes with the hook provider to support application operations.
- Example:
  - Navigator class in FileInfo folder

Isolate

- Build components that can be composed to build large complex systems
  - Components support modifying part of the system without rebuilding unmodified parts.
- Support Isolation:
  - Use dependency inversion
  - Package as a Dynamic Link Library (DLL)
- Example:
  - DLLProtocolDemo in class code folder

Design Process

- Distinguish between Application and Solution side development:
  - Application side development decomposes project requirements into a set of application specific classes that model the application entities.
  - Solution side development recomposes the project with reusable classes that support the application processing.
  - We care about different things on each side.

The End