

Design Patterns for Mere Mortals



Philip Japikse
Phil.japikse@pinng.com
MVP, MCSD, MCDBA, CSM, CSP
Principal Consultant
Pinnacle Solutions Group

ST. LOUIS DAY OF DOT NET

WE WOULD LIKE TO THANK OUR SPONSORS

PLATINUM

QUILOGY[®]
The Art & Science of Business[®]

Microsoft



GOLD

TALX | Provider of EQUIFAX
WORKFORCE SOLUTIONS

Scottrade

SILVER



telerik
deliver more than expected

Infragistics
Powering The Presentation Layer

SYLLOGISTEKS
Simplifying Business with Technology

BRONZE

DAUGHERTY
Business Solutions

ENVISION

ST. LOUIS DAY OF DOT NET



Who am I?

- Principal Consultant, Pinnacle Solutions Group, Inc.
- Microsoft MVP, MCSD, MCDBA, CSM, CSP
- Enterprise Application Architect
- Trainer/Mentor
- National and Regional Speaker
- Director, Cincinnati .NET User's Group

“The goal is not to bend developers to the will of some specific patterns, but to get them to *think about their work and what they are doing*”

--Phil Haack

SOLID

- S – Single Responsibility Principle
 - There should never be more than one reason for a class to change
- O – Open/Closed Principle
 - Classes should be Open for extension, Closed for modification
- L - Liskov Substitution Principle
 - Derived classes should be substitutable for their base classes
- I - Interface Segregation Principle
 - Make fine grained interfaces that are client specific
- D - Dependency Inversion Principle
 - Depend on abstractions, not concrete implementations

DRY and SOC

- Don't Repeat Yourself
- Separation of Concerns

What are Design Patterns?

- General Reusable Solutions To A Common Problem
 - Conceptual
 - Templates
 - Defined by Purpose and Structure
 - Method of Communication
- Support SOLID development
- NOT CODE!

Types of Design Patterns

- **Creational**
 - Deal with instantiation of objects
- **Structural**
 - Deal with Composition and Relations
- **Behavioral**
 - Deal with responsibilities and communication between objects

Creational

- Singleton
 - Ensures class has only one instance
- Factory Types
 - Simple Factory (Not a “true” pattern)
 - Encapsulates Object Creation in one place
 - Factory Method
 - Allow subclasses to determine what is initiated
 - Abstract Factory
 - Provides an interface for creating families of related objects without specifying their concrete class
- Demo

Structural

- Adapter
 - Converts the interface of a class into another interface the client expects
- Decorator
 - Attaches additional responsibilities to an object at runtime
- Façade
 - Provides a unified interface to a set of interfaces
- Demo

Behavioral

- Command
 - Encapsulates a request as an object
- Strategy
 - Encapsulates an algorithm inside a class
- Demos

Resources

- “Design Patterns: Elements of Reusable Object Oriented Design”
 - Eric Gamma, Richard Helm, Ralph Johnson, John Vlissides
- “Head First Design Patterns”
- www.dofactory.com

Contact Me

- phil@skimed.com
- www.pinnsg.com
- www.japikse.blogspot.com
- www.twitter.com/skimedic
- (513) 619-6323

Questions?

