Applying Design Patterns to AutoCAD .NET Application Development

James E. Johnson
Sr. Application developer
Twitter : @jamescad

Class summary

Design patterns provide reusable solutions to common software issues that occur frequently when doing software design. This class is an introduction for .NET developers on how to use design patterns applications. The 23 common design patterns known as ‘GoF’ patterns are considered the base for other design patterns. This class will demonstrate how to use some common patterns with AutoCAD software add-in code samples.

Key learning objectives

At the end of this class, you will :

- Have a basic understanding of where and why to use design patterns.
- Have seen how to apply design pattern code and refactoring existing code for design patterns.
- Know how design patterns can help you build better applications
- Have sample code using design patterns that can be immediately used in your applications.

Introduction to Design Patterns

Introduction

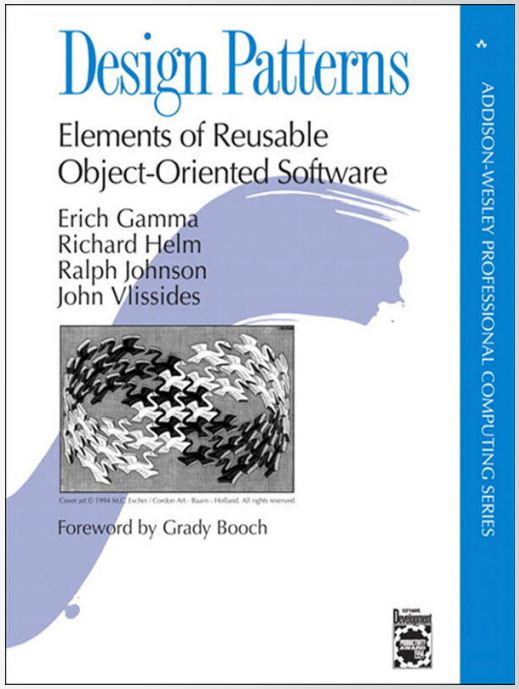
- Design Patterns are generally described as reusable software solutions to recurring problems in application development.
- Consider Design Patterns as “Recipes” or “Templates” for solving software development problems.

Introduction

- Design Patterns are an outline, or considered a “best” or “preferred” way to solve a problem.
- They have evolved from the need to solve the same problem by many people over time.
- They can be considered a common frame of reference on how to solve a problem.

Introduction

- The book :
  "*Design Patterns: Elements of Reusable Object- Oriented Software* (Addison-Wesley)" by Erich Gamma, Richard Helm, Ralph Johnson, and John Vlissides outlines and describes 23 commonly used design patterns.



Introduction

- The authors of that book are commonly referred to as the “Gang of Four (GOF)”.
- In the 20+ years since that book was published there have been numerous additional books published.
- There are many websites and “youtube” videos that have been dedicated to discussing and showing sample code for Design Patterns.

Introduction

- The (GOF) patterns are considered the foundation for other patterns and are categorized into three types:
  - Creational Patterns
  - Structural Patterns
  - Behavioral Patterns

Introduction

- **Creational Patterns:** Provide ways to instantiate single objects or groups of related objects.
- Defined patterns are:
  - Abstract Factory
  - Builder
  - Factory Method
  - Prototype
  - Singleton

Introduction

- **Structural Patterns:** Provide ways to define relationships between classes or objects.
- Defined patterns:
  - Adapter
  - Bridge
  - Composite
  - Decorator
  - Facade
  - Flyweight
  - Proxy

Introduction

- **Behavioral Patterns:** Provide communication between classes and objects.
- Defined patterns:
  - Chain of Responsibility
  - Command
  - Interpreter
  - Iterator
  - Mediator
  - Memento
  - Observer
  - State
  - Strategy
  - Template Method
  - Visitor

Getting Started with Design Patterns

Getting Started with Design Patterns

- Does not require any additional tools.
- Does not require any extraordinary programming skill or capabilities.

- Requires knowledge of Object Oriented Programming.
- Requires an understanding to definitions of terms like data types, polymorphism, interface and inheritance

Getting Started with Design Patterns

- **Data types:** A programming classification identifying various types of data.
- **Polymorphism:** A programming concept that provides the ability of objects to take on multiple types.
- **Inheritance:** An object or class can be based on another object or class. Inheritance allows creating new classes that reuse, extend, and modify the behavior that is defined in other classes.

Getting Started with Design Patterns

- **Interface:** An interface contains the signatures of methods, properties, events or indexers.
- **Encapsulation:** The process of binding data members (variables, properties) and functions (methods) into a single unit.
- **IS-A Relationship:** This can be accomplished with Class inheritance or Interface inheritance.
- **HAS-A Relationship:** This is done by using instance variables that reference other objects.

Understanding where and why to use design patterns

- Design patterns can shorten the development process.
- Prevent issues that can cause major problems.
- Learn and experiment with creating sample code for as many design patterns as possible.
- Never approach a software design with trying to fit it into using a ‘particular’ design pattern.

Look at Code examples…

Abstract Factory Pattern

- The Abstract Factory is a Creational Design Pattern
- The abstract factory pattern is used to provide an interface for a group of similar factories or dependent objects without specifying the concrete classes.
- Sample code…

Factory Method Pattern

- The Factory Method Pattern is a Creational Design Pattern.
- The factory method pattern is used to defer instantiation with class constructors replaced into subclasses.
- Sample Code…

Builder Pattern

- The Builder Pattern is a Creational Design Pattern.
- The builder pattern is used to separate the creation of complex objects from its representation to enable the construction process to create different representations.
- Sample Code…

Strategy Pattern

- The Strategy pattern is a Behavioral Pattern.
- The strategy pattern defines a family of algorithms, encapsulate each one, and make them interchangeable.
- Sample Code…

Command Pattern

- The Command Pattern is a Behavioral Design Pattern,
- The command pattern is used encapsulate a request as an object, thereby letting you parameterize clients with different requests, queue or log requests, and support undoable operations. The command may then be executed immediately or held for later use.
- Sample Code…

Singleton Pattern

- The Singleton Pattern is a Creational Design Pattern.
- The singleton pattern is used to ensure that only one instance of a particular class is ever created and provides access to the object.
- Often considered an Anti-Pattern due to overuse and that it makes the instance in a ‘Global’ state that can cause problems in some applications.

Applying design pattern code and refactoring existing code for design patterns

- Starting a new project using design patterns is often difficult as projects evolve from test code or existing code that often does not use design patterns.
- Create projects the same way that you have always worked and then refactor using design patterns.
- Do **Not** select a pattern and start designing around that pattern, this will often produce undesirable outcome.

How design patterns can help you build better applications

- **Readability:** creates code that is better understood.
- **Maintainability:** makes the code easier to maintain.
- **Communication:** provides a common vocabulary.
- **Intention:** code is easier to understand.
- **Re-use:** common solutions to common problems. assists in code that is easier to use for other solutions.
- **Less code:** derived for common functionality.
- **Time Tested:** provides proven and sound solutions.

Have sample code using design patterns that can be immediately used in your applications.

- Along with handout several sample applications that have been demonstrated have been copied for download.

- Samples are basic C# and are not intended for deploying without additional refinement.

Thanks for Attending…

Be heard! Provide AU session feedback.

- Via the Survey Stations, email or mobile device.
- AU 2016 passes awarded daily!
- Give your feedback after each session.
- Give instructors feedback in real-time.

Forget to take notes? No problem!

After AU visit:
AutodeskUniversity.com

Click on **My AU** to find:

- Class Recordings
- Presentations
- Handouts

All of your sessions will be there to enjoy again and again.

Instruction Manuals Outdated?

Visit:

AutodeskUniversity.com

Click on **My AU** to start building your own desk reference (with materials from this decade).

Learn something worth sharing?

After AU visit:

AutodeskUniversity.com

Click on **My AU** to share your AU experience with:

- Colleagues
- Peers
- Professionals

Save hundreds of sessions worth sharing.

Too many sessions, too little time?

After AU visit:

AutodeskUniversity.com

- Recorded sessions
- Presentations and handouts
- Key learnings

Don’t miss a second! Find hundreds of sessions waiting for you.

More Questions? Visit the AU Answer Bar

- Seek answers to all of your technical product questions by visiting the **Answer Bar**.
- Open daily 8am-10am and Noon-6pm and located just outside of Hall C on Level 2.
- Staffed by Autodesk developers, QA, & support engineers ready to help you through your most challenging technical questions.