The Object-Oriented Design of the Expression Tree Processing App

Douglas C. Schmidt
Learning Objectives in This Lesson

- Understand the OO design of the expression tree processing app.

en.wikipedia.org/wiki/Unified_Modeling_Language has more on OOD notations.
Lesson Introduction
Lesson Introduction

• Object-oriented design (OOD) is a method of planning a system of interacting objects to solve software problem(s).

[Diagram showing Object-Oriented Design, Polymorphism, Extensibility, Abstraction, and Encapsulation]

en.wikipedia.org/wiki/Object-oriented_design has more information on OO design.
Object-oriented design (OOD) is a method of planning a system of interacting objects to solve software problem(s).

OOD employs “hierarchical data abstraction.”

Components are designed based on stable class & object roles & relationships

Rather than functions corresponding to actions

C++ Iterator

Level_Order Iterator

In_Order Iterator

Post_Order Iterator

Pre_Order Iterator

en.wikipedia.org/wiki/Liskov_substitution_principle has more information.
Object-oriented design (OOD) is a method of planning a system of interacting objects to solve software problem(s).

OOD employs “hierarchical data abstraction.”

It also associates actions with specific objects and/or classes of objects.

Emphasize *high cohesion* & *low coupling*

[en.wikipedia.org/wiki/Low-Coupling_/__High-Cohesion_pattern](en.wikipedia.org/wiki/Low-Coupling_/__High-Cohesion_pattern) has more information.
Lesson Introduction

• Object-oriented design (OOD) is a method of planning a system of interacting objects to solve software problem(s).
• OOD employs “hierarchical data abstraction.”
• It also associates actions with specific objects and/or classes of objects.
• Well-designed OO programs group classes & objects via patterns & combine them to form frameworks.

www.dre.vanderbilt.edu/~schmidt/patterns-frameworks.html has more information.
OO Design of Expression Tree Processing App

Douglas C. Schmidt
• Create an OO design based on modeling classes & objects in “expression tree” domain.
• Conduct *scope, commonality, & variability* analysis to determine stable APIs & variable extension points.

Conduct *scope, commonality, & variability* analysis to determine stable APIs & variable extension points.

- Model a *tree* as a collection of *nodes.*
• Conduct *scope, commonality, & variability* analysis to determine stable APIs & variable extension points.

• Model a *tree* as a collection of *nodes*.

**OO Design of Expression Tree Processing App**

*Note the different types of nodes in a tree.*
OO Design of Expression Tree Processing App

- Conduct *scope, commonality, & variability* analysis to determine stable APIs & variable extension points.
- Model a *tree* as a collection of *nodes*.
- Represent *nodes* as class hierarchy, capturing properties of each node.
- e.g., the “arities” (binary & unary nodes)

See [en.wikipedia.org/wiki/Arity](en.wikipedia.org/wiki/Arity)
Apply “Gang of Four” (GoF) patterns to guide the development of a framework of extensible classes.

en.wikipedia.org/wiki/Design_Patterns has information on the “Gang of Four” (GoF) book.
• Apply “Gang of Four” (GoF) patterns to guide the development of a framework of extensible classes.

• A **framework** is an integrated set of software components that collaborate to provide a reusable architecture for a family of related applications.

See [www.dre.vanderbilt.edu/~schmidt/frameworks.html](http://www.dre.vanderbilt.edu/~schmidt/frameworks.html)
OO Design of Expression Tree Processing App

- Apply “Gang of Four” (GoF) patterns to guide the development of a framework of extensible classes.
- A framework is an integrated set of software components that collaborate to provide a reusable architecture for a family of related applications.
- Frameworks exhibit three characteristics that differentiate them from other forms of systematic reuse.

Application-Specific Functionality

[Image of a sculpture and puzzle pieces]

[Link: www.dre.vanderbilt.edu/~schmidt/reuse-lessons.html] has info on systematic reuse.
• Apply “Gang of Four” (GoF) patterns to guide the development of a framework of extensible classes.

• A framework is an integrated set of software components that collaborate to provide a reusable architecture for a family of related applications.

• Frameworks exhibit three characteristics that differentiate them from other forms of systematic reuse.
  1. Inversion of control (IoC)

Apply “Gang of Four” (GoF) patterns to guide the development of a framework of extensible classes.

A framework is an integrated set of software components that collaborate to provide a reusable architecture for a family of related applications.

Frameworks exhibit three characteristics that differentiate them from other forms of systematic reuse.

1. Inversion of control (IoC)
   - The framework controls the main execution thread

Application-Specific Functionality
• Apply “Gang of Four” (GoF) patterns to guide the development of a framework of extensible classes.

• A framework is an integrated set of software components that collaborate to provide a reusable architecture for a family of related applications.

• Frameworks exhibit three characteristics that differentiate them from other forms of systematic reuse.
  1. Inversion of control (IoC)
     • The framework controls the main execution thread
     • Decides how/when to run app code via callbacks

See en.wikipedia.org/wiki/Callback_(computer_programming)
• Apply “Gang of Four” (GoF) patterns to guide the development of a framework of extensible classes.

• A framework is an integrated set of software components that collaborate to provide a reusable architecture for a family of related applications.

• Frameworks exhibit three characteristics that differentiate them from other forms of systematic reuse.

  1. Inversion of control (IoC)
     • The framework controls the main execution thread
     • Decides how/when to run app code via callbacks
     • e.g., an Android looper dispatches a handler, which then dispatches a runnable

See blog.mindorks.com/android-core-looper-handler-and-handlerthread-bd54d69fe91a
• Apply “Gang of Four” (GoF) patterns to guide the development of a framework of extensible classes.

• A framework is an integrated set of software components that collaborate to provide a reusable architecture for a family of related applications.

• Frameworks exhibit three characteristics that differentiate them from other forms of systematic reuse.

  1. Inversion of control (IoC)
    • The framework controls the main execution thread
    • Decides how/when to run app code via callbacks
    • IoC is often called “The Hollywood Principle”

See www.dre.vanderbilt.edu/~schmidt/Coursera/articles/hollywood-principle.txt
Apply “Gang of Four” (GoF) patterns to guide the development of a framework of extensible classes.

A framework is an integrated set of software components that collaborate to provide a reusable architecture for a family of related applications.

Frameworks exhibit three characteristics that differentiate them from other forms of systematic reuse.

1. Inversion of control (IoC)
2. Domain-specific structure & functionality

See [en.wikipedia.org/wiki/Domain-driven_design](en.wikipedia.org/wiki/Domain-driven_design)
Apply “Gang of Four” (GoF) patterns to guide the development of a framework of extensible classes.

A framework is an integrated set of software components that collaborate to provide a reusable architecture for a family of related applications.

Frameworks exhibit three characteristics that differentiate them from other forms of systematic reuse.

1. Inversion of control (IoC)
2. Domain-specific structure & functionality
   - e.g., capabilities that can be reused in 1+ domain(s)
Apply “Gang of Four” (GoF) patterns to guide the development of a framework of extensible classes.

A *framework* is an integrated set of software components that collaborate to provide a reusable architecture for a family of related applications.

Frameworks exhibit three characteristics that differentiate them from other forms of systematic reuse.

1. Inversion of control (IoC)
2. Domain-specific structure & functionality
   - e.g., capabilities that can be reused in 1+ domain(s)

Application-specific functionality can systematically reuse framework components.
Apply “Gang of Four” (GoF) patterns to guide the development of a framework of extensible classes.

A framework is an integrated set of software components that collaborate to provide a reusable architecture for a family of related applications.

Frameworks exhibit three characteristics that differentiate them from other forms of systematic reuse.

1. Inversion of control (IoC)
2. Domain-specific structure & functionality
3. Semi-complete applications
• Apply “Gang of Four” (GoF) patterns to guide the development of a framework of extensible classes.

• A **framework** is an integrated set of software components that collaborate to provide a reusable architecture for a family of related applications.

• Frameworks exhibit three characteristics that differentiate them from other forms of systematic reuse.
  1. Inversion of control (IoC)
  2. Domain-specific structure & functionality
  3. Semi-complete applications
    • *Hook methods* plug app logic into the framework

See [codebetter.com/davelaribee/2008/06/16/hook-methods](http://codebetter.com/davelaribee/2008/06/16/hook-methods)
Apply “Gang of Four” (GoF) patterns to guide the development of a framework of extensible classes.

A framework is an integrated set of software components that collaborate to provide a reusable architecture for a family of related applications.

Frameworks exhibit three characteristics that differentiate them from other forms of systematic reuse.

1. Inversion of control (IoC)
2. Domain-specific structure & functionality
3. Semi-complete applications
   - Hook methods plug app logic into the framework
   - Mediate interactions among common abstract & variant concrete classes/interfaces

Application-Specific Functionality

e.g., Java Runnable is an abstract interface providing basis for concrete variants.
• Integrate pattern-oriented language & library features with frameworks.
• Both an app-specific framework...

    Expression_Tree tree = ...;
    Visitor print_visitor = ...;

    for (auto iter = tree.begin(order);
         iter != tree.end(order);
         ++iter)
        (*iter).accept(print_visitor);

*Factory Method, Bridge, Composite, Iterator, Strategy, & Visitor patterns*

This app-specific framework exhibits high pattern density!
OO Design of Expression Tree Processing App

- Integrate pattern-oriented language & library features with frameworks.
- Both an app-specific framework... & off-the-shelf frameworks...

See developer.android.com & en.wikipedia.org/wiki/Standard_Template_Library
Complexity resides in (stable) structure & APIs, rather than (variable) algorithms.
