Design Patterns in C++: Chain of Responsibility to Memento

CHAIN OF RESPONSIBILITY

Dmitri Nesteruk
QUANTITATIVE ANALYST

@dnesteruk http://activemesa.com

Course Overview

4th course in a series of courses on C++ Design Patterns

- Covers 1st half of behavioral design patterns (Chain of Responsibility to Memento)

Covers every pattern from GoF book

- Motivation
- Classic implementation
- Pattern variations
- Library implementations
- Pattern interactions
- Important considerations (e.g., testability)

Patterns demonstrated via live coding!

Demo

Uses modern C++ (C++11/14/17)

Demos use Microsoft Visual Studio 2015, MSVC, ReSharper C++

Some simplifications:

- Classes are often defined inline (no .h/.cpp separation)
- Pass by value
- Liberal import of namespaces (e.g., `std::`) and headers

Course Structure

Chain of Responsibility

Command

Interpreter

Iterator

Mediator

Memento

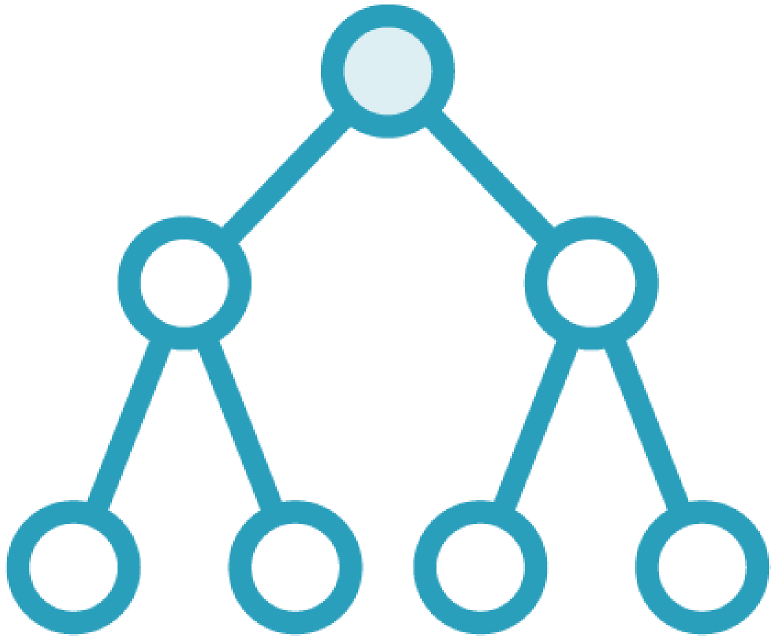
Overview

Motivation

Command Query Separation

Pointer Chain

Broker Chain



Unethical behavior by an employee; who takes the blame?

- Employee
- Manager
- CEO

You click a graphical element on a form

- Button handles it, stops further processing
- Underlying group box
- Underlying window

CCG computer game

- Creature has attack and defense values
- Those can be boosted by other cards

Chain of Responsibility

A chain of components who all get a chance to process a command or query, optionally having a default processing implementation and an ability to terminate the processing chain.

Command Query Separation

Command = asking for an action or change (e.g., please set your attack value to 2).

Query = asking for information (e.g., please give me your attack value).

CQS = having separate means of sending commands and queries; antithetical to e.g., direct field access.

Summary

Chain of Responsibility can be implemented as a pointer chain or a centralized construct

Enlist objects in the chain, possibly controlling their order

Remove object from chain when no longer applicable (e.g., in its own destructor)