Design Patterns in C++: Structural - Adapter to Decorator

ADAPTER

Dmitri Nesteruk
QUANTITATIVE ANALYST

@dnesteruk http://activemesa.com

Course Overview

Second course in a series of courses on C++ Design Patterns

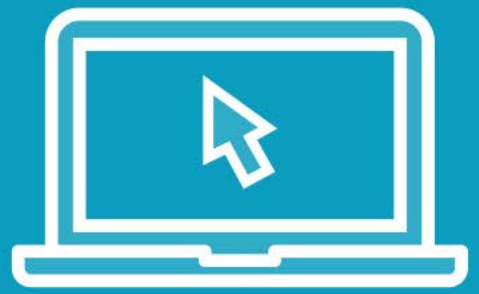
- Covers half of structural design patterns (Adapter to Decorator)

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 everywhere
- Liberal import of namespaces (e.g., `std::`) and headers

Course Structure

Adapter

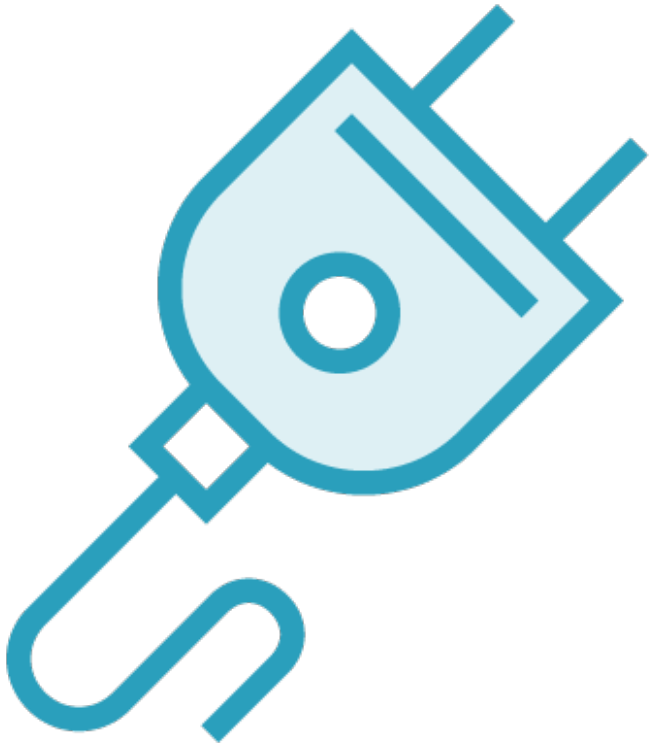
Bridge

Composite

Decorator

Overview

- Motivation
- Stack as an Adapter
- Simple String Adapter
- Vector/Raster Demo
- Adapter Caching



Electrical devices have different power (interface) requirements

- Voltage (5V, 220V)
- Socket/plug type (Europe, UK, USA)

We cannot modify our gadgets to support every possible interface

- Some support possible (120/220V)

Thus, we use a device (an adapter) that gives us the interface we require

Adapter

A construct which adapts an existing interface X to conform to the required interface Y.

Summary

- Implementing an Adapter is easy
- Determine the API you have and the API you need
- Create a component which aggregates (has a reference to, ...) the adaptee
- Intermediate representations can pile up; use caching and other optimizations