Module 07: "Adapter"

Agenda

- Introductory Example: Computing Areas of Shapes
- Challenges
- Implementing the Adapter Pattern
- Pattern: Adapter
- Overview of Adapter Pattern
- Object vs. Class Adapters

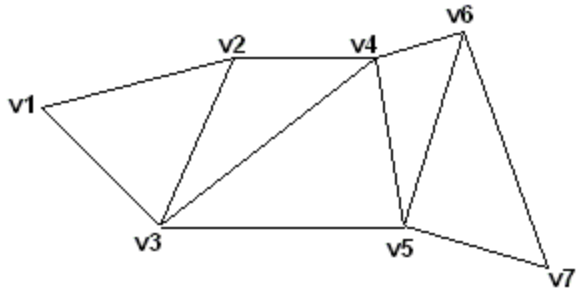
Introductory Example: Computing Areas of Shapes

```
public interface IAreaCalculator
{
    double Compute( Rectangle rectangle );
}
```

```
public class ShapeProcessor
{
    ...
    public double GetArea( TriangleStrip ts )
    {
        ...
    }
}
```

Background: Triangle Strips

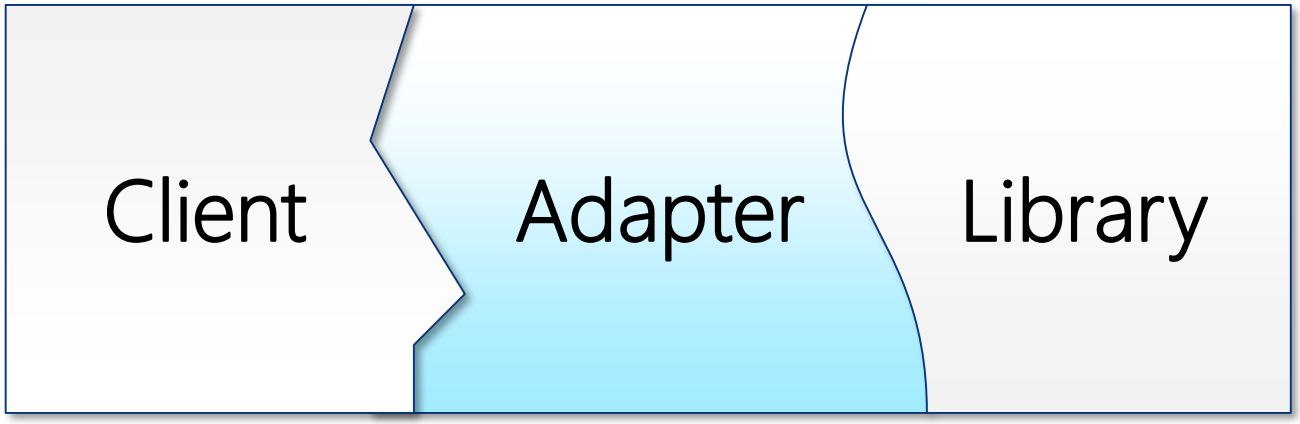
- Used low-level by graphics cards and APIs in
  - 2D or 3D



  - See https://msdn.microsoft.com/en-us/library/windows/desktop/bb206274(v=vs.85).aspx

Challenges

- Problems:
  - Client interface does not match what Library provides



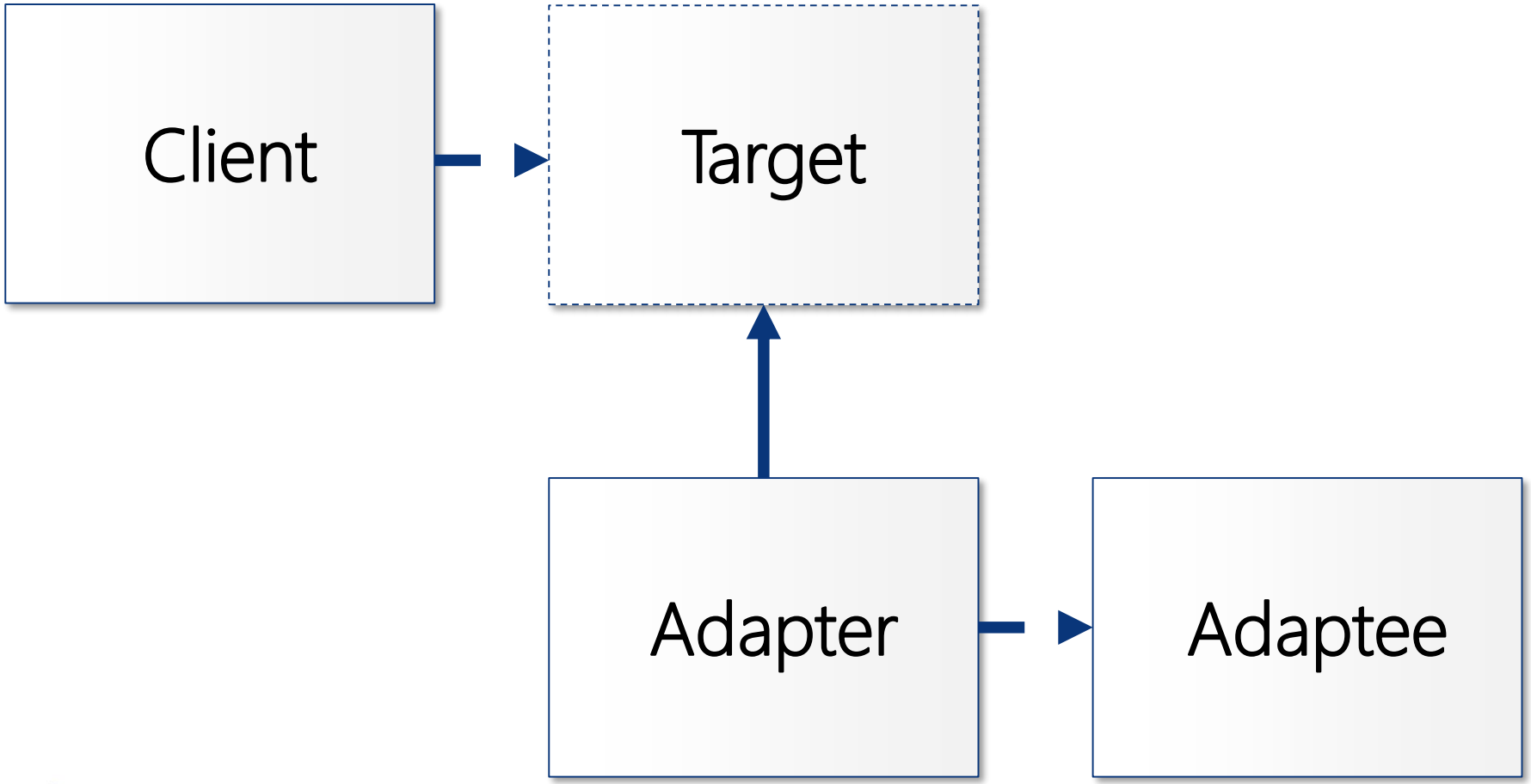
Pattern: Adapter

- *Convert the interface of a class into another interface clients expect. Adapter lets classes work together that couldn't otherwise because of incompatible interfaces.*

- Outline
  - Adapt Client interface to Adaptee interface
  - Adapter implements Target interface and invokes Adaptee
  - Potentially: Also loosely couple or future-proof Client

- Origin: Gang of Four

Overview of Adapter Pattern



Overview of the Adapter Pattern

- Adaptee
  - Existing interface, abstract, or concrete class that needs adapting
- Client
  - Concrete component communicating via the Target interface
- Target
  - Interface or abstract class that the Client expects
- Adapter
  - Concrete class exposing the Target interface to Client
  - Implements adaptation by invoking Adaptee operations

Object Adapters

- Object Adapters use composition

```
class ShapeProcessorAdapter : Client.IAreaCalculator
{
    private Library.ShapeProcessor _adaptee;

    public ShapeProcessorAdapter( Library.ShapeProcessor adaptee )
    {
        _adaptee = adaptee;
    }

    public double Compute( Rectangle rectangle )
    {
        ...
        return _adaptee.GetArea(triangles);
    }
}
```

Class Adapters

- Class Adapters (when possible) use inheritance

```
class ShapeProcessorClassAdapter :
   Library.ShapeProcessor, Client.IAreaCalculator
{
    public ShapeProcessorAdapter()
    {
    }

    public double Compute( Rectangle rectangle )
    {
        ...
        return GetArea(triangles);
    }
}
```

Object vs. Class Adapters

- Which approach is better…?
- … and why?


WINCUBATE
Jesper Gulmann Henriksen
PhD, MCT, MCSD, MCPD
Phone : +45 22 12 36 31
Email : jgh@wincubate.net
WWW : http://www.wincubate.net
Hasselvangen 243
8355 Solbjerg
Denmark