Module 17:
"Strategy"

Agenda

▸ Introductory Example: Running a Bar

▸ Challenges

▸ Implementing the Strategy Pattern

▸ Pattern: Strategy

▸ Overview of Strategy Pattern

▸ Variation: Strategies as Delegates

▸ .NET Framework Example: Sorting Collections

Introductory Example: Running a Bar

```csharp
enum Billing
{
    Normal,
    StudentDiscount,
    Regular
}
```

```csharp
class Customer
{
    public BarTab Tab { get; }
    public Customer( Billing billing ) { ... }
    public void PlaceOrder( Order order ) { ... }
}
```

```csharp
Customer customer = new Customer( Billing.Normal );

customer.PlaceOrder( new Order { Product = new Peanuts(), Count = 1 });

customer.PlaceOrder( new Order { Product = new Beer(), Count = 3 });

customer.PlaceOrder( new Order { Product = new PepsiMax(), Count = 2 });


customer.Tab.Print();
```

Challenges

▸ What if a new Billing options would be introduced?
- Happy Hour?
- Code will throw exception!

▸ Have to manually extend switch statement!

▸ Need to change other(!) classes

▸ Breaks the Open/Closed Principle

▸ A lot of ugly, unnecessary coupling!

Pattern: Strategy

▸ *Define a family of algorithms, encapsulate each one, and make them interchangeable. Strategy lets the algorithm vary independently from clients that use it.*

▸ Outline
- Avoid unnecessary coupling
- Configure a class with one of a family of algorithms at run-time
- Strategy object implements algorithm

▸ Origin: Gang of Four

Overview of Strategy Pattern

Component

Strategy

Execute()

Execute()

Concrete Strategy

Execute()

Concrete Strategy
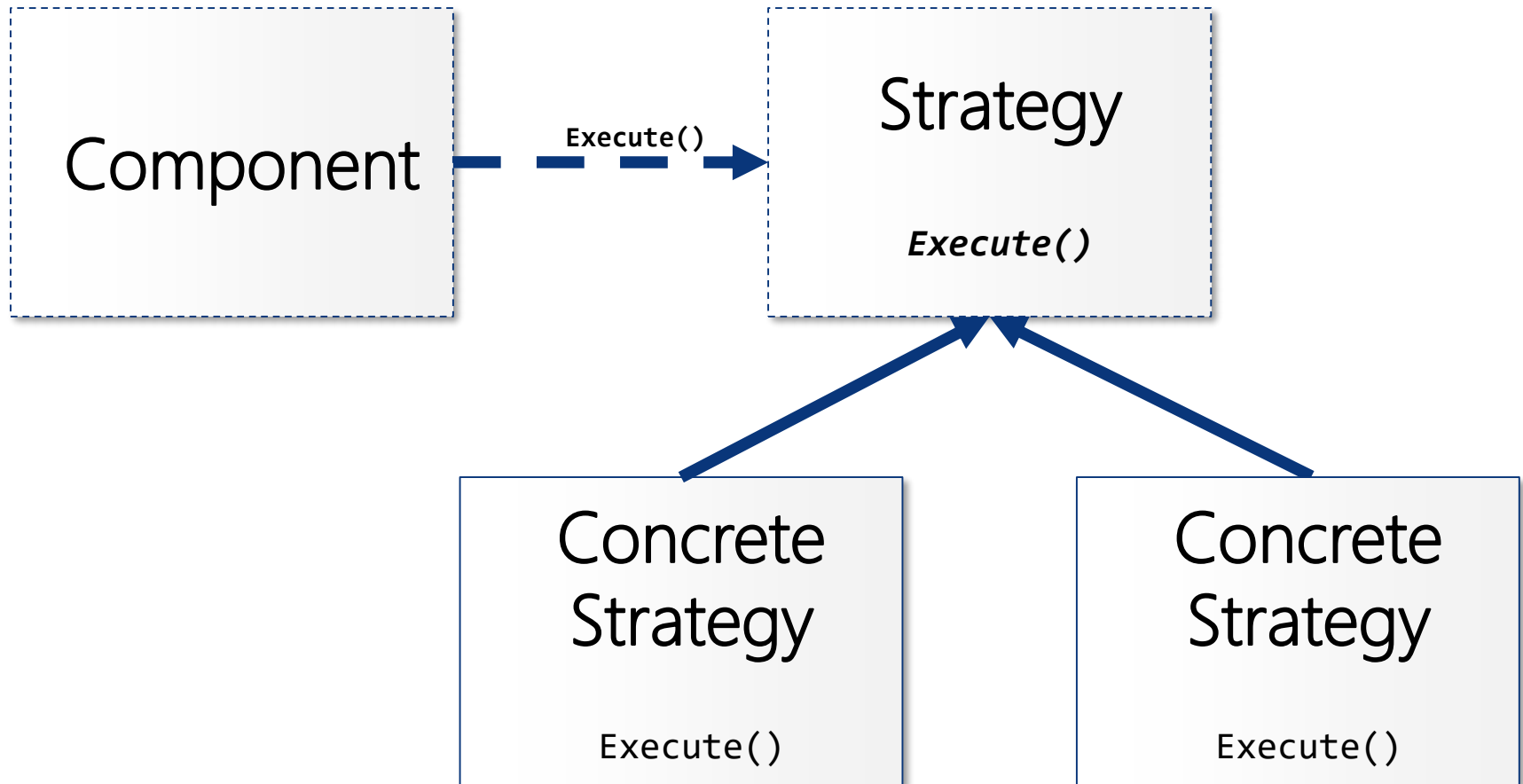
Execute()

Overview of Strategy Pattern

▸ Component
- Concrete class parameterized by a Strategy supplied to it
- Employs the Strategy by invoking `Execute()` whenever needed

▸ Strategy
- Interface or abstract base class for algorithm declaring abstract `Execute()` method

▸ Concrete Strategy
- Implements a concrete strategy in the `Execute()` method

Variation: Strategies as Delegates

▸ Strategies are essentially stateless "algorithm" objects

▸ In .NET we can implement Strategy using delegates
  • Method names
  • Anonymous Methods
  • Lambda Expressions

▸ Can either be
  • Injected into constructors
  • Passed as method arguments
    • Easier to change dynamically

Strategy vs. Template Method

▶ Strategy
- Based on Composition
- Can be change at run-time
- No dictated algorithm structure

▶ Template Method
- Based on Inheritance
- Can be changed at compile-time only
- Fixed predefined set of algorithm steps
  - Some can be refined
- Can have a default pre-implemented functionality

.NET Framework Example: Sorting Collections

▸ Sorting collections implements comparisons as Strategy
- **IComparable** or **IComparable<T>** are strategy interfaces

```csharp
abstract class Product : IComparable<Product>
{
    public abstract string Name { get; }
    public abstract decimal SuggestedPrice { get; }

    public int CompareTo( Product other )
    {
        if (SuggestedPrice < other.SuggestedPrice) { return -1; }
        else if (SuggestedPrice > other.SuggestedPrice) { return 1; }
        else { return 0; }
    }
}
```

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