Software Architecture & Design SEC3071

Lecture No. 35

Muhammad Shahid
Department of Computer Science
National Textile University

shahid.abdullah@hotmail.com

Last Lecture Review

- Creational Pattern
- Factory Method Pattern
- Factory Pattern – Class Diagram
- Practical Example - Pizza Store

Agenda – What will you Learn Today?

Adapter Design Pattern

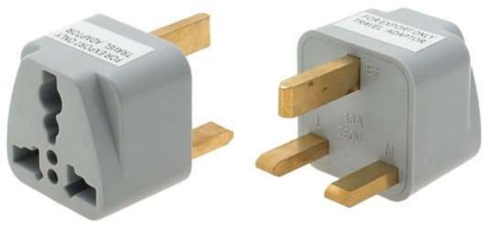
Structural Design Pattern

Structural Design Patterns

- Deals with how classes and objects deal with to form large structures
- Structural Design patterns use inheritance to compose interfaces or implementations
- Structural Design Patterns basically ease the design by identifying the relationships between entities

Structural Design Patterns

- Deals with objects delegating responsibilities to other objects
- This behavior results in a layered architecture of components with low degree of coupling
- Facilitate inter-object communication when one object is not accessible to the other by normal means or when an object is not usable because of its incompatible interface



Adapter Design Pattern

Adapter Design Pattern

- Clients of a class access the services offered by the class through its interface
- Sometimes, an existing class may provide the functionality required by a client, but its interface may not be what the client expects

Adapter Design Pattern

- This could happen due to various reasons such as:
  - – The existing interface may be too detailed
  - – It may lack in detail
  - – The terminology used by the interface may be different from what the client is looking for

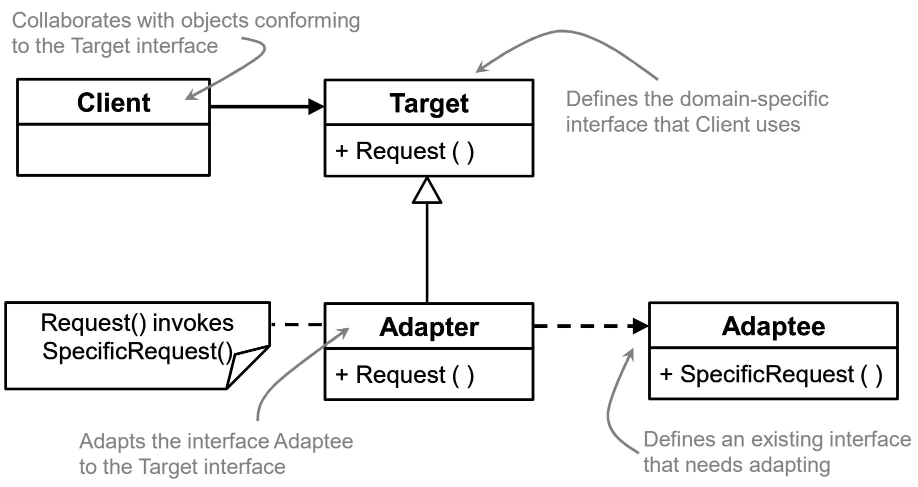
Adapter Design Pattern – Defined

"Adapter pattern convert the interface of the class into a form what client expects. Adapter let the classes work together which couldn't otherwise due to incompatible interfaces."

Applicability – When to Use

- We want to use the existing class and its interface does not match with the one you need
- In case of reusable classes due Non-Compatible interfaces it is not possible to reuse them

Adapter Pattern – Class Diagram


Collaborates with objects conforming to the Target interface
Client
Target
+ Request ( )
Defines the domain-specific interface that Client uses
Request() invokes SpecificRequest()
Adapter
+ Request ( )
Adaptee
+ SpecificRequest ( )
Adapts the interface Adaptee to the Target interface
Defines an existing interface that needs adapting

Chemical Bank

- National chemicals is a well known company which manufacture chemicals and do research on compounds. Each compound has boiling point, melting point, molecular weight and molecular formula. The system uses a legacy chemical databank. Chemical compound system has to access the chemical databank but compound system are not compatible with chemical databank. We need to develop an application through which both classes can communicate.

Code Implementation

```
public class Compound
{
       public string chemical;
        public float boilingPoint;
        public float meltingPoint;
        public double molecularWeight;
        public string molecularFormula;
       public Compound(string chemical)
       {
              this.chemical = chemical;
       }
       public virtual void Display()
       {
              Console.WriteLine("Compound:{0}", chemical);
       }
} // End of Compound class
```

Code Implementation

```
public class ChemicalDatabank
{
        public float GetMeltingPoint(string compound)
        {
                switch (compound.ToLower())
                {
                        case "water":       return 100.0f;
                        case "benzene":     return 80.1f;
                        case "ethanol":     return 78.3F;
                        default:            return 0f;
                }
        } // End of GetMeltingPoint method
```

Code Implementation

```
        public float GetBoilingPoint(string compound)
        {
                switch (compound.ToLower())
                {
                        case "water":       return 0.0f;
                        case "benzene":     return 5.5f;
                        case "ethanol":     return -114.1F;
                        default:            return 0f;
                }
        } // End of GetBoilingPoint method
```

Code Implementation

```
public string GetMolecularFormula(string compound)
{
        switch (compound.ToLower())
        {
                case "water":       return "H20";
                case "benzene":     return "C6H6";
                case "ethanol":     return "C2H5OH";
                default:            return " ";
        }
} // End of GetMolecularFormula method
```

Code Implementation

```
        public double GetMolecularWeight(string compound)
        {
                switch (compound.ToLower())
                {
                        case "water":       return 18.015;
                        case "benzene":     return 78.1134;
                        case "ethanol":     return 46.0688;
                        default:            return 0d;
                }
        } // End of GetMolecularWeight method

} // End of ChemicalDatabank class
```

Code Implementation

```
public class RichCompound : Compound
{
    ChemicalDatabank bank = new ChemicalDatabank();

    public RichCompound(string chem):base(chemical) { }

    public override void Display()
    {
        boilingPoint = bank.GetBoilingPoint(chem);
        meltingPoint = bank.GetMeltingPoint(chem);
        molecularWeight = bank.GetMolecularWeight(chem);
        molecularFormula = bank.GetMolecularFormula(chem);
```

Code Implementation

```
    //Displaying the information
        base.Display();
        Console.Write("Melting Point:{0}", meltingPoint);
        Console.Write("Boiling Point:{0}", boilingPoint);
        Console.Write("Weight:{0}", molecularWeight);
        Console.Write("Formula:{0}", molecularFormula);
    }

} // End of RichCompound class
```

Code Implementation

```
static void Main(string[] args)
{
        RichCompound water = new RichCompound("water");
        water.Display();

        RichCompound benzene = new RichCompound("benzene");
        benzene.Display();

        RichCompound ethanol = new RichCompound("ethanol");
        ethanol.Display();
}
```

Code Implementation

```
Compound:                   water
Melting Point:              100
Boiling Point:              0
Weight:                     18.015
Formula:                    H20
```

```
Compound:                   benzene
Melting Point:              80.1
Boiling Point:              5.5
Weight:                     78.1134
Formula:                    C6H6
```

```
Compound:                   ethanol
Melting Point:              78.3
Boiling Point:              -114.1
Weight:                     46.0688
Formula:                    C2H5OH
```

Recap

- Structural Design Patterns
- Adapter Design Pattern
  - Definition
  - Applicability
  - Class Diagram
  - Implementation
- Adapter Pattern – Examples
  - Calculating Square
  - Chemical Bank

Questions

?