Software Architecture & Design SEC3071

Lecture No. 36

Muhammad Shahid
Department of Computer Science
National Textile University

shahid.abdullah@hotmail.com

Last Lecture Review

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

Agenda – What will you Learn Today?

Façade Design Pattern

Façade Design Pattern

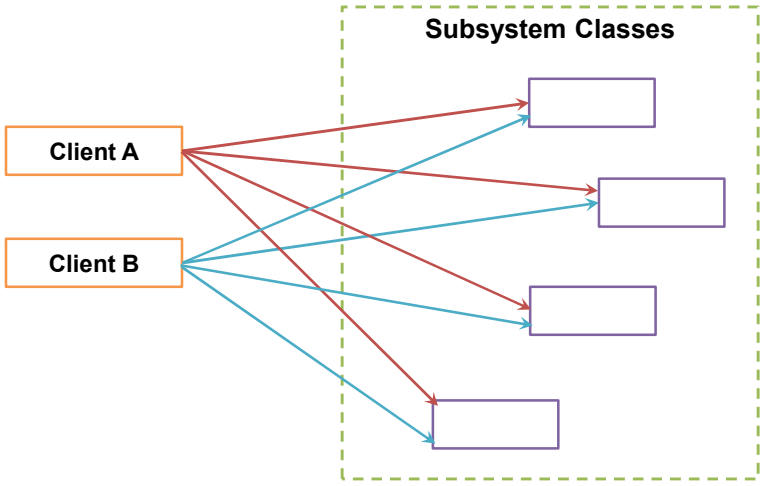
Façade Design Pattern Defined

"Façade provides a unified interface to a set of interfaces in a subsystem. It define a higher level interface which is easier to use"

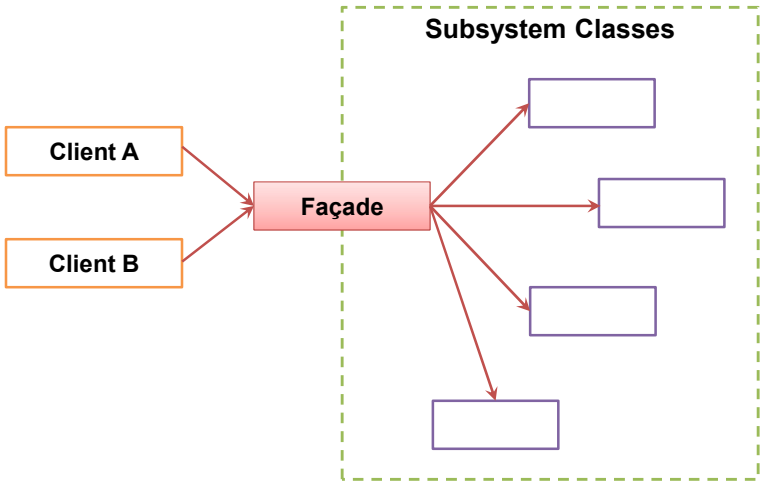
Façade Design Pattern Defined

- Façade decouple the client from interacting with the subsystems instead Façade take up the responsibility of dealing with the subsystems itself
- Façade will not add any extra functionality it will just simply the access to functionality
- Client can also access subsystems directly as if there is no Facade

Client Access without Facade



Client Access with Facade



Design Principle

- The **Principle of Least Knowledge**

"Talk only to your immediate friends"

- When creating software design for any object be careful of the number of classes it is interacting with and how it will be interacting with them

Guidelines for Implementing Principle

- Suppose we have an object with several methods, now for that object we should invoke methods only that belongs to:
  1) An Object itself
  2) Object passed in as a parameter
  3) Any method that object creates or instantiates
  4) Any component of the Object

The Principle of Least Knowledge

```
public class Car
{
	Engine engine;
	public Car() { }    // Constructor
	public void Start(Key key)// Start() function
	{
		Doors door = new Doors;
		boolean authorized = key.truns();
		if(authorized)
		{
			engine.Start();
			UpdateDashboardDisplay();
			doors.Lock();
		}
	} // End of Start() function
} // End of class
```


The Principle of Least Knowledge

```
public class Car
{
	Engine engine;
	public Car() { }    // Constructor
	public void Start(Key key)// Start() function
	{
		Doors door = new Doors;
		boolean authorized = key.truns();
		if(authorized)
		{
			engine.Start();
			UpdateDashboardDisplay();
			doors.Lock();
		}
	} // End of Start() function
} // End of class
```

Here is component of this class. We can call its methods

You can call a method on a object passed as parameter

Here we are creating a new object, its methods are legal.

You can call a component of the object

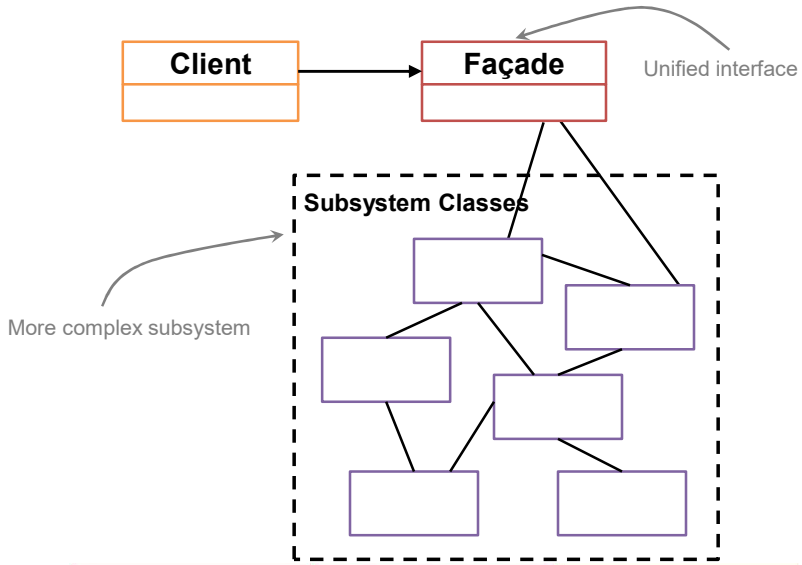
You can call a local method within the object

You can call a method on an object you create or instantiate

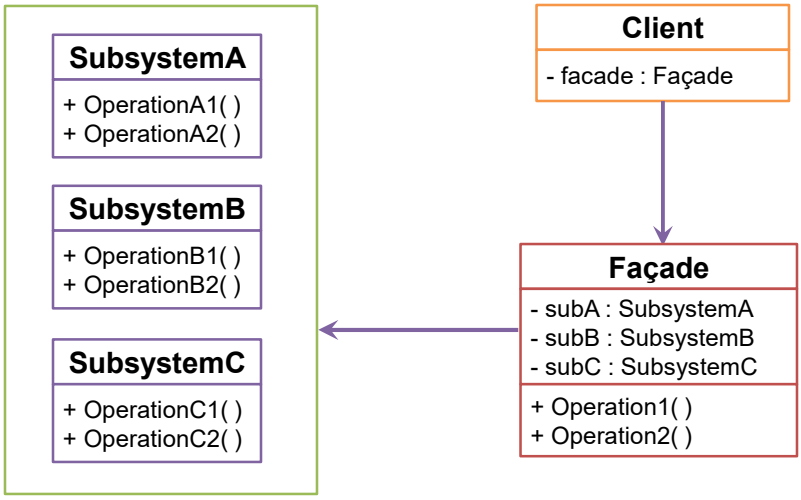
Applying Principle in Facade

- There can be several Façade within One Façade with the increase in complexity
- We aim to maintain minimum possible communication with other classes

Façade Pattern – Class Diagram

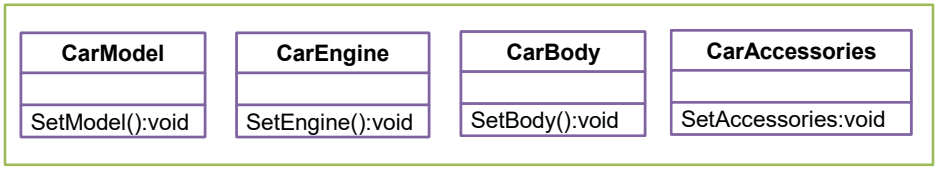


Façade Pattern – Implementation

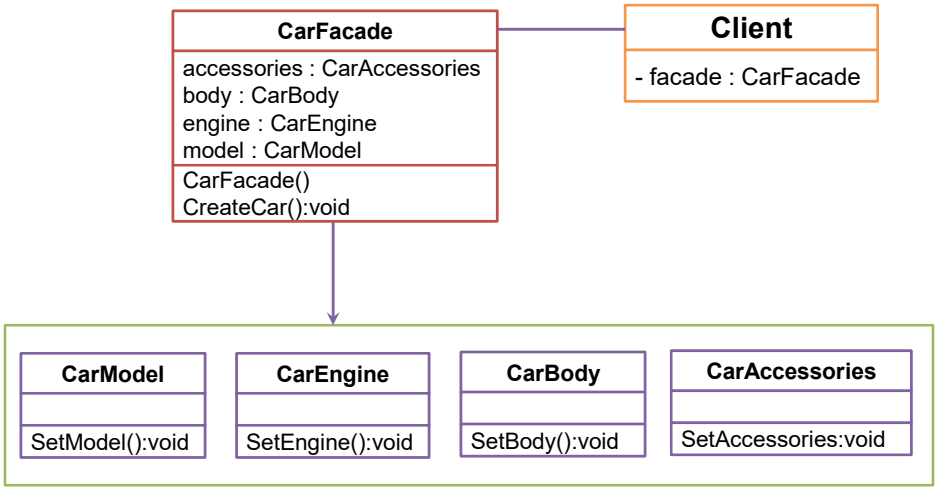


Building a Car

Building a Car



Building a Car



Building a Car

```
public class CarModel
{
	public string model;

	public string Model
	{
		get { return model; }
		set { model = value; }
	}

} // End of CarModel class
```


Building a Car

```
public class CarEngine
{
	public string engine;

	public string Engine
	{
		get { return engine; }
		set { engine = value; }
	}

} // End of CarEngine class
```


Building a Car

```
public class CarBody
{
	public string body;

	public string Body
	{
		get { return body; }
		set { body = value; }
	}

} // End of CarBody class
```


Building a Car

```
public class CarAccessories
{
	public string accessories;

	public string Accessories
	{
		get { return accessories; }
		set { accessories = value; }
	}

} // End of CarAccessories class
```


Building a Car

```
public class CarFacade
{
    public CarModel model;
    public CarEngine engine;
    public CarBody body;
    public CarAccessories access;

    public CarFacade()
    {
        model = new CarModel();
        engine = new CarEngine();
        body = new CarBody();
        access = new CarAccessories();
    }
```


Building a Car

```
    public void CreateCar(string model, string engine,
                          string body, string accessories)
    {
        model.Model = model;
        engine.Engine = engine;
        body.Body = body;
        access.Accessories = accessories;
    }
```


Building a Car

```
    public void DisplayCar()
    {
        Console.Write("-------- Car Information -------");
        Console.Write("Model:{0}", model.Model);
        Console.Write("Engine:{0}", engine.Engine);
        Console.Write("Body:{0}", body.Body);
        Console.Write("Accessories:{0}",access.Accessories);
    }
} // End of CarFaçade class
```


Building a Car

```
static void Main(string[ ] args)
{
        CarFacade facade = new CarFacade();
        facade.CreateCar("Hinda City", "1300 CC", "Solid-
        White", "Antena – High lights");
        facade.DisplayCar();
}
```

```
-------------- Car Information --------------
   Model:                Honda City
   Engine:               1300 CC
   Body:                 Solid - White
   Accessories:          Antenna - High lights
```


Recap

- Structural Design Patterns
- Façade Design Pattern
- Client Access without & without Facade
- The Principle of Least Knowledge (PLK)
- Applying PLK in Facade
- Façade Pattern – Class Diagram
- Façade Pattern – Implementation
- Façade DP Example - Building a Car

Questions