|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | Samreen Zehra | **Student ID** | SP17-MSCS-0015 |
| **Course** | Design Patterns | **Date of Submission** | April 09, 2018 |

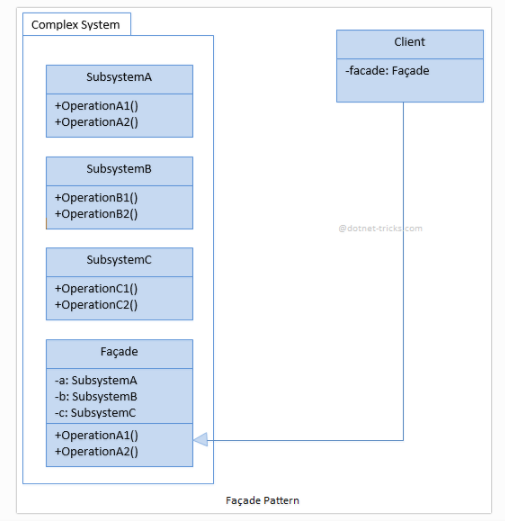
**Assignment No 3**

# **Façade pattern**

**Intent:** Provide a unified interface to a set of interfaces in a subsystem. Facade defines a higher-level interface that makes the subsystem easier to use.

The facade pattern is typically used when

* a simple interface is required to access a complex system,
* a system is very complex or difficult to understand,
* an entry point is needed to each level of layered software, or
* the abstractions and implementations of a subsystem are tightly coupled.



**Example 1: MortgageApplication**

Participants

The classes and objects participating in this pattern are:

1. **Facade (MortgageApplication)** knows which subsystem classes are responsible for a request and delegates client requests to appropriate subsystem objects.
2. Subsystem classes **(Bank, Credit, Loan)** :

* implement subsystem functionality.
* handle work assigned by the Facade object.
* have no knowledge of the facade and keep no reference to it.

using System;

namespace Facade

{

using System;

class MainApp

{

static void Main()

{

// Facade

Mortgage mortgage = new Mortgage();

// Evaluate mortgage eligibility for customer

Customer customer = new Customer("Ann McKinsey");

bool eligible = mortgage.IsEligible(customer, 125000);

Console.WriteLine("\n" + customer.Name + " has been " + (eligible ? "Approved" : "Rejected"));

// Wait for user

Console.ReadKey();

}

}

/// The 'Subsystem ClassA' class

class Bank

{

public bool HasSufficientSavings(Customer c, int amount)

{

Console.WriteLine("Check bank for " + c.Name);

return true;

}

}

/// The 'Subsystem ClassB' class

class Credit

{

public bool HasGoodCredit(Customer c)

{

Console.WriteLine("Check credit for " + c.Name);

return true;

}

}

/// The 'Subsystem ClassC' class

class Loan

{

public bool HasNoBadLoans(Customer c)

{

Console.WriteLine("Check loans for " + c.Name);

return true;

}

}

/// Customer class

class Customer

{

private string \_name;

// Constructor

public Customer(string name)

{

this.\_name = name;

}

// Gets the name

public string Name

{

get { return \_name; }

}

}

/// The 'Facade' class

class Mortgage

{

private Bank \_bank = new Bank();

private Loan \_loan = new Loan();

private Credit \_credit = new Credit();

public bool IsEligible(Customer cust, int amount)

{

Console.WriteLine("{0} applies for {1:C} loan\n",

cust.Name, amount);

bool eligible = true;

// Check creditworthyness of applicant

if (!\_bank.HasSufficientSavings(cust, amount))

{

eligible = false;

}

else if (!\_loan.HasNoBadLoans(cust))

{

eligible = false;

}

else if (!\_credit.HasGoodCredit(cust))

{

eligible = false;

}

return eligible;

}

}

}

**Example 2: CarManufacturing**

Participants

The classes and objects participating in this pattern are:

1. **CaeFacade**  knows which subsystem classes are responsible for a creating a car and delegates client requests to appropriate subsystem objects.
2. Subsystem classes **(Model, Engine, Body, Accessories)** :

namespace Facade

{

// The 'Subsystem ClassA' class

class CarModel

{

public void SetModel()

{

Console.WriteLine(" CarModel - SetModel");

}

}

/// The 'Subsystem ClassB' class

class CarEngine

{

public void SetEngine()

{

Console.WriteLine(" CarEngine - SetEngine");

}

}

// The 'Subsystem ClassC' class

class CarBody

{

public void SetBody()

{

Console.WriteLine(" CarBody - SetBody");

}

}

// The 'Subsystem ClassD' class

class CarAccessories

{

public void SetAccessories()

{

Console.WriteLine(" CarAccessories - SetAccessories");

}

}

// The 'Facade' class

public class CarFacade

{

private readonly CarAccessories accessories;

private readonly CarBody body;

private readonly CarEngine engine;

private readonly CarModel model;

public CarFacade()

{

accessories = new CarAccessories();

body = new CarBody();

engine = new CarEngine();

model = new CarModel();

}

public void CreateCompleteCar()

{

Console.WriteLine("\*\*\*\*\*\*\*\* Creating a Car \*\*\*\*\*\*\*\*\*\*");

model.SetModel();

engine.SetEngine();

body.SetBody();

accessories.SetAccessories();

Console.WriteLine("\*\*\*\*\*\*\*\* Car creation is completed. \*\*\*\*\*\*\*\*\*\*");

}

}

// Facade pattern demo

class Program

{

static void Main(string[] args)

{

var facade = new CarFacade();

facade.CreateCompleteCar();

Console.ReadKey();

}

}

}

**Example 3: ComputerFacade**

Participants

The classes and objects participating in this pattern are:

1. **ComputerFacade** knows which subsystem classes are responsible for a managing Computer operations and delegates client requests to appropriate subsystem objects.
2. Subsystem classes **(CPU, Memory, HarDrive)** :

class CPU

{

public void freeze()

{

Console.WriteLine(" CPU - freeze");

}

public void jump(long position)

{

Console.WriteLine(" CPU - jump to {0}"+ position);

}

public void execute()

{

Console.WriteLine(" CPU - execute");

}

}

class HardDrive

{

public byte[] read(long lba, int size)

{

Console.WriteLine(String.Format("HardDrive - read at location ",lba));

return new byte[100];

}

}

class Memory

{

public void load(long position, byte[] data)

{

Console.WriteLine(String.Format(" Memory - loading at position: {0} ",position));

}

}

/\* Facade \*/

class ComputerFacade

{

private CPU processor;

private Memory ram;

private HardDrive hd;

public ComputerFacade()

{

this.processor = new CPU();

this.ram = new Memory();

this.hd = new HardDrive();

}

public void start()

{

long BOOT\_ADDRESS=1235644, BOOT\_SECTOR=2659574;

int SECTOR\_SIZE=128;

processor.freeze();

ram.load(BOOT\_ADDRESS, hd.read(BOOT\_SECTOR, SECTOR\_SIZE));

processor.jump(BOOT\_ADDRESS);

processor.execute();

}

}

/\* Client \*/

class You

{

public static void main(String[] args)

{

ComputerFacade computer = new ComputerFacade();

computer.start();

}

}