Facade

Definitie

Voorziet een gecombineerde interface van een verzameling van interfaces in een subsysteem. Voorziet een higher-level interface die het subsysteem gemakkelijker te gebruiken maakt. Dit is een structurele pattern.

Gebruiksfrequentie

Dit wordt heel veel gebruikt.

Deelnemers

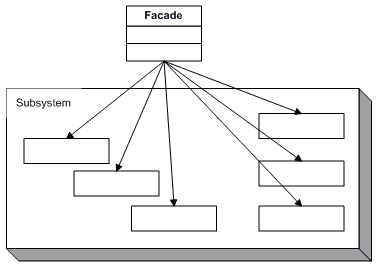
Facade (bv: MortgageApplication)

* weet welke subsysteem klasses die verantwoordelijk zijn voor een request
* stuurt de client requests naar de correcte subsysteem objecten

Subsystem Classes (bv: Bank, Credit, Loan)

* implementeert subsysteem functionaliteit
* ontvangt en verwerkt opdrachten van het facade object
* weet niets van de facade en er worden geen referenties bijgehouden

UML Design Pattern



Voorbeeld in code

/// <summary>

/// MainApp startup class for Real-World

/// Facade Design Pattern.

/// </summary>

class MainApp

{

/// <summary>

/// Entry point into console application.

/// </summary>

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"));

Console.ReadKey();

}

}

/// <summary>

/// The 'Subsystem ClassA' class

/// </summary>

class Bank

{

public bool HasSufficientSavings(Customer c, int amount)

{

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

return true;

}

}

/// <summary>

/// The 'Subsystem ClassB' class

/// </summary>

class Credit

{

public bool HasGoodCredit(Customer c)

{

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

return true;

}

}

/// <summary>

/// The 'Subsystem ClassC' class

/// </summary>

class Loan

{

public bool HasNoBadLoans(Customer c)

{

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

return true;

}

}

/// <summary>

/// Customer class

/// </summary>

class Customer

{

private string \_name;

// Constructor

public Customer(string name)

{

this.\_name = name;

}

// Gets the name

public string Name

{

get { return \_name; }

}

}

/// <summary>

/// The 'Facade' class

/// </summary>

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;

}

}

}

Verwachte output:

