User Story:

As a Scrum team member, I am going to remove phantom or ghost classes using remove middleman refactoring.

Motivation:

Sometimes in code, a class has too many methods that simply delegate to other objects. These are the set of “Phantom” or “Ghost” classes. Fowler calls these “Middlemen”. Middlemen classes simply take calls and forward them on to other components without doing any work.

Before:

public class Company

{

public void Checkout (IEnumerable<Product> products, Emlpoyee employee)

{

if (employee.IsFlagged)

{

return;

}

if (employee.IsPresent)

{

return;

}

}

}

public class Emlpoyee

{

public decimal Balance {get; private set;}

public bool Present {get; private set;}

public bool IsFlagged

{

get {return Balance < 30m;}

}

public bool IsPresent

{

get {return Present = true;}

}

}

public class Product

{

}

Why refactor:

The server class does not do anything itself and simply creates needless complexity. Every time a new feature is added to the delegate, you need to create a delegating method for it in the server class.

Solution:

This is an unneeded layer and can be removed completely with minimal effort. So, delete these methods and force the client to call the end methods directly. So, create a getter for accessing the delegate class object from the server class object. Replace calls to delegating methods in the server class with direct calls for methods in the delegate class.

**After:**

public class PizzaShop

{

public HomeDeliveryProvider HomeDeliveryProvider {get; set;}

public PizzaShop(HomeDeliveryProvider objDeliveryProvider)

{

HomeDeliveryProvider = objDeliveryProvider;

}

public void Get (int id)

{

HomeDelivery account = HomeDeliveryProvider.GetPizza(id);

}

}

public class HomeDeliveryProvider

{

public HomeDelivery GetPizza (int id)

{

// get Pizza

}

}

public class HomeDelivery

{

}