

Паттерн применяется при выводе рекомендаций для пользователя: общих рекомендаций (продукция с наибольшим количеством покупок) и персональных рекомендаций (на основе покупок пользователя (какой тип товара пользователь покупал чаще всего) выводится наиболее продаваемая продукция определённого типа).

Паттерн помогает расширить функционал без существенного изменения кода, достаточно просто использовать RecommendationManager для управления плагинами. Также в процессе работы программы эти плагины можно отключать или добавлять новые.

Без использования паттерна весь функционал плагинов пришлось бы реализовывать непосредственно в коде, и, если будет необходимо это функционал расширить, то придётся изрядно перелопачивать код.

Пример кода без использования паттерна:

// Всю реализацию плагинов перенесём в отдельные функции

List<string> GetRecommendations(List<Product> product)

{

Product best1 = product[0], best2 = product[1], best3 = product[2];

int max1 = 0, max2 = 0, max3 = 0;

for (int i = 0; i < product.Count; i++)

{

if (product[i].sale >= max1)

{

max3 = max2; max2 = max1; max1 = product[i].sale;

best3 = best2; best2 = best1; best1 = product[i];

}

else

{

if (product[i].sale >= max2)

{

max3 = max2; max2 = product[i].sale;

best3 = best2; best2 = product[i];

}

else

{

if (product[i].sale >= max3)

{

max3 = product[i].sale;

best3 = product[i];

}

}

}

}

return new List<string> { best1.type + ": " + best1.name, best2.type + ": " + best2.name, best3.type + ": " + best3.name };

}

List<string> GetUserRecommendations(List<string> userHistory, List<Product> product)

{

// ...

// Как видно приходится делать 2 разных метода, так как есть два вида рекомендаций

}

// Теперь посморим на их вызов в коде, начнём с общих рекомендаций

public void LoadForm()

{

for (int i = 0; i < 10; i++)

{

Monitor mon = new Monitor(monitor[i]);

products.Add(mon);

Keyboard key = new Keyboard(keyboard[i]);

products.Add(key);

Mouse mou = new Mouse(mouse[i]);

products.Add(mou);

}

List<string> recommendations = GetRecommendations(products);

recomendLabel1.Text = "1. " + recommendations[0];

recomendLabel2.Text = "2. " + recommendations[1];

recomendLabel3.Text = "3. " + recommendations[2];

// Т.к весь функционал в функциях, то их "отключить" нельзя, и поскольку общие рекомендации трогать больше нет необходимости

// весь код с реализацией теперь будет просто мешать, захламляя пространство

}

// Вывод пользовательских рекомендаций

private void buyMonitorButton\_Click(object sender, EventArgs e)

{

userHistory.Add(monitorComboBox.Text);

List<string> recommendations = GetUserRecommendations(userHistory, products);

userRecomendLabel1.Text = "1. " + recommendations[0];

userRecomendLabel2.Text = "2. " + recommendations[1];

userRecomendLabel3.Text = "3. " + recommendations[2];

}

Вроде бы, визуально мало что поменялось, однако если нужно будет что-то менять в реализации вывода рекомендаций или, вообще, добавить новый функционал, никак не связанный с рекомендациями, придётся изменять основной код, что сильно усложнит жизнь.

Так же существует возможность создавать плагины и динамически их загружать, используя AssemblyLoadContext. Для этого нужно подключить System.Runtime.Loader, однако у меня так и не получилось это сделать.

Пример создания плагина через AssemblyLoadContext:

https://makolyte.com/csharp-generic-plugin-loader/