1) Некорректное именование переменных, методов и классов

class Result

class API

id вместо более информативного order\_id

2) Async/await возвращает void, следует возвращать Task, так как при использовании async void результат работы метода непредсказуем.

public async Task Refund(int order\_id)

{

using (var client = new HttpClient())

{

ApiResult answer = new ApiResult();

var http = new HttpClient();

var url = String.Format(GetRefundUrl + order\_id);

var response = await http.GetAsync(url);

var result = await response.Content.ReadAsStringAsync();

answer = JsonConvert.DeserializeObject<ApiResult>(result);

Console.WriteLine(url + " " + answer.Code + " " + answer.Definition + " " + answer.Data + " " + order\_id);

}

}

Вместо

public async void Refund(int order\_id)

{

using (var client = new HttpClient())

{

ApiResult answer = new ApiResult();

var http = new HttpClient();

var url = String.Format(GetRefundUrl + order\_id);

var response = await http.GetAsync(url);

var result = await response.Content.ReadAsStringAsync();

answer = JsonConvert.DeserializeObject<ApiResult>(result);

Console.WriteLine(url + " " + answer.Code + " " + answer.Definition + " " + answer.Data + " " + order\_id);

}

}

3) Лишние try/catch, например, в следующем методе отлавливается ошибка сохранения изменений в базе данных.

public void Transaction(Order o, Card c)

{

using (var transaction = db.Database.BeginTransaction())

{

try

{

UpdateOrder(o);

UpdateCard(c);

Save();

transaction.Commit();

}

catch (Exception ex)

{

transaction.Rollback();

throw new BankErrorException("Bank Error: " + ex.Message);

}

}

}

Соответственно, обработка этого исключения в других методах, например, PayBank, уже не требуется и ведет лишь к нагромождению лишнего кода и, как следствие, к ухудшению его читабельности.

public void PayBank(Card card, Order order)

{

card.limit -= (double)order.amount\_kop / 100;

order.card\_number = card.card\_number;

order.status = "Success";

try

{

repo.Transaction(order, card);

}

catch(BankErrorException ex)

{

throw new BankErrorException("Bank Error");

}

}

Для устранения данной ошибки следует убрать блоки try/catch, которые заключают в себе уже обработанные методы.

4) Излишняя декомпозиция. Код разбит на очень мелкие методы, которые нигде не используются и не несут смысловой нагрузки. Пересмотрел архитектуру, некоторые методы объединил , некоторые реорганизовал. В результате класс получился более логичным.

Например, вместо

public void CheckCardLimit(int amount, string number)

{

double limit = GetCardLimit(number);

if (limit < 0 || limit - (double)amount / 100 >= 0)

return;

else throw new Exception("Not Enough Money");

}

public double GetCardLimit(string number)

{

var card = repo.GetCard(number);

if (card != null)

return card.limit;

throw new Exception("Card Not Found");

}

Удобнее и правильнее следующий способ:

public void CheckCardLimit(int amount, Card card)

{

double limit = card.limit;

if (!(limit < 0 || limit - (double)amount / 100 >= 0))

throw new Exception("Not Enough Money");

}

5) Static-классы в приложении лучше не применять, так как они усложняют использование механизма наследования. Лучше создавать экземпляры классов в случае необходимости.

6) Неиспользование констант там, где они необходимы (например, url-адреса). Использование констант повышает удобство изменения кода, так как в случае необходимости достаточно будет обновить только значение переменной и не искать во всём проекте места, где это значение используется.

public async Task Refund(int order\_id)

{

using (var client = new HttpClient())

{

ApiResult answer = new ApiResult();

var http = new HttpClient();

var url = String.Format(GetRefundUrl + order\_id);

var response = await http.GetAsync(url);

var result = await response.Content.ReadAsStringAsync();

answer = JsonConvert.DeserializeObject<ApiResult>(result);

Console.WriteLine(url + " " + answer.Code + " " + answer.Definition + " " + answer.Data + " " + order\_id);

}

}

Вместо

public async Task Refund(int order\_id)

{

using (var client = new HttpClient())

{

ApiResult answer = new ApiResult();

var http = new HttpClient();

var url = String.Format("http://localhost:58453/api/order/GetRefund/" + order\_id);

var response = await http.GetAsync(url);

var result = await response.Content.ReadAsStringAsync();

answer = JsonConvert.DeserializeObject<ApiResult>(result);

Console.WriteLine(url + " " + answer.Code + " " + answer.Definition + " " + answer.Data + " " + order\_id);

}

}