Декомпозиция

Фоменко М.Ю.

Про задачи

Есть часы

- Мы, как разработчики, должны разобрать их
- Должны собрать такие же своими руками

Вам нужно приобрести навык «не ломаться, когда видите огромную задачу». Ее нужно научиться разделить на части и выполнить каждую маленькую часть.

Разбор List<T> (снова)

```
public class List<T> : IList<T>, IList, IReadOnlyList<T>
{
    private const int DefaultCapacity = 4;

    internal T[] _items; // Do not rename (binary serialization)
    internal int _size; // Do not rename (binary serialization)
    internal int _version; // Do not rename (binary serialization)
```

Хранилище данных
Счетчик

```
public void Add(T item)
{
    _version++;
    T[] array = _items;
    int size = _size;
    if ((uint)size < (uint)array.Length)
    {
        _size = size + 1;
        array[size] = item;
    }
    else
    {
        AddWithResize(item);
    }
}
```

Логика (добавление)

Разбор List<T> (Add)

```
public void Add(T item)
{
    _version++;
    T[] array = _items;
    int size = _size;
    if ((uint)size < (uint)array.Length)
    {
        _size = size + 1;
        array[size] = item;
    }
    else
    {
        AddWithResize(item);
    }
}
```

```
private void AddWithResize(T item)
{
    Debug.Assert(_size == _items.Length);
    int size = _size;
    Grow(size + 1);
    _size = size + 1;
    _items[size] = item;
}
```

```
int newCapacity = _items.Length == 0 ? DefaultCapacity : 2 * _items.Length;
```

Проблема добавления

Раз мы работаем с массивом, значит добавление нового элемента в список – быстрая задача?

А что будет если у нас 1024 элемента и мы добавляем 1025ый?

Нам нужно создать новый массив, каким размером?

+1, +10, *2, *10?

А теперь проблема ресайза

Увеличение ёмкости массива требует копирования массива в новую ячейку памяти, что отнимает много времени. Чтобы снизить вероятность копирования массивов, список удваивает свою ёмкость.

X2 – компромиссное значение между «много» и «часто»

```
if (value != _items.Length)
{
    if (value > 0)
    {
        T[] newItems = new T[value];
        if (_size > 0)
        {
            Array.Copy(_items, newItems, _size);
        }
        _items = newItems;
    }
    else
    {
        _items = s_emptyArray;
    }
}
```

```
int newCapacity = _items.Length == 0 ? DefaultCapacity : 2 * _items.Length;
```

Все еще проблема ресайза

Копируем поэлементно все элементы из старого массива в новый. Это O(N).

Каждый наш Add() – лотерея. Мы либо мгновенно вставим новый, либо перетаскиваем старый массив в новый.

Метод Resize – еще один отдельный модуль. Его задача увеличивать массив.

Скрытие логики (инкапсуляция)

- Зачем мы прячем _data и _count, почему эти поля помечены как private(internal)?

- *«Да дайте мне прямой доступ к массиву, чё вы?»*
- *«Ага, чтобы вы залезли ручками в _data[5] и сломали всю логику счётчика _count? Или изменили размер массива в обход Resize()? Нет, дружок. private — это не для красоты. Это защита от дурака. В первую очередь, от самого себя.*

Модификаторы доступа

- private: закрытый или приватный компонент класса или структуры. Приватный компонент доступен только в рамках своего класса или структуры.
- public: публичный, общедоступный компонент класса или структуры. Такой компонент доступен из любого места в коде, а также из других программ и сборок.
- protected: такой компонент класса доступен из любого места в своем классе или в производных классах. При этом производные классы могут располагаться в других сборках.

Пример нарезки задачи

- Задача — калькулятор. Почему это не так просто, как кажется?

- Декомпозиция — разделение большого и сложного на небольшие простые части.

Пример нарезки задачи

Шаг 1: Шок (это нормально). Посмотреть на неё целиком: «Нужен калькулятор для (3+4)*2».

Шаг 2: Нарезать на самые крупные куски (Black Box подход).

Шаг 3: Определить интерфейсы (что модули будут давать друг другу).

Шаг 4: Повторяем, пока задачи не станут очевидно понятными.

Пример нарезки задачи

Шаг 1: Нужен калькулятор для (3+4)*2

Шаг 2: Что прога должна делать глобально? Она должна взять строку и выдать число. Значит, у нас есть как минимум два этапа: ПОНИМАНИЕ и ВЫЧИСЛЕНИЕ.

[СТРОКА] -> [МОДУЛЬ ПОНИМАНИЯ] -> [МАГИЯ] -> [МОДУЛЬ ВЫЧИСЛЕНИЯ] -> [ЧИСЛО]

Шаг 3: Что за «магия»? Пусть это будет Обратная Польская Нотация (разберем 1 декабря)

Модуль_1(string infix) -> возвращает string rpn

Модуль_2(string rpn) -> возвращает double result

Пример нарезки задачи

Шаг 4: Модуль_1. «Преобразовать в ОПН». Всё ещё сложновато. Надо строку (3+4)*2 разбить на токены (, 3, +, 4,), *, 2.

Вот еще подзадача — Токенизатор (или Лексер).

Пример нарезки задачи

Итог нарезки калькулятора:

- Написать функцию, которая режет строку на токены.
- Написать функцию, которая берёт токены и преобразует их в ОПН.
- Написать функцию, которая берёт ОПН и считает результат.
- Написать Main, который дёргает эти три функции по очереди.

Вывод: Мы превратили одну непонятную, страшную задачу в четыре маленькие и ясные. Каждую из них уже можно сесть и написать. Проектирование.

Лаба 6

Мы напишем три маленькие утилиты и склеим их вместе.

Разбор по модулям:

- Токенизатор:string.Split или цикл по символам. Изолированная задача.
- Конвертер (Сортировочная станция): самая сложная логика, но она решает только одну задачу — преобразование. Конвертору не важно как потом это будет считаться.
- Вычислитель: тупой, как пробка, алгоритм со стеком. Ему не важно откуда взялась эта ОПН-строка, он просто её считает.

Лаба 6

Если у вас баг в вычислениях, вы лезете только в Вычислитель. Вам не нужно трогать Конвертер.

Если парсер ошибочно обрабатывает скобки, вы чините только Конвертер. Модули независимы.

Код перестал быть монолитом и стал похож на конструктор LEGO.

Финал

Вскрыли чужой «чёрный ящик» (List) и увидели, что он состоит из простых деталей.

Мы взяли свою большую задачу (Калькулятор) и сами нарезали её на такие же простые «чёрные ящики», определив, как они будут общаться между собой.

Прежде чем написать хоть одну строчку кода, возьмите листок бумаги и ручку. Режьте вашу задачу на куски, пока вам не станет абсолютно понятно, что делает каждый из них. И только потом открывайте IDE. Иначе вы напишете sheetcode.

Спасибо за внимание

- ОПН на следующей лекции (а мб и сейчас)