C# 3.0 Features (Linq – killer-feature) – 2007 год

* +9 новых особенностей и большинство из них 6/7 предназначены для реализации нового языка Linq.

Язык - встроен в язык C#

Linq = language integrated query – интегрированный (в C#) язык запросов (к любым структурированным перечислимым ДАННЫМ)

Благодаря Linq C# приобрел черты Декларативных языков.

1. Implicitly typed local variables – неявно типизированные локальные переменные

var

1. Var

Тип переменной задается не слева от имени (как привыкли), а тип определяется автоматически из факта присвоения исходя из их типа данных результата выражения СПРАВА.

ЭТОТ тип определяется до КОМПИЛЯЦИИ

1. Automatic properties – автоматические свойства

- для удобства работы с Linq.

-Разрешается не прописывать полностью геттер и сеттер для Проперти + можно не прописывать Приватное поле для проперти

- авто создаст КОМПИЛЯТОР.

1. Object initializer – инициализатор объекта – позволяет задавать значение пропертей сразу при создании объекта без конструктора.
2. Collection initializer (инициализация коллекции) – позволяет при создании коллекции – сразу заполнять значениями. Работает при с любой коллекцией с интерфейсом IList -> .Add()
3. Anonymous Types – анонимные типы. Создана для Linq. Позволяет создавать классы на основе их использования в режиме Runtime. (Под классом мы не пишем – генерируется авто во время работы программы) => использование Проперти (методов не получится). Одноразовый класс (Повторное использование не предусмотрено).
4. Anonymous delegates and Lambda Operator And Lambda Expressions – анонимные типы, лямбда оператор и лямбда выражения. 2007

Выражение с лямбда оператором => называется Лямбда выражением – является анонимным делегатом. (По сути – сокращенная запись анонимных делегатов) Синтаксический сахар.

В C#3.0 для удобства добавили 2 универсальных делегата с большим кол-вом перегрузок. (Чтобы не писать каждый раз новые делегаты).

1. Action<T1,T2,T3,T4, …> T2,T3,T4, перегрузки

Метод void с нужным числом параметров нужного типа.

1. Func<T1,T2,T3, …> T3 – возвращаемый тип, а остальные – параметры
2. Extension methods – Методы расширения. (Прекрасно вписывается в принципы S.O.L.I.D.)

Причина появления

Разработчики должны были добавить ко всем существующим в .NET коллекциям множество новых методов.

Было два классических способа решения

1. Просто переписать сущ. классы – добавить новые методы. (плохо – нарушает SOLID, это может сломать и работу системы, и работу сторонних программ, которые пользовались коллекциями)
2. Вообще создать новые коллекции – не очень красивые, по факту в 2 раза увеличилось бы кол-во ИМЕН для, по сути, одинаковых коллекций.
3. Близкий ко 2 -> наследование + добавить новые методы.
4. Был придуман 4 способ в C# - новая фича для этого способа. К любому классу можно добавить новый метод без перекомпиляции (абс безопасно с соблюдением SOLID).

Алгоритм создания метода расширения

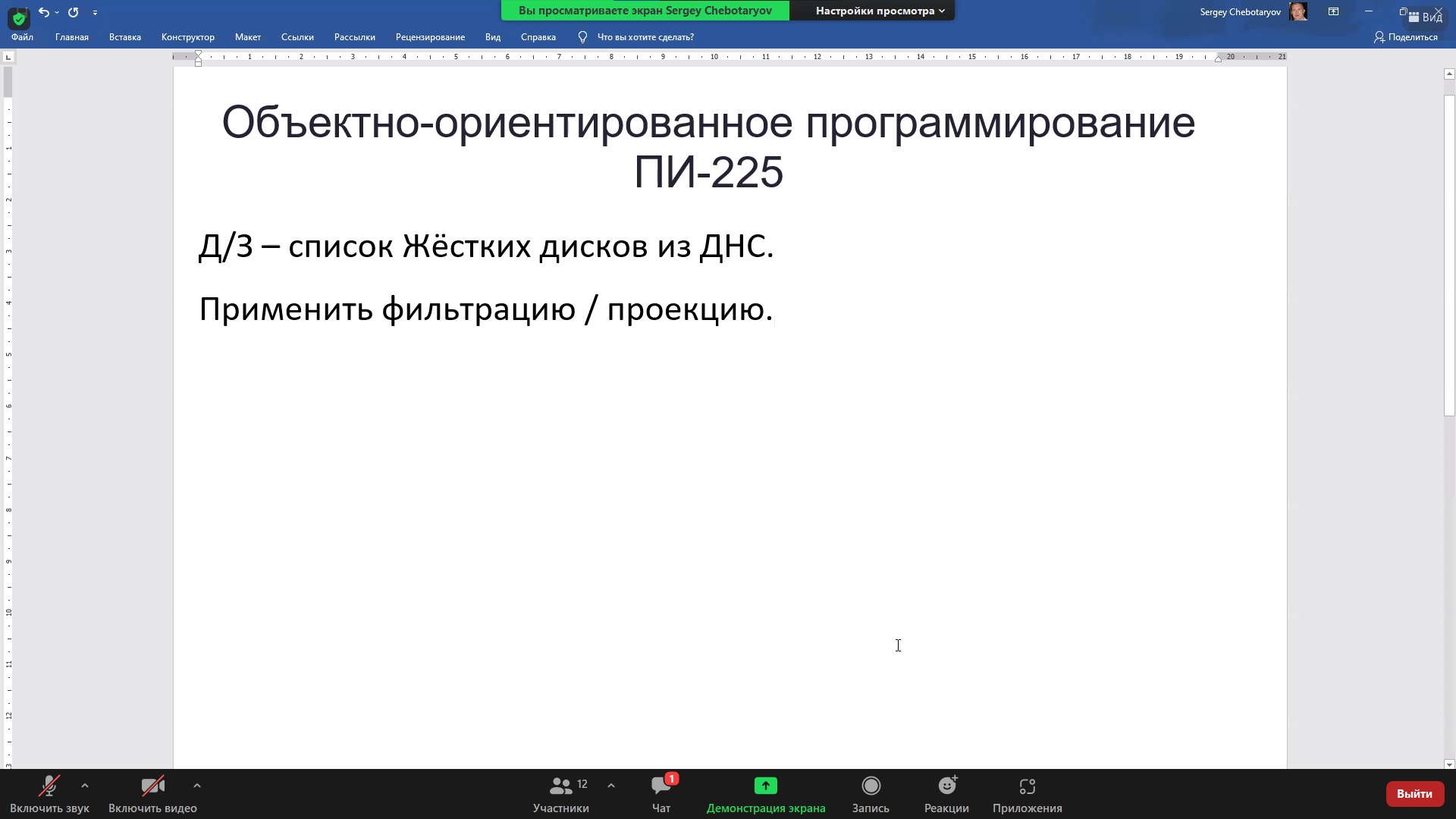
1. Необходимо добавить класс – static
2. В этот класс добавить публичный статический метод – выполняющий необходимую работу
3. В этом методе должен быть параметр – объект того класса, к которому нужно добавить этот метод.
4. К этому параметру дописывается “новое” ключевое слово this.

В C#3.0 Добавили фичу Методы Расширения и еще более 2 десятков методов во все коллекции.

1. Язык Linq (Все остальные фичы для нее добавлены)

Нужно вызубрить ВСЕ методы расширения

|  |  |
| --- | --- |
| Фильтрация | .Where() |
| Проекция | .Select(), .SelectMany() |
| Квантификация | .Any(), .All() |
| Агрегация | .Sum(), Average(), .Max() |
| Поиска | .First(), .Last, .FirstorDefault(), .LastOrDefault() |
| Группировка | .GroupBy(), join() |
| Разбиение  Сортировка | .Skip(), Take(), .SkipWhile, .TakeWhile()  .OrderBy(), .ThenBy() |
| Приведение типа | .OfType<T>(), Cast<T>() |
| Преобразования | .ToArray(), .ToList(), .ToDictionary(), .AsQuaryable(), .Asenumerable() |
| Операции со множествами | .Union(), .intersect(), .Except() |



Дз   
разбить студентов по курсам без Linq

В 2007 вышел .net 3.5

В котором появилось 2 новых фреймворка

1. WPF => Windows Presentation Foundation (новая система разработки интерфейса пользователя)
2. WCF => Windows Communication Foundation (новая система сетевого взаимодействия)

Также вышел еще один фреймворк

1. TPL = Tasks parallel library (До этого все пользовались Thread – System.Threading; 2001)

WPF – framework for UI (2007 = .NET3.5, C#3.0 + Linq (+WCF + WPF + TPL) + EntityFramework(Db) OOP + Patterns + SOLID)

(.NET, MonoGame, Windows Forms)

ПРИЧИНЫ:

1. Windows Forms(2001 c 1995) к 2007 году стал морально устаревать: не поддерживает аппаратное ускорение, стиль визуально устаревший (юзеры красиво), тач-скрины не поддерживает.
2. Microsoft – унифицировать технологий создать UI (wpf – web-app, windows app, mobile app)
3. Унифицировать проектирование и разр UI м\у дизайнерами и программистами.

Wpf 1,2,3,4 … Windows 8 => ~~WPF, XNA,~~ - JavaScript

Wpf 5 (MAUI .net core)

WPF FEATURES

1. Аппаратное ускорение графики (DirectX) – графика быстрая: спецэффекты, шейдеры, 3д.
2. Декларативный стиль описания UI – вёрстка (как HTML) – специальный язык XAML = eXtensible Application Markup language (xml)
3. Векторная графика – Все элементы интерфейса – масштабирование + математические преобразования без потери качества
4. Собственная встроенная система учета плотности пикселей.
5. Никакого абсолютного позиционирования (никаких координат!) – только относительное! – АДАПТИВНАЯ верстка
6. DataBinding – привязка данных
7. Система векторной анимации по ключевым кадрам
8. Система стилей (почти как css)
9. Система триггеров
10. Система шаблонов контролов
11. Поддержка SOLID и множество гибких интерфейсов.