OTUS

# Разработчик C#

Best Practice по разработке на C# и .NET Framework с практикой Scrum-методики

Длительность курса: 156 академических часов

## 1 C# Basic: Необычные особенности привычных вещей

1 **Знакомство, рассказ о формате Scrum, краткий обзор курса**

познакомиться.

Домашние задания

1 Знакомимся с командой и гит репозиторием

Цель: В этом ДЗ мы прорабатываем организационные моменты и лучше знакомимся друг с другом.

Для всех:
1) Зарегистрироваться на Гитлабе и сделать коммит в общий проект, при сдаче ДЗ необходимо приложить ссылку на коммит в gitlab репозитории;
Если присоединились к курсу после первого занятия, то нужно написать в slack-канал потока о том, что начали заниматься на курсе после первого занятия и хотели бы присоединиться к scrum-команде, можно продублировать сообщение руководителю программы Алексею Ягур (Aleksey Yagur) в slack, он с радостью поможет найти подходящую команду.
2) Написать “о себе” в закрытой группе команды, при сдаче ДЗ также приложить ссылку на сообщение в Slack;
3) Написать три способа траты баллов.

Для SCRUM-мастеров:
4) Выбрать название для команды;
5) Создать закрытый чат команды в слаке (добавить в него @Alex Yagur);
6) Создать проект на Гитлабе;
7) Выбрать проект для команды;
8) Создать доски задач: беклог, в работе, сделано;
9) Наполнить верхнеуровневый беклог.

| | | |
|---|---|---|
| 2 | **Операторы и методы, их перегрузка и расширения** | объяснить назначение всех операторов, какие из них могут быть переопределены;<br>создавать индексаторы;<br>писать изящный код с использованием методов расширения и параметрического полиморфизма. |
| 3 | **Классы как воплощение принципов ООП** | использовать классы более профессионально;<br>реализовывать основные принципы ООП на C#;<br>разбираться в тонкостях наследования и полиморфизма. |
| 4 | **Интерфейсы и их особенности** | работать с интерфейсами;<br>выстраивать грамотную архитектуру в ООП стиле.<br><br>Домашние задания<br><br>1 Создаём набор классов и их интерфейсов<br><br>Задание 1:<br>Реализация IEnumerable<T> на примере чтения списка элементов из xml, json ил csv файла.<br>В качестве десериализатора можно использовать https://github.com/ExtendedXmlSerializer/home или http://csvhelper.com/<br><br>Здесь более пошагово на примере xml и некоего класса Person:<br>https://pastebin.com/6FJZanYv<br><br>Задание 2:<br>Создать интерфейс IAlgorithm, добавить в него метод. Например, сортировка. Применить интерфейс к классу из первого задания.<br>(Помните, что реализовав IEnumerable<T>, для вашего класса становятся доступны методы LINQ?)<br><br>Задание 3:<br>Реализуйте интерфейсы из https://gist.github.com/viktor-nikolaev/46e588fc1c52bbf03186597af23fcd61<br>Напишите тесты IAccountService.AddAcount() с использованием Moq. |
| 5 | **50 оттенков LINQ** | работать с LINQ.<br><br>Домашние задания<br><br>1 Обрабатываем данные цепью методов<br><br>Цель: Разработать приложение, имитирующее работу банкомата, задание поможет отработать использование LINQ запросов на реалистичной задаче.<br>Изначально имеется несколько коллекций с заранее заполненными данными, имитирующими таблицы в базе данных и класс, который с ними работает. |

Коллекция Пользователи - хранит информацию о пользователях, зарегистрированных в банке
List<User> = ..
Id, Имя, фамилия, отчество, телефон, данные паспорта, дата регистрации, логин, пароль
Коллекция Счета - хранит информацию о счетах пользователей, у каждого пользователя может быть несколько счетов.
List<Account>=...
Id, дата открытия счёта, сумма на счёте, Id владельца
Каждый счёт связан с пользователем по Id
Коллекция История операций - хранит историю о всех операциях с конкретным счётом, имеется 2 типа операции, пополнение и вывод денег со счёта.
List<OperationHistory> = ...
Id, дата операции, тип операции, сумма, Id счёта
Каждая запись в истории связана с конкретным счётом
Класс ATMManager - отвечает за бизнес-логику работы банкомата.

Заполнение коллекций тестовыми данными и инициализация ATMManager уже реализована.
Необходимо реализовать LINQ запросы:

1. Вывод информации о заданном аккаунте по логину и паролю;
2. Вывод данных о всех счетах заданного пользователя;
3. Вывод данных о всех счетах заданного пользователя, включая историю по каждому счёту;
4. Вывод данных о всех операциях пополнения счёта с указанием владельца каждого счёта;
5. Вывод данных о всех пользователях у которых на счёте сумма больше N (N задаётся из вне и может быть любой);

---

6 **Особенности встроенных коллекций**

разобраться в базовых интерфейсах для коллекций;
понять какие существуют базовые стандартные коллекции;
объяснить внутреннее строение массива;
перечислить группы встроенных коллекций и в каких случаях их следует применять.

---

7 **Строки и регулярные выражения**

использовать классы String и StringBuilder;
объяснить назначение кодировок;
использовать строковые и символьные функции;
объяснить, что такое иммутабельность и интернирование;
создавать регулярные выражения.

Домашние задания

1 Список URL-адресов

Цель: Составление регулярного выражения для парсинга HTML-страницы.

Составить регулярное выражение для поиска URL адресов в тексте (создавайте сами!)
Написать программу, которая:
1. На вход принимает URL-адрес,

2. Загружает HTML-текст с этого адреса.
3. Находит URL-адреса, используя регулярное выражение.
4. Выводит на экран список всех найденных URL-адресов.

В отчёте написать:
1. Составленное регулярное выражение.
2. Ссылку на репозиторий с вашим проектом.
3. Сколько времени ушло на выполнение задания.

---

| | | |
|---|---|---|
| 8 | **Ретроспектива и планирование** | уверенно участвовать в ретроспективах;<br>планировать объём работ на будущее. |

1 **Отражение (Reflection)**

обрабатывать экземпляры разных или неизвестных заранее классов;
создавать универсальные тесты;
писать свои компоненты.

Домашние задания

1 Рефлексия и её применение

Цель: Цель написать свой класс-сериализатор данных любого типа в формат CSV, сравнение его быстродействия с типовыми механизмами серализации.
Полезно для изучения возможностей Reflection, а может и для применения данного класса в будущем.

Основное задание:
1. Написать сериализацию свойств или полей класса в строку
2. Проверить на классе: class F { int i1, i2, i3, i4, i5; Get() => new F(){ i1 = 1, i2 = 2, i3 = 3, i4 = 4, i5 = 5 }; }
3. Замерить время до и после вызова функции (для большей точности можно сериализацию сделать в цикле 100-100000 раз)
4. Вывести в консоль полученную строку и разницу времен
5. Отправить в чат полученное время с указанием среды разработки и количества итераций
6. Замерить время еще раз и вывести в консоль сколько потребовалось времени на вывод текста в консоль
7. Провести сериализацию с помощью каких-нибудь стандартных механизмов (например в JSON)
8. И тоже посчитать время и прислать результат сравнения
9. Написать десериализацию/загрузку данных из строки (ini/csv-файла) в экземпляр любого класса
10. Замерить время на десериализацию
11. Общий результат прислать в чат с преподавателем в системе в таком виде:

Сериализуемый класс: class F { int i1, i2, i3, i4, i5;}
код сериализации-десериализации: ...
количество замеров: 1000 итераций
мой рефлекшен:
Время на сериализацию = 100 мс
Время на десериализацию = 100 мс
стандартный механизм (NewtonsoftJson):
Время на сериализацию = 100 мс
Время на десериализацию = 100 мс

---

2 **Атрибуты**

использовать существующие атрибуты;
проверять наличие атрибутов на классах, функциях, полях, и т.д.;
создавать свои собственные атрибуты.

---

3 **Как устроена Сериализация?**

использовать механизмы сериализации и результирующие форматы сериализации;

| | | |
|---|---|---|
| | | применять стандартные способы сериализации. |
| 4 | **Исключения и нюансы работы с ними** | объяснить, что из себя представляет исключение;<br>разобраться как исключение устроено "под капотом";<br>научиться бросать и перехватывать исключения;<br>познакомиться с Best Practice в работе с исключениями. |
| 5 | **Базы данных: организация работы с потоками данных** | объяснить основы работы с базами данных.<br><br>Домашние задания<br><br>1 Подключаем базы данных к проекту<br><br>Цель: В этом домашнем задании вы научитесь создавать базу данных с таблицами, а также писать скрипты наполнения таблиц данными. Но самое главное - вы создадите приложение, способное получать данные из базы и добавлять новые.<br><br>1. Создать базу данных PostgreSQL для одной из компаний на выбор: Авито, СберБанк, Otus или eBay. Написать скрипт создания 3 таблиц, которые должны иметь первичные ключи и быть соединены внешними ключами.<br>2. Написать скрипт заполнения таблиц данными, минимум по пять строк в каждую.<br>3. Создать консольную программу, которая выводит содержимое всех таблиц.<br>4. Добавить в программу возможность добавления в таблицу на выбор. |
| 6 | **Базы данных: реляционные базы и работа с ними** | научиться работать с реляционными базами данных. |
| 7 | **Базы данных: NoSQL базы и их особенности** | научиться работать с NoSQL базами данных |
| 8 | **Работа с методами как с переменными (delegates, events)** | создавать делегаты для передачи методов в функции;<br>создавать события и подписки на них;<br>писать менее связный код и упростить расширение функционала ПО.<br><br>Домашние задания<br><br>1 Делегаты и события<br><br>Цель: В этом задании требуется реализовать механизмы делегатов и событий для получения практического навыка их применения<br><br>1. Написать обобщённую функцию расширения, находящую и возвращающую максимальный элемент коллекции. |

| | | |
|---|---|---|
| | | Функция должна принимать на вход делегат, преобразующий входной тип в число для возможности поиска максимального значения.<br>public static T GetMax<T>(this IEnumerable<T> e, Func<T, float> getParametr) where T : class;<br>2. Написать класс, обходящий каталог файлов и выдающий событие при нахождении каждого файла<br>3. Оформить событие и его аргументы с использованием .NET соглашений:<br>public event EventHandler<FileArgs> FileFound;<br>FileArgs – будет содержать имя файла и наследоваться от EventArgs<br>4. Добавить возможность отмены дальнейшего поиска из обработчика<br>5. Вывести в консоль сообщения, возникающие при срабатывании событий и результат поиска максимального элемента |
| 9 | **Дженерики, их реализация и ограничения** | работать с обобщениями. |
| 10 | **Сборщик мусора, деструкторы и финализаторы, Disposable Pattern** | разобраться с тем, как хранятся объекты в памяти в .NET;<br>познакомиться с алгоритмом выделения физической памяти для приложений;<br>понять алгоритм работы Сборщика Мусора (поколения, стратегии, карточный стол);<br>начать отличать Деструкторы от Финализаторов;<br>научиться использовать Disposable Pattern. |
| 11 | **Дополнительные возможности языка: от директив препроцессора до указателей** | писать небезопасный код и создавать указатели, если, столкнётесь с тем, что это понадобится;<br>использовать механизмы условной компиляции, чтобы обеспечить зависимость поведения проекта от окружения или обеспечить большее удобство работы с кодом;<br>применять динамические объекты и заготовки кода, чтобы ускорить написание проектов. |
| 12 | **Что полезного в новых версиях C#?** | проанализировать нововведения в стандартах 4.7 - 4.8 |
| 13 | **Ретроспектива и планирование** | уверенно участвовать в ретроспективах;<br>планировать объём работ на будущее. |

1 **Введение в параллелизм в .NET. Отличия процесса, потока, домена и таска**

дать описание разным примитивам параллелизма для лучшего понимания их назначения и отличий;
получить примеры практического использования инструментов параллелизма на практической задаче.

Домашние задания

1 Параллельная загрузка данных из файла

Цель: Сделать параллельный обработчик файла с данными клиентов на основе подготовленного проекта с архитектурой.
Задание поможет отработать основные инструменты параллелизма на реалистичной задаче.

Программа-обработчик должна в параллельном режиме обработать файл с данными клиентов.

Каждая строка файла содержит:

id (целое число)
ФИО (строка),
Email (строка)
Телефон (строка).

Данные отсортированы по id. Нужно десериализовать данные клиента в объект и передать объект в метод класса, который сохраняет его в БД, вместо сохранения в БД можно сделать просто задержку.

Задания

1. Запуск генератора файла через создание процесса, сделать возможность выбора в коде, как запускать генератор, процессом или через вызов метода. Если вдруг встретится баг с генерацией, то его нужно исправить и написать об этом при сдаче работы.
2. Распараллеливаем обработку файла по набору диапазонов Id, то есть нужно, чтобы файл разбивался на диапазоны по Id и обрабатывался параллельно через Thread, сколько диапазонов столько потоков. Хорошо сделать настройку с количеством потоков, чтобы можно было настроить оптимальное количество потоков под размер файла с данными. Предусмотреть обработку ошибок в обработчике и перезапуск по ошибке с указанием числа попыток. Проверить обработку на файле, в котором 1 млн. записей, при сдаче задания написать время, за которое был обработан файл и количество потоков.
3. Вместо создания потоков через new Thread() использовать ThreadPool, при сдаче задания написать время, за которое был обработан файл и количество потоков.
4. Добавить сохранение в реальную БД, можно SQL Lite для простоты тестирования или для лучшего понимания специфики загрузки полноценную базу данных (MS SQL Server, PostgreSQL, Mongo и т.д.)
5. Сделать обработку файла в CSV формате, то есть

| | | |
|---|---|---|
| | | написать генератор и разбор файла.<br>6. Дать обратную связь по 2-м домашним заданиям других студентов на курсе.<br><br>Инструкция<br><br>1. Сделать форк репозитория из ссылки в материалах, можно изменить структуру проектов, классов и интерфейсов, как считаете нужным и в ReadMe.md описать за что отвечает проект, класс, интерфейс.<br>2. Реализовать 1 пункт задания, сделав в main проекта запуск процесса-генератора файла, его нужно будет собрать отдельно и передать в программу путь к .exe файлу, также сделать в Main вызов кода генератора из подключенного проекта, выбор между процессом или вызовом метода сделать настройкой (например аргумент командной строки или файл с настройками) со значением по умолчанию для метода.<br>3. Реализовать 2 пункт задания, сделав свои реализации для IDataLoader и IDataParser.<br>4. По желанию реализовать 3 пункт задания, сделав дополнительную реализацию IDataLoader.<br>5. По желанию реализовать 4 пункт задания, сделав дополнительную реализацию для ICustomerRepository и инициализацию БД при старте приложения, можно использовать EF.<br>6. По желанию реализовать 5 пункт задания, сделав дополнительную реализацию для IDataParser и IDataGenerator.<br>7. По желанию дать обратную связь по 2-м домашним заданиям других студентов на курсе, можно найти репозитории по форкам к этому репозиторию. Обратную связь можно описать, создав issue к репозиторию, пример обратной связи можно посмотреть из ссылки на проект в материалах, который рассматривается в рамках занятия. Чтобы обратная связь была качественной обязательно нужно похвалить работу, написав, что сделано хорошо и написать, что можно улучшить с пояснениями почему это сделает работу более качественной. Эти рекомендации работают и для code review, так как позволяют более конструктивно обсуждать коммиты. |
| 2 | **Асинхронные операции** | писать асинхронный код. |
| 3 | **Примитивы синхронизации потоков** | познакомиться с примитивными и гибридными средствами управления доступа к общим ресурсам;<br>познакомиться со средствами синхронизации потоков;<br>рассмотреть дополнительные инструменты для упрощения организации многопоточной работы;<br>применить полученные знания в практическом занятии. |
| 4 | **Внутрипроцессное взаимодействие** | применять потоки, задачи, Parallel LINQ;<br>распараллеливать расчёты для ускорения вычислений;<br>оценивать целесообразность применения механизмов;<br>параллельной обработки данных. |

| | | |
|---|---|---|
| | | Домашние задания<br><br>1 Добавляем способы загрузки данных из файла |
| 5 | **Межпроцессное взаимодействие** | изучить способы взаимодействия процессов/программ друг с другом. |
| 6 | **Магические слова async / await** | изучить механизм, скрытый под ключевыми словами async/await; научиться правильному использованию этих ключевых слов. |
| 7 | **Порождающие шаблоны проектирования** | объяснить назначение изученных шаблонов проектирования; применять их в своих проектах.<br><br>Домашние задания<br><br>1 Реализуем шаблоны |
| 8 | **Структурные шаблоны проектирования** | объяснить назначение изученных шаблонов проектирования и применять их в своих проектах. |
| 9 | **Поведенческие шаблоны проектирования** | объяснить назначение изученных шаблонов проектирования и применять их в своих проектах. |
| 10 | **Ретроспектива и планирование** | уверенно участвовать в ретроспективах;<br>планировать объём работ на будущее. |

# 4 Клиент-серверная архитектура и микросервисы

| | | |
|---|---|---|
| 1 | **Архитектура проекта** | формировать архитектуру приложения;<br>разделять код на слои и звенья;<br>принимать более взвешенные решения о структуре проекта. |
| 2 | **Авторизация и аутентификация** | разработать и внедрить в приложение систем авторизации и аутентификации. |
| 3 | **WCF, ASMX, Web Api, REST** | отличать технологии удалённых вызовов друг от друга;<br>создавать готовый RESTful API сервер в интернете за пару кликов мышки;<br>реализовывать клиенты к RESTful API серверам.<br><br>Домашние задания<br><br>1 Добавляем взаимодействие между клиентом и сервером |
| 4 | **Паттерны корпоративных приложений** | сформулировать основные причины интеграции корпоративных систем и способы их интеграции;<br>сформулировать какой способ взаимодействия предпочтителен для различных типов задач и в чем причина популярности микросервисов. |
| 5 | **Насколько твёрдые SOLID принципы?** | изучить значение каждой буквы акронима;<br>узнать как применять эти принципы на практике.<br><br>Домашние задания<br><br>1 Создаём два микросервиса с общением через брокер сообщений |
| 6 | **В поисках лучшего брокера сообщений** | разобраться в сходствах и отличиях четырёх наиболее популярных брокеров сообщений;<br>научиться писать код, получающий сообщения из очереди. |
| 7 | **CI/CD** | разобраться в принципах CI/CD;<br>понимать как организовать правильный CI/CD;<br>сформулировать основные инструменты для CI и CD, их отличия;<br>настроить собственный CI/CD (пример на GitLab CI). |
| 8 | **Ретроспектива и планирование** | уверенно участвовать в ретроспективах;<br>планировать объём работ на будущее. |

# 5 Процессы и подходы

| | | |
|---|---|---|
| 1 | **Waterfall, Scrum, Kanban и прочие методологии** | разобраться в современных методологиях, необходимости и особенностям их применения. |
| 2 | **Unit, Sandbox, Blackbox, Whitebox, Integration tests** | разбираться во всём этом многообразии типов тестов;<br>понимать когда и какие нужно применять. |
| 3 | **Domain Driven Development: Основы** | изучить теоретическую часть подхода Domain Driven Development. |
| 4 | **Domain Driven Development: Практикум** | научиться применять подход самостоятельно на практике. |
| 5 | **Исследование и анализ работы программ** | разбираться в инструментах исследования и анализа работы программ;<br>выбирать и настраивать подходящую систему логирования;<br>объяснять что из себя представляют метрики и как устроена трассировка;<br>перечислить минимум три системы для ведения программной документации. |

# 6 Проектная работа

| | | |
|---|---|---|
| 1 | **Консультация по проектам и домашним заданиям** | получить ответы на вопросы по проекту, ДЗ и по курсу.<br><br>Домашние задания<br><br>1 Проектная работа |
| 2 | **Защита проектных работ** | защитить проект и получить рекомендации экспертов. |