ЛАБОРАТОРНАЯ РАБОТА №1

Общая задача:

Необходимо создать программу, которая будет выполнять функции записной телефонной книжки. Программа должна обеспечивать возможность:

1) Создания новой записи, включая возможность указания следующих сведений для каждой из записей:
    a. Фамилия;
    b. Имя;
    c. Отчество (поле не является обязательным);
    d. Номер телефона (только цифры);
    e. Страна;
    f. Дата рождения (поле не является обязательным);
    g. Организация (поле не является обязательным);
    h. Должность (поле не является обязательным);
    i. Прочие заметки (поле не является обязательным).
2) Редактирования созданных записей.
3) Удаления созданных записей.
4) Просмотра созданных учетных записей.
5) Просмотра всех созданных учетных записей, с краткой информацией, включающей следующие основные поля:
    a. Фамилия;
    b. Имя;
    c. Номер телефона.

Программа должна представлять собой классическое консольное приложение, реализованное с использованием стека технологий .Net, предоставлять интерактивный интерфейс для взаимодействия с пользователем и иметь возможность сессионного хранения созданной информации.

Коротко о главном

Вот так выглядит наша первая с вами лабораторная работа. На самом деле ничего особенно сложного нет. Более того, когда-то давно примерно такой была моя первая самостоятельно написанная программа, написанная на моем первом полноценном ООП языке. Именно с неё все и началось, долгая дорога из бессонных ночей и тонны прочитанной информации в интернете для того чтобы открыть для себя мир разработки. Сформулировал эту задачу для меня мой старый друг, который к тому моменту уже не одну собаку съел на этом не лёгком поприще, и думаю ему было достаточно забавно наблюдать за тем как я мучаюсь. Данная задача очень хорошо подходит для начинающих программистов, в основном за счет того, что она не слишком сложна и имеет потенциал для того чтобы переписываться и помочь оттачивать свои навыки и применять новые знания на более высоких материях, на каких именно я расскажу чуть позже, а пока что мы поговорим с вами о такой важной вещи как декомпозиция задачи.

Поскольку это наша первая лабораторная, с декомпозицией задачи вам помогу я. Сейчас мы проведём декомпозицию вместе, попутно рассуждая о том, что может нам понадобиться для решения получившихся подзадач, порядок решения самих подзадач я естественно умолчу, предоставив уже вам свободу выбора и возможность дать волю своей фантазии, ну а те, кто были на занятии – можете сравнить с тем что у вас получилось сделать там, с той логикой что я опиши ниже. Но прежде, напомню вам 1 простую вещь, моя логика может не совпадать с вашей и это не делает ваше решение хуже или лучше, чем моё, оно просто другое! Приступим, для начала нам необходимо определить, что именно будет нашим основном классом, основным элементом, который будет содержать в себе всю функциональную логику работы программы. Как не трудно догадаться сама сущность записной книжки и будет являться основным, Main классом нашей программы. Именно в ней и будет происходить взаимодействие с пользователем и работа с записями, я бы в общем случае назвал этот класс Notebook,ну или если угодно - Address book. Такое название вполне однозначно соотносится с тем, что именно будет содержать в себе программа. Теперь давайте определим, что именно нам необходимо для того, чтобы программа реализовывала свои функциональные возможности. Тут тоже все достаточно просто, нам понадобиться: класс, который должен представлять собой одну страницу в записной книжке, с содержанием одной единственной записи, включающей все требуемые поля, а также нам понадобятся управляющие воздействия, поступающие от пользователя, интерпретируя которые наша программа бы смогла отображать ту или иную информацию, необходимую для осуществления требуемых действий. Я бы назвал этот класс Note или Record, ну а взаимодействие с пользователем вынес в рамках отдельного метода, реализованного с использованием конструкции «switch-case-default», либо в отдельном классе отвечающим за интерфейс взаимодействия с пользователем. Поскольку консольные приложения дают достаточно скудные возможности для диалога, он должен быть достаточно прост, и я бы рекомендовал оперировать одно символьными командами для работы с программой. Итак, подведем промежуточный итог, нам необходимо:

1) Создать новый проект в VS2017, назвать его как-то (я бы назвал NotebookApp), с использованием классического консольного шаблона.
2) Переименовать основной класс в соответствии с его назначением (я бы переименовал его в Notebook).
3) Создать новый класс, который будет являться основной моделью данных приложения (я бы назвал его Note).

4) Определиться где мы будем хранить код, отвечающий за интерфейс взаимодействия с пользователем (я бы не очень долго думая, создал его прямо в основном классе, а потом отрефакторил код при необходимости).

Отлично, общая структура приложения уже начинает вырисовываться. Давайте теперь подумаем над каждым из 3 его основных составляющих. Начнем с самого простого, с класс Note. Именно совокупность его экземпляров, будет содержать в себе наша записная книжка, а значит и все поля, которые должна помнить наша книжка, должны быть объявлены именно там. Обратите внимание, что в нашем случае, мы должны иметь возможность однозначно идентифицировать каждую из наших записей для дальнейшей работы с ними, поэтому необходимо подумать, как это сделать. Также вам необходимо задуматься о порядке создании корректных для дальнейшей работы записей, ведь ограничения по обязательности полей даны не просто так.

Теперь давайте перейдем ко второму элементу интерфейс взаимодействия. Тут все тоже просто, как реализовывать простейшие вопросно-ответные системы мы с вами уже разбирали, если кто забыл – кошачий тотализатор вам в помощь. Тут вам просто необходимо продумать все возможные варианты развития событий исходя из состава предоставляемых возможностей, докрутить вспомогательные контекстные элементы, который будут зависеть от ранее выбранных действий пользователя и его текущего местонахождения в программе. Чтобы было проще, поставьте себя на место пользователя и подумайте, что именно вы бы хотели увидеть в рамках общения с такой программой и какие именно подсказки вам бы для этого понадобились. Если кто-то и тут не совсем поймет, что нужно сделать – ищите подсказки всё в том же тотализаторе! Ну и плюс конечно же наша любимая защита от «дурака» или явно некорректных действий пользователя. Помните одну простую вещь, ни одно из действий пользователей, которое мы ему позволяем осуществить в программе, не должно привести к её поломке или аварийному завершению.

Ну и, пожалуй, самый сложный класс, основной класс программы, который будет содержать основные методы, отвечающие за функциональные возможности программы. Помимо основного метода, тут должны быть методы, которые отвечают за:

1) Создание новой записи (я бы назвал его CreateNewNote).
2) Редактирование ранее созданной записи (я бы назвал его EditNote).
3) Удаление ранее созданной записи (я бы назвал его DeleteNote).
4) Просмотр созданной записи (я бы знавал его ReadNote).
5) Просмотр всех созданных записей в общем списке (я бы назвал его ShowAllNotes).

Кроме этого, в данном классе должны хранится все созданные пользователем записи, поскольку мы еще не проходили с вами ни работу с файлами, ни работу с БД, мы будем использовать именно сессионное хранение информации, т.е. до тех пор, пока программа запущена и работает она в себе что-то содержит, как только мы её отключили она завершается и все записи удаляются. Не самая надежная система хранения, но для нас пока что подойдет. Думаю, что одна из коллекций сможет вам помочь вам справиться с этой непростой задачей!

В принципе вот и все, что может вам понадобиться для выполнения данной лабораторной работы. Как видите все достаточно просто, детализировать или проводить дальнейшую декомпозиции задач до конкретных действий по их реализации я не стану, думаю, что вы уже вышли из того состояния, когда необходимо было комментировать и подсказывать каждый следующий шаг. Поэтому на данной позитивной ноте, мы закончим разбор будущей программы и оставим вас с ней на пару часов!

Чуть не забыл! Вам ведь еще эту работу и сдавать придется, а по сему придется вам рассказать про то как это сделать:

1) Вам будет необходимо зарегистрироваться в онлайн репозитории под названием Bitbucket. Вот ссылка на него https://bitbucket.org/product. Её интерфейс прост и понятен, даже есть поддержка русского языка (кривоватая конечно, но спасибо и на этом). Эта штука, позволяет вам публиковать в открытом доступе файлы своих проектов, она одна из двух наиболее популярных и бесплатных систем на данный момент, вторая это GitLab, её суть примерно похожа, но там есть не очень крутые ограничения в бесплатной версии, поэтому мы будем использовать именно «ведёрко».
2) В качестве системы управления версиями своих проектов мы будем использоваться GIT. На занятии я уже немного рассказывал вам об этой штуке. Подробнее можно почитать здесь: https://ru.wikipedia.org/wiki/Git . Я не стану писать подробной инструкции по тому как он работает и с чего там нужно начинать, ибо в интернете их просто тонна, да и официальный сайт предоставляет исчерпывающие инструкции по его пользованию, особенно на ранних этапах. Вот вам ссылка на оригинальные статьи от создателей «ведёрка» https://www.atlassian.com/git/tutorials/learn-git-with-bitbucket-cloud, там же есть и маленький гайд по тому, с чего стоит начать.
3) Прежде чем начинать копаться в гайдах и пытаться понять почему что-то не работает, не забудьте установить сам Git на свой компьютер… https://git-scm.com/downloads вот тут можете его скачать, там же еще и документация валяется, но это для слабых =)
4) Для тех кто не любит читать, я нашел достаточно простой и быстрый гайд, который можно посмотреть: https://www.youtube.com/watch?v=xCQHK5KouL0
5) Теперь вам осталось лишь скинуть мне ссылку на свой проект в этом репозитории (убедитесь сначала, что он публичный) по электронной почте с пометкой в теме письма в формате: Лабораторная работа №1. Фамилия + имя.

Ну и еще пару слов. Хороший код, всегда покрыт тестами, я понимаю, что в текущих реалиях говорить о тестах пока рано, но готовится к этому всё равно нужно. Поэтому в качестве дополнительного задания, вам необходимо самостоятельно найти информацию про порядок тестирования программного обеспечения на различных этапах его создания. Посмотреть что такое модульное или Unit тестирование. Посмотреть общую идеологию использования подхода AAA (arrange,act,assert), ну и на последок погуглить про принцип и основы работы с инструментом для тестирования MsTest, он поддерживает и встроен в VS2017 по умолчанию, достаточно лишь знать как прикрутить к его к тестируемому проекту. Об этом и о много другом вы можете узнать, в том числе, и из официальных гайдов корпорации Microsoft: https://docs.microsoft.com/ru-ru/visualstudio/test/unit-test-your-code?view=vs-2017 . Вот теперь точно всё, удачи!