Прикладное программирование

Лабораторная работа №1

Консольное приложение учета

Задача: Необходимо разработать консольную программу для ведения учета записей по варианту

В программе должен быть предусмотрено:

1) добавление, удаление, изменение и просмотр списка основных записей. Основная запись должна содержать все указанные в варианте поля, но может содержать и дополнительные
2) разделение основной записи на подвиды. Подвиды вам нужно придумать самостоятельно (минимум два), они должны дополнять основную запись новыми полями. Например, основная запись: «Здание» с полями: «Адрес» и «Этажность». Подвид 1: «Жилое здание» с доп. полем «Жилая площадь». Подвид 2: «Промышленное здание» с доп. полем «Класс опасности»

Программа должна содержать меню из следующих разделов

1) Добавить запись (выбор подвида записи и ввод данных)
2) Просмотреть записи (список записей, отсортированных по заданному критерию, см. таблицу с вариантами). В списке должны быть отражены: номер/идентификатор записи, название подтипа записи, поля основной записи
3) Изменить запись (поиск записи по номеру/идентификатору и ввод измененной версии записи)
4) Удалить запись (поиск записи по номеру/идентификатору и подтверждение удаления)
5) Выход. Выход из программы

Структура программы:

Программа должна быть разделена на три логических блока:

1) Интерфейс
2) Хранение данных
3) Бизнес-логика

Логический блок «Интерфейс»:

Это класс `Program`.

- Его задача создавать классы бизнес-логики и хранения данных, а также взаимодействовать с пользователем
- Только в этом классе должно быть обращение к классу `Console`

Логический блок «Хранение данных»:

Создайте класс `MemoryDataSource`, который будет отвечать за хранение записей вашей программ в оперативной памяти. Этот класс должен реализовывать интерфейс `IDataSource`:

```
/// <summary>
/// Ваш класс основной записи
/// </summary>
abstract public class MainRecord
{
    public int id; // Должен генерироваться автоматически после записи в хранилище.
                   // 0 обозначает новую, еще не сохраненную запись
    // ... остальные поля ...
}

/// <summary>
/// Интерфейс для реализации хранилища записей
/// </summary>
public interface IDataSource
{
    /// <summary>
    /// Сохранение записи в хранилище.
    /// Если у записи id == 0, то значит выполняется добавление новой записи,
    /// для нее нужно сгенерировать id (порядковый номер). Иначе - обновление записи
    /// </summary>
    /// <param name="record">Добавляемая или обновляемая запись</param>
    /// <returns>Запись из хранилищая с id</returns>
    MainRecord Save(MainRecord record);

    /// <summary>
    /// Возвращает одну запись из хранилища по ее идентификатору
    /// </summary>
    /// <param name="id">Идентификатор записи</param>
    /// <returns>Найденная запись или null, если записи с таким id нет</returns>
    MainRecord Get(int id);

    /// <summary>
    /// Удаляет одну запись из хранилища по ее идентификатору
    /// </summary>
    /// <param name="id">Идентификатор записи</param>
    /// <returns>true, если запись успешно удалена</returns>
    bool Delete(int id);

    /// <summary>
    /// Возвращает все записи из хранилища
    /// </summary>
    /// <returns>Все записи</returns>
    List<MainRecord> GetAll();
}

/// <summary>
/// Хранение записей в оперативной памяти
/// ! Для предотвращения изменения записей извне,
/// все методы, возвращающие записи должны возвращать
/// их копии
/// </summary>
public class MemoryDataSource : IDataSource
{
    private List<MainRecord> records = new List<MainRecord>();

    public MainRecord Save(MainRecord record)
    {
        // ... реализация ...
    }

    public MainRecord Get(int id)
    {
        // ... реализация ...
```

```
        }

        public bool Delete(int id)
        {
            // ... реализация ...
        }

        public List<MainRecord> GetAll()
        {
            // ... реализация ...
        }

    }
```

Где `MainRecord` - класс вашей основной записи (переименуйте его согласно вашему варианту)

Логический блок «Бизнес-логика»:

Создайте класс основной логики, содержащий:

1) **Методы добавления и изменения основной записи.** Должны принимать на вход объект класса основной записи и возвращать объект обратно с уже присвоенным идентификатором (идентификатор присваивается методом *Save* в MemoryDataSource, см. выше). Внутри методов должна быть проверка правильности введенных данных (см. вариант). Если пользователь ввел неверные данные, методы должны бросать исключение (Exception). В этом случае нужно запросить повторный ввод данных в классе интерфейса
2) **Метод удаления.** Должен принимать на вход объект класса основной записи и возвращать true в случае успешного удаления.
3) **Метод получения списка записей.** Должен возвращать список записей, отсортированный по полям, указанным в таблице с вариантом. Сначала сортируется по первому полю, если первое поле равно – по второму.
4) **Метод получения записи по идентификатору.** Должен вызывать метод *Get* из MemoryDataSource. Если записи с таким идентификатором нет, то возвратить null

Конструктор класса бизнес логики должен принимать на вход объект класса хранилища данных (IDataSource)

В отчете нужно предоставить:

1) Постановка задачи
2) Код программы
3) UML-диаграмму классов вашей программы
4) Демонстрацию работы программы

Варианты:

Вариант	Тематика программы	Основная запись	Сортировка списка по	Проверка
1	Учет товаров на доставку	Заказ на доставку: • ФИО клиента • Дата • Название товара • Количество	Дата, ФИО клиента	1. Дата в формате: ГГ-ММ-ДД: 19-10-14 2. Количество больше 0

2	Учет номеров в гостинице	Бронирование/Заезд: • Имя клиента • Номер • Даты заезда/отъезда	Дата заезда, номер комнаты	1. Выбран один из следующих номеров: 1, 2, 5, 13, 42, 54 2. Имя клиента не пустое
3	Список покупок	Покупка: • Товар • Категория • Сумма • Дата	Дата покупки, название товара	1. Дата в формате: ГГ/ММ/ДД: 19/10/14 2. Сумма больше или равна 0
4	Учет задач	Задача: • Описание • Статус • Исполнитель • Дата завершения	Статус, исполнитель	1. Выбран один из следующих статусов: • Новая, • В работе, • Решена, • Закрыта 2. Длина описания больше 20 символов
5	Учет товаров магазина	Движение товара • Название товара • Дата • Изменение баланса на складе (+ или -)	Дата, название товара	1. Дата в формате: ГГГГ.ММ.ДД: 2019.10.14 2. Название товара не пустое
6	Учет документов	Документ • Название • Дата добавления • Дата завершения • Описание	Дата добавления, название	1. Даты в формате: ГГГГ-ММ-ДД: 2019-10-14 2. Дата завершения больше или равна дате начала
7	Учет оборудования	Оборудование • Название • Серийный номер • Дата постановки на учет • Дата последнего техобслуживания	Название, серийный номер	1. Даты в формате: ГГГГ/ММ/ДД: 2019/10/14 2. Серийный номер в формате: 99-999
8	Динамика курсов валют	Изменение курса • Валютная пара • Новая стоимость • Дата	Валютная пара, дата	1. Валютная пара в формате: XXX-YYY: USD-RUB 2. Валютная пара составлена из след. валют: USD, RUB, EUR, UAH, CNY, BYN
9	Учет книг в библиотеке	Заявка на книгу • Книга • Читатель • Дата получения • Дата возврата	Имя читателя, название книги	1. Даты в формате: ГГ/ММ/ДД: 19/10/14

		• Состояние книги		2. Состояние книги одно из следующих: • Новая • Мелкие повреждения • Средние повреждения • Рухлядь
10	Учет заявок на ремонт	Заявка на ремонт • Клиент • Дата начала • Дата окончания • Стоимость	Дата начала, имя клиента	1. Даты в формате: ГГГГ-ММ/ДД: 2019-10-14 2. Дата окончания больше или равна дате начала
11	Список ресторанов	Заведение • Название • Средний чек • Категория (1 или несколько) • Город	Город, название	1. Каждая из категорий состоит из одного или двух слов. Если категорий несколько, то разделитель – запятая 2. Средний чек больше 0
12	Учет сотрудников	Сотрудник • ФИО • Должность • Отдел • Оклад	Отдел, ФИО	1. ФИО содержит не меньше трех слов 2. Отдел один из следующих: • Разработка • Исследование • Маркетинг • Бухгалтерия • Юридический • Управление
13	Маршрут путешествия	Точка маршрута • Адрес • Перечень планируемых мероприятий • Дата прибытия • Дата отбытия	Дата прибытия, адрес	1. Каждое из планируемых мероприятий заключено в двойные кавычки. Мероприятия разделены точкой с запятой 2. Адрес не пуст
14	Ведение списка студентов	Студент • ФИО • Группа • Специальность • Год зачисления • Год выпуска	Группа, ФИО	1. Группа в формате: 9-99-9аа: 5-19-1од 2. Год выпуска больше года зачисления
15	История сообщений	Сообщение • Текст • От кого • Кому • Дата и время	Кому, дата и время	1. Дата и время в формате: ГГГГ-ММ-ДД ЧЧ:ММ:СС: 2019-10-14 14:10:22 2. Текст не длиннее 200 символов

16	Учет приходов и расходов	Движение средств • Название • Контрагент • Изменение баланса (+, -) • Дата	Контрагент, дата	1. Дата в формате: ГГ.ММ.ДД 19.10.14 2. Контрагент не пустой
17	Ведение плана продаж	Запланированная продажа • Продукт • ФИО менеджера по продажам • Сумма сделки • Дата закрытия сделки • Выполнена или нет	Имя менеджера, дата закрытия сделки	1. Дата закрытия сделки в формате: ГГГГ/ММ/ДД: 2019/10/14 2. ФИО менеджера в формате: Xxxx Y.Z.: Иванов И.И.
18	Учет достижений	Достижение • Наименование • Тип • ФИО • Дата	Тип, наименование	1. Дата в формате: ГГГГ.ММ.ДД: 2019.10.14 2. Тип один и следующих: Спорт, Учеба, Работа, Семья
19	Записная книжка	Заметка • Текст • Дата напоминания • Важность • Выполнена или нет	Дата напоминания, текст	1. Дата напоминания в формате: ГГГГ-ММ/ДД: 2019-10-14 2. Важность: число от 1 до 5
20	Книга рецептов	Рецепт • Название • Категория • Перечень ингредиентов (название, кол-во) • Инструкция	Категория, название	1. Перечень ингредиентов в формате: Ингредиент1: 400, Ингридиент2: 200, … 2. Инструкция не пустая

Контрольные вопросы:

1. Что такое класс и из чего он может состоять?
2. Как создаются объекты классов? Что такое конструктор?
3. Чем отличаются друг от друга спецификаторы доступа: private, protected, public и internal?
4. Что такое абстрактный класс?
5. Что такое интерфейс?
6. Что такое исключение? Когда его следует применять?
7. Что такое MVC? Какую роль играют компоненты: модель, представление и контроллер
8. Для чего предназначены UML-диаграммы?

Пример структуры программы:

```
// Класс бизнес-логики
public class BusinessLogic
{
    private IDataSource dataSource;

    public BusinessLogic(IDataSource source)
    {
        dataSource = source;
    }

    // Получение отсортированного списка
    public List<MainRecord> GetList()
    {
        List<MainRecord> list = dataSource.GetAll();

        // ... сортировка  ...

        return list;
    }

    // [... остальные методы ...]
}

class Program
{
    static BusinessLogic logic;

    static void PrintMenu()
    {
        // [...]
    }

    static void PrintList()
    {
        List<MainRecord> list = logic.GetList();

        // [... печать заголовка списка ...]

        foreach (MainRecord record in list)
        {
            // [... печать элемента ...]
        }
    }

    // [... прочие методы ...]

    static void Main(string[] args)
    {
        logic = new BusinessLogic(new MemoryDataSource());
        bool exit = false;

        while (!exit)
        {
            PrintMenu();
            string command = Console.ReadLine();
            switch (command)
            {
                // [...]
                case "2":
                    PrintList();
                    break;
                // [...]
            }
        }
    }
}
```