**Лабораторная работа №2**
**Проектирование классов. Включение классов. Инкапсуляция**

## Краткие теоретические сведения

### Списки. Класс List

Списки похожи на массивы, только с более широким спектром управления. В отличие от массивов в список легко добавить новый элемент в конец или вставить элемент в любое место, удалить элемент, сократив список, сортировать простым методом Sort, инвертировать последовательность элементов, добавить к списку другой список.

Списки представляют собой однородную коллекцию значений какого-то типа: чисел, строк или других объектов.

Это может быть список строк файла, список участников мероприятия, список карт в карточной игре на руках у игрока, список слов, загруженных из предложения.

```
List<Buyer> buyers;
List<int> a = new List<int>(); // Создать список из целых чисел
List<double> b = new List<double>(); // Создать список из дробных чисел
List<bool> c = new List<bool>(); // Создать список из булевских значений
"да"/"нет"
List<string> d = new List<string>(); // Создать список строк
List<Card> e = new List<Card>(); // Создать список объектов класса, например,
Card
List<string[]> f = new List<string[]>(); // Создать список массивов
List<List<int>> g = new List<List<int>>(); // Создать список числовых списков
```

Метод Add

С помощью метода Add можно добавить элементы в список.

```
        List<int> x = new List<int>();    //Создали пустой список для хранения
целых чисел
        x.Add(5);
        x.Add(27);
        x.Add(-6);
        x.Add(14);
        x.Add(70);
        x.Add(14);
        x.Add(178);
```

Другой же, более сокращённый способ будет выглядеть так:

```
        List<int> x = new List<int>() { 5, 27, -6, 14, 70, 14, 178 };
```

Чтобы получить первое число, используется индекс со значением 0 (счёт начинается с нуля)

```
int m = x[0];     //Значение равно 5
int n = x[1];     //Значение равно 27
```

А чтобы изменить значение 3-го элемента (индекс 2) на значение 100, используется такая запись:

```
x[2] = 100;
```

Метод Remove

Метод Remove позволяет удалить из списка элементы с этим значением. Например, попробуем удалить все элементы со значением 14. Будем считать, что у нас добавлены те элементы, что выше

```
List<int> x = new List<int>() { 5, 27, -6, 14, 70, 14, 178 };
x.Remove(14);
```

Тогда будут удалены сразу 2 элемента (под номерами 3 и 5). Обратите внимание, что индексы так же идут по порядку. Все элементы, что находились "правее", сдвинулись влево.

Метод RemoveAt

Данный метод позволяет удалить элемент с определённым индексом

```
List<int> x = new List<int>() { 5, 27, -6, 14, 70, 14, 178 };
x.RemoveAt(2);    //Будет удалён 3-ий элемент по счёту
```

Свойство Count

Считает количество элементов в списке

```
List<int> x = new List<int>() { 5, 27, -6, 14, 70, 14, 178 };
int n = x.Count;       //7 элементов
int m = x.Count - 1;   //Получить индекс последнего элемента
```

Метод Insert

Позволяет вставить новый элемент в определённую позицию. Скажем, мы хотим перед вторым (1-ым по индексу) элементом вставить число 1000. Вставив число, все остальные, что после него, сдвинутся на 1 элемент правее. Пишется это так:

```
List<int> x = new List<int>() { 5, 27, -6, 14, 70, 14, 178 };
x.Insert(1, 1000);
```

Метод IndexOf

Данный метод позволяет определить, есть ли в списке элемент с этим значением. Метод IndexOf находит позицию элемента в списке. Если найти не удалось, будет возвращено значение -1.

```
List<int> x = new List<int>() { 5, 27, -6, 14, 70, 14, 178 };
int a = x.IndexOf(5);     //Получим 0-ую позицию
int b = x.IndexOf(-6);    //Получим 2-ую позицию
int k = x.IndexOf(70);    //Получим 4-ую позицию
int q = x.IndexOf(166);   //Получим -1        }
```

Другие методы

```
List<int> x = new List<int>() { 5, 27, -6, 14, 70, 14, 178 };
x.Reverse();//Получить обратный порядок элементов, т.е. 178, 14, 70, 14, -
6, 27, 51
x.Sort();//Сортировать элементы по порядку с увеличением
int a = x.Min();//Найти наименьшее значение в списке. Получим -6
int b = x.Max();//Найти наибольшее значение в списке. Получим 178
int c = x.Sum();//Найти сумму элементов. Получим 302
double d = x.Average(); //Найти среднее значение чисел. Получим примерно
43,14
```

**Индивидуальные задания на лабораторную работу**

1,12,23 Карточка иностранного слова представляет собой структуру, содержащую иностранное слово и его перевод. Для моделирования электронного словаря иностранных слов реализовать класс *Dictionary*. Данный класс имеет поле-название словаря и содержит массив структур WordCard, представляющих собой карточки иностранного слова. Название словаря задается при создании нового словаря, но должна быть предоставлена возможность его изменения во время работы. Карточки добавляются в словарь и удаляются из него. Реализовать поиск определенного слова как отдельный метод. Аргументом операции индексирования должно быть иностранное слово. В словаре не должно быть карточек-дублей.

2,13,24 Реализовать класс Money, используя для представления суммы денег массив структур. Структура имеет два поля: номинал купюры и количество купюр данного достоинства. Номиналы представить как перечислимый тип *nominal*. Элемент массива структур с меньшим индексом содержит меньший номинал. Реализовать арифметические операции сложения, вычитания и умножения на число, и операции сравнения

3,14,25 Один тестовый вопрос представляет собой структуру *Task* со следующими полями: вопрос, пять вариантов ответа, номер правильного ответа, начисляемые баллы за правильный ответ. Для моделирования набора тестовых вопросов реализовать класс *TestContent*, содержащий массив тестовых вопросов. Реализовать методы добавления и удаления тестовых

вопросов, а также метод доступа к тестовому заданию по его порядковому номеру в списке. В массиве не должно быть повторяющихся вопросов.

4,15,26 Карточка персоны содержит фамилию и дату рождения. Реализовать класс *ListPerson* для работы с картотекой персоналий. Класс должен содержать массив карточек персон. Реализовать методы добавления и удаления карточек персон, а также метод доступа к карточке по фамилии. Фамилии в массиве должны быть уникальны.

5,16,27 Товарный чек содержит список товаров, купленных покупателем в магазине. Один элемент списка представляет собой пару: товар-сумма. Товар — это класс *Goods* с полями кода и наименования товара, цены за единицу товара, количества покупаемых единиц товара. В классе должны быть реализованы методы доступа к полям для получения и изменения информации, а также метод вычисления суммы оплаты за товар. Для моделирования товарного чека реализовать класс *Receipt*, полями которого являются номер товарного чека, дата и время его создания, список покупаемых товаров. В классе *Receipt* реализовать методы добавления, изменения и удаления записи о покупаемом товаре, метод поиска информации об определенном виде товара по его коду, а также метод подсчета общей суммы, на которую были осуществлены покупки.

6,17,28 Информационная запись о файле в каталоге содержит поля: имя файла, расширение, дата и время создания, атрибуты «только чтение», «скрытый», «системный», размер файла на диске. Для моделирования каталога реализовать класс *Directory*, содержащий название родительского каталога, количество файлов в каталоге, список файлов в каталоге. Один элемент списка включает в себя информационную запись о файле, дату последнего изменения, признак выделения и признак удаления. Реализовать методы добавления файлов в каталог и удаления файлов из него; метод поиска файла по имени, по расширению, по дате создания; метод вычисления полного объема каталога.

7,18,29 Используя класс *Bill* (разовый платеж за телефонный разговор, содержащий поля: фамилия плательщика, номер телефона, тариф за минуту разговора, скидка в процентах, время начала разговора, время окончания разговора, сумма к оплате), реализовать класс *ListPayer*. Класс содержит список плательщиков за телефонные услуги, дату создания списка, номер списка. Поля одного элемента списка — это: плательщик (класс *Bill*), признак оплаты, дата платежа, сумма платежа. Реализовать методы добавления плательщиков в список и удаления их из него; метод поиска плательщика по номеру телефона и по фамилии, по дате платежа. Метод вычисления полной стоимости платежей всего списка.

8,19,30 Учебный план специальности является списком дисциплин, которые студент должен изучить за время обучения. Одна дисциплина представляет собой структуру с полями: номер дисциплины в плане, тип

дисциплины (федеральная, региональная, по выбору), название дисциплины, семестр, в котором дисциплина изучается, вид итогового контроля (зачет или экзамен), общее количество часов, необходимое для изучения дисциплины, количество аудиторных часов, которые состоят из лекционных часов и часов практики. Реализовать класс *PlanEducation* для моделирования учебного плана специальности. Класс должен содержать код и название специальности, дату утверждения, общее количество часов специальности по стандарту и список дисциплин. Один элемент списка дисциплин должен содержать запись о дисциплине, количество часов для самостоятельной работы (разность между общим количеством часов и аудиторными часами), признак наличия курсовой работы, выполняемой по данной дисциплине. Реализовать методы добавления и удаления дисциплин; метод поиска дисциплины по семестру, по типу дисциплины, по виду итогового контроля; метод вычисления суммарного количества часов всех дисциплин; метод вычисления количества экзаменов и зачетов по семестрам.

9,20 Нагрузка преподавателя за учебный год представляет собой список дисциплин, преподаваемых им в течение года. Одна дисциплина представляется информационной структурой с полями: название дисциплины, семестр проведения, количество студентов, количество часов аудиторных лекций, количество аудиторных часов практики, вид контроля (зачет или экзамен). Реализовать класс *WorkTeacher*, моделирующий бланк назначенной преподавателю нагрузки. Класс содержит фамилию преподавателя, дату утверждения, список преподаваемых дисциплин, объем полной нагрузки в часах и в ставках. Дисциплины в списке не должны повторяться. Объем в ставках вычисляется как частное от деления объема в часах на среднюю годовую ставку, одинаковую для всех преподавателей кафедры. Элемент списка преподаваемых дисциплин содержит дисциплину, количество часов, выделяемых на зачет (0,35 ч на одного студента) или экзамен (0,5 ч на студента), сумму часов по дисциплине. Реализовать добавление и удаление дисциплин; вычисление суммарной нагрузки в часах и ставках.

10,21 Одна запись в списке запланированных дел представляет собой структуру *DailyItem*, которая содержит время начала и окончания работы, описание и признак выполнения. Реализовать класс *DailySchedule*, представляющий собой план работ на день. Реализовать методы добавления, удаления и изменения планируемой работы. При добавлении проверять корректность временных рамок (они не должны пересекаться с уже запланированными мероприятиями). Реализовать метод поиска свободного промежутка времени. Условие поиска задает размер искомого интервала, а также временные рамки, в которые он должен попадать. Метод поиска возвращает структуру *DailyItem* с пустым описанием вида работ. Реализовать

операцию генерации объекта *Redo* (еще раз), содержащего список дел, не выполненных в течение дня, из объекта типа *DailySchedule*.

11,22 Прайс-лист компьютерной фирмы включает в себя список моделей продаваемых компьютеров. Одна позиция списка – *Model* – содержит марку компьютера, тип процессора, частоту работы процессора, объем памяти, объем жесткого диска, объем памяти видеокарты, цену компьютера в условных единицах и количество экземпляров, имеющихся в наличии. Реализовать класс *PriceList*, полями которого являются дата его создания, номинал условной единицы в рублях и список продаваемых моделей компьютеров. В списке не должно быть двух моделей одинаковой марки. В классе *PriceList* реализовать методы добавления, изменения и удаления записи о модели, метод поиска информации о модели по марке компьютера, по объему памяти, диска и видеокарты (равно или не меньше заданного), а также метод подсчета общей суммы.