

### Теоретический материал

C# является полноценным объектно-ориентированным языком. Это значит, что программу на C# можно представить в виде взаимосвязанных взаимодействующих между собой объектов.

Описанием объекта является **класс**, а объект представляет экземпляр этого класса.

По сути класс представляет новый тип, который определяется пользователем. Класс определяется с помощью ключевого слова `class`:

```
class название_класса  
  
    {  
  
        // содержимое класса  
  
    }
```

Начиная с версии C# 12, если класс имеет пустое определение, то фигурные скобки после названия типа можно не использовать.

Класс может хранить некоторые данные. Для хранения данных в классе применяются **поля**. По сути **поля класса** - это переменные, определенные на уровне класса.

Кроме того, класс может определять некоторое поведение или выполняемые действия. Для определения поведения в классе применяются методы.

```
class Person

{

    public string name = "Undefined"; // имя

    public int age;           // возраст

    public void Print()

    {

        Console.WriteLine($"Имя: {name} Возраст: {age}");

    }

}
```

### **Создание объекта класса**

После определения класса мы можем создавать его объекты. Для создания объекта применяются конструкторы.

По сути конструкторы представляют специальные методы, которые называются так же как и класс, и которые вызываются при создании нового объекта класса и выполняют инициализацию объекта.

Общий синтаксис вызова конструктора:

```
new конструктор_класса(параметры_конструктора);
```

```
Person mike = new Person();
```

Для обращения к функциональности класса - полям, методам (а также другим элементам класса) применяется точечная нотация точки - после объекта класса ставится точка, а затем элемент класса:

**объект.поле\_класса**

**объект.метод\_класса(параметры\_метода)**

## Задание 1

### Задача:

#### Система бронирования

Разработать ПО со следующей архитектурой классов и функционалом:

#### Класс «Стол»:

Хранимая информация:

- ID стола;
- Расположение стола: (например: у окна, у прохода, у выхода, в глубине);
- Количество мест;
- Расписание занятости стола по часам.

Методы:

- Изменение информации стола;
- Создание стола;
- Вывод информации о столе.

#### Класс «Бронирование»:

Хранимая информация:

ID клиента;  
Имя клиента;  
Номер телефона клиента;  
Время начала брони;  
Время окончания брони;  
Комментарий;  
Назначенный столик.

Методы:

Создание брони;  
Изменение брони;

## Отмена брони.

Информация, вносимая в объекты класса «Бронирование», должна вносить изменения в объекты класса «Стол».

### Общие требования к функционалу:

- Программный продукт должен позволять создавать набор из  $n$  ( $n > 0$ ) столов (каждый стол представляет собой объект класса);
- Программный продукт должен позволять создавать набор из  $n$  ( $n > 0$ ) бронирований (каждое бронирование представляет собой объект класса);
- Иметь возможность редактирования информации выбранного стола по его ID (если он не фигурирует в активном бронировании);
- Иметь возможность вывода полной информации о столе по его ID;

\*\*\*\*\*

### Пример вывода информации о столе:

ID: -----01.

Расположение:-----«у окна».

Количество мест: -----4

Расписание:

9:00-10:00 -----

10:00-11:00 -----

11:00-12:00 -----

12:00-13:00 -----ID 3, Макс, 88005553535

13:00-14:00 -----ID 3, Макс, 88005553535


14:00-15:00 ----- ID 3, Макс, 88005553535

15:00-16:00 -----

16:00-17:00 ----- ID 7, Анна, 5745552377

17:00-18:00 -----

\*\*\*\*\*

	<ul style="list-style-type: none"> <li>• Вывод перечня всех доступных для бронирования столов, соответствующих фильтру;</li> <li>• Вывод перечня всех бронирований;</li> <li>• Поиск информации о бронировании по 4 последним цифрам номера телефона и имени клиента.</li> </ul> <p>Итоговый проект должен содержать 3 файла классов.</p>
<b>Решение:</b>	
	
<b>Ответ:</b>	
