Лабораторная работа № 1

Классы и объекты. Инкапсуляция

Цель лабораторной работы: познакомиться с созданием классов и объектов на языке C#. Изучить основные конструкции, используемые при построении классов и объектов. Разобраться с понятием инкапсуляции.

Теоретическая часть

Класс – это тип, описывающий устройство объектов. **Поля** – это переменные, принадлежащие классу.

Методы – это функции (процедуры), принадлежащие классу.

Объект – это экземпляр класса, сущность в адресном пространстве компьютера.

Можно сказать, что класс является шаблоном для объекта, описывающим его структуру и поведение. Поля класса определяют структуру объекта, методы класса – поведение объекта.

С точки зрения практической реализации (в самом тексте программы) класс является типом данных, а объект – переменной этого типа.

Объявление класса состоит из двух частей: объявление заголовка класса и объявление тела класса. Заголовок класса состоит из модификатора доступа, ключевого слова **class** и имени самого класса. **Тело класса** – есть конструкция, заключенная в фигурные скобки и содержащая объявление полей и методов, принадлежащих классу.

Пример объявления класса:

```
public class MyClass {
    int a;
}
```

Объявление объекта (создание объекта как экземпляра класса) состоит из двух частей: создание переменной-ссылки на область памяти, в которой будет располагаться объект, выделение памяти для объекта и заполнение этой памяти начальными значениями. Иначе говоря, инициализация данной переменной-ссылки. Объявление переменной-ссылки, а иными словами объекта, подчиняется общему правилу объявления переменных в C#. Переменные могут объявляться в любом месте в теле методов, за исключением тел условных операторов. Переменная, объявленная вне тела метода, но внутри тела класса, становится полем. Синтаксис объявления переменных имеет вид:

<имя_типа> <имя_переменной>.

Пример объявления целочисленной переменной:

```
int a;
```

Пример объявления переменной-объекта класса MyClass:

```
MyClass MyObj;
```

При объявлении переменная может быть сразу инициализирована (ей может быть присвоено какое-либо значение).

Пример инициализации переменной при объявлении: **int a=0**;

Выделение памяти осуществляет оператор **new**, а задачу заполнения памяти начальными значениями решает специальный метод объекта, называемый конструктором. **Конструктор** – метода объекта, объявленный следующим образом: для этого метода всегда используется модификатор доступа **public**, нет типа возвращаемого значения (нет даже void), имя метода совпадает с именем класса. Однако компилятор C# не требует обязательного определения конструктора для класса. Если конструктор не объявлен, компилятор вызовет так называемый конструктор по умолчанию, который создаст сам.

Таким образом, создание объекта класса MyClass будет иметь вид:

```
MyClass MyObj = new MyClass();
```

Классы (и объекты) являются прямым воплощением такой идеи объектно-ориентированного программирования как инкапсуляция данных и кода.

Инкапсуляция – это механизм объединения данных и кода, манипулирующего этими данными, а также защиты того и другого от внешнего вмешательства, неправильного использования или от несанкционированного доступа. **Объект** – это то, что поддерживает инкапсуляцию (объединяет в себе данные и код, работающий с ними). Для закрытия данных внутри объекта используются **модификаторы доступа**, в частности модификатор **private**, который используется по умолчанию, если не употреблен другой. Данные или методы, объявленные с этим модификатором, будут недоступны вне класса (объекта), то есть к ним можно будет обратиться через открытые (объявленные с модификатором **public**) методы или свойства класса. **Свойства класса** – нечто среднее между полем и методом, представляет собой конструкцию вида:

```
<Модификатор доступа><Тип свойства><Имя свойства>
        {
            get{return <значение>}
            set{<поле>=value}
        }
```

Обычно свойства связываются с закрытыми полями класса и помогают осуществить доступ к этим полям из внешних (относительно класса) частей программы. Свойства вместе с модификаторами доступа реализуют механизм защиты данных от несанкционированного доступа. Как мы видим, свойство имеет заголовок и тело. В заголовке указывается модификатор доступа (обычно **public**), тип возвращаемого свойством значения и имя свойства. В теле объявлено два метода **get** и **set**. Больше ничего в теле свойства объявлять нельзя. Метод **get** имеет ключевое слово **return** и возвращает какое-либо значение (обычно значение какого-либо поля, хотя не обязательно). Метод **set** имеет ключевое слово **value** и присваивает (устанавливает) это значение полю объекта.

Пример объявления свойства в классе MyClass:

```
public class MyClass
{
    int a; //поле
    public int A// свойство
    {
        get { return a;}
        set { a=value;}
    }
}
```

Пример использования описанного свойства в программе:

```
MyClass MyObj = new MyClass();
MyObj.A = 6; // полю a объекта MyObj присвоится
значение 6.
int b = MyObj.A; // переменной b присвоится
значение поля a объекта MyObj.
```

Так как программа на языке C# может иметь множество классов со множеством методов, то необходимо каким-то образом определять точку, откуда начнется выполнятся программа. Эта точка называется точкой входа и представляет собой метод любого класса, объявленный с заголовком **static void Main(string[] args)**. Точка входа может принадлежать любому классу, из описанных в программе, но должна быть в программе одна.

Практическая часть

В рамках консольного приложения создать класс A с полями a и b и свойством c. Свойство – значение выражения над полями a и b (выражение и типы полей – см. вариант). Поля инициализировать при объявлении класса. Конструктор оставить по умолчанию. Проследить, чтобы поля a и b напрямую в других классах были недоступны. Создать класс Programm с одним методом – точкой входа. В теле метода создать объект класса A, вывести на экран значение свойства c.

Таблица 1. Варианты заданий

Вариант	Тип полей a b,	Выражения
1	float	/,-
2	float	/=, +
3	int	*,-
4	int	*=,+
5	int	(постфиксный)++,*=,-
6	int	%=,+
7	float	/=,(постфиксный)--,*
8	float	*=,++(префиксный),/
9	decimal	+=,-
10	decimal	(постфиксный)++,-
11	decimal	++(префиксный),-