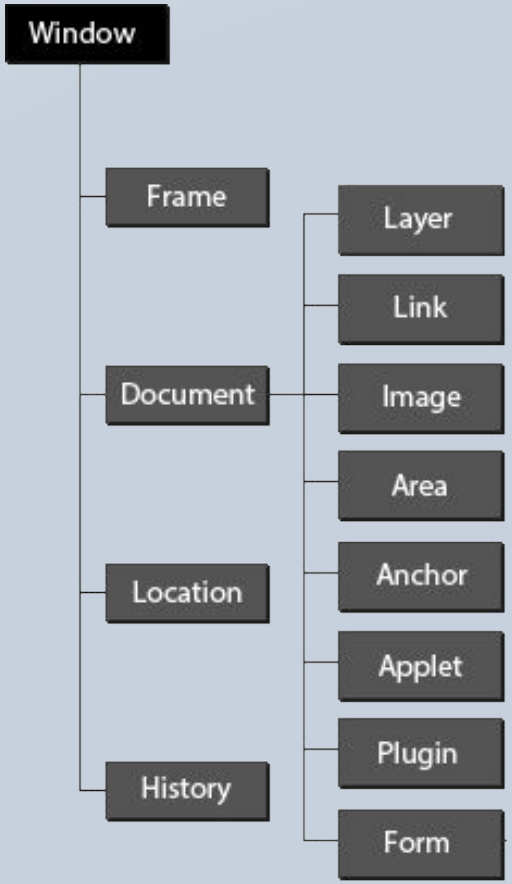
РЕАЛИЗАЦИЯ КЛАССОВ И ОБЪЕКТОВ

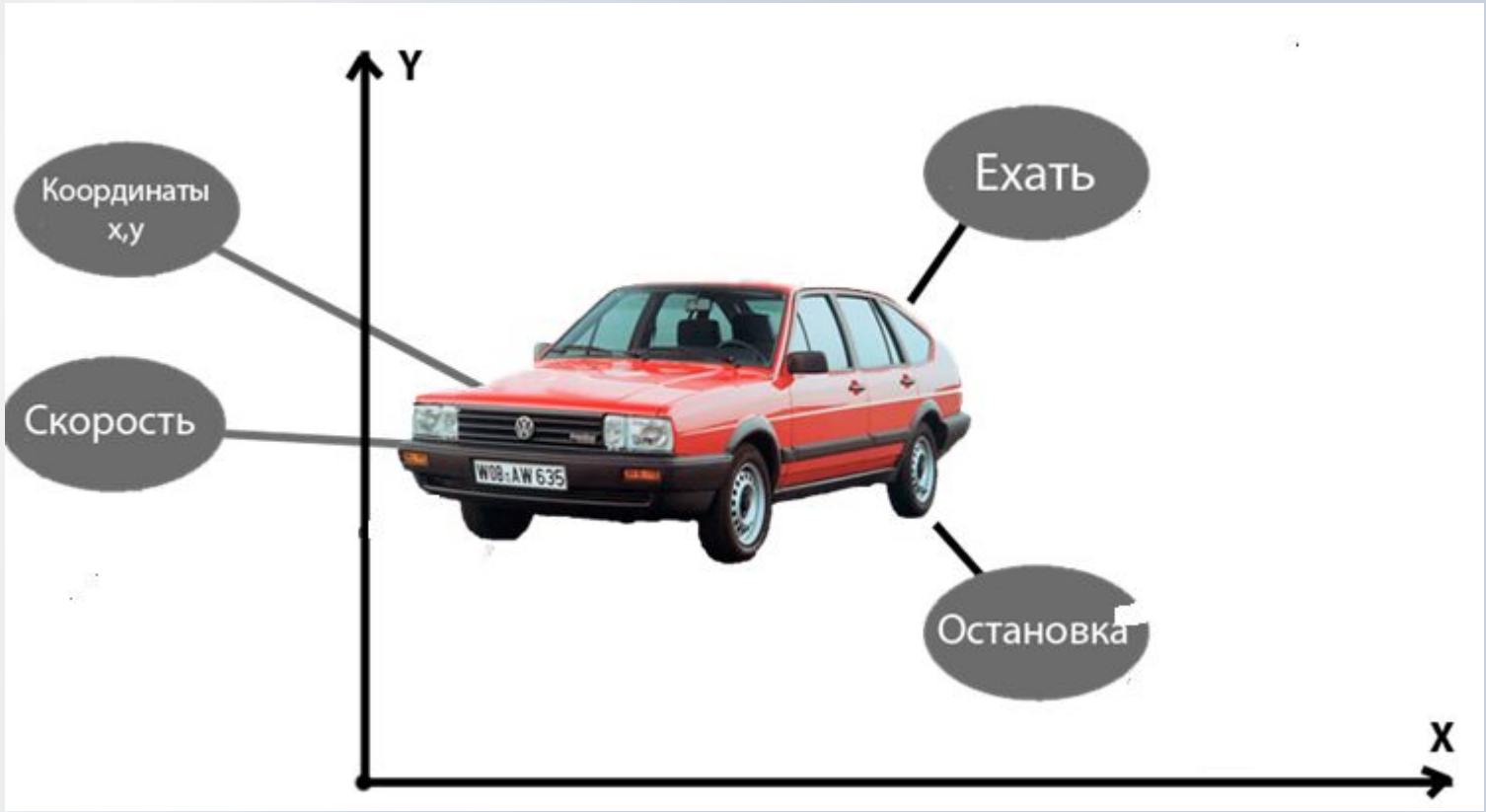


Понятие объекта

Объект – это некоторая сущность в виртуальном пространстве.
Каждый объект обладает:

- **свойствами (атрибутами, характеристикам, свойствами);**
- **методами (действиями, операциями, умениями).**

Пример : автомашина - пассат



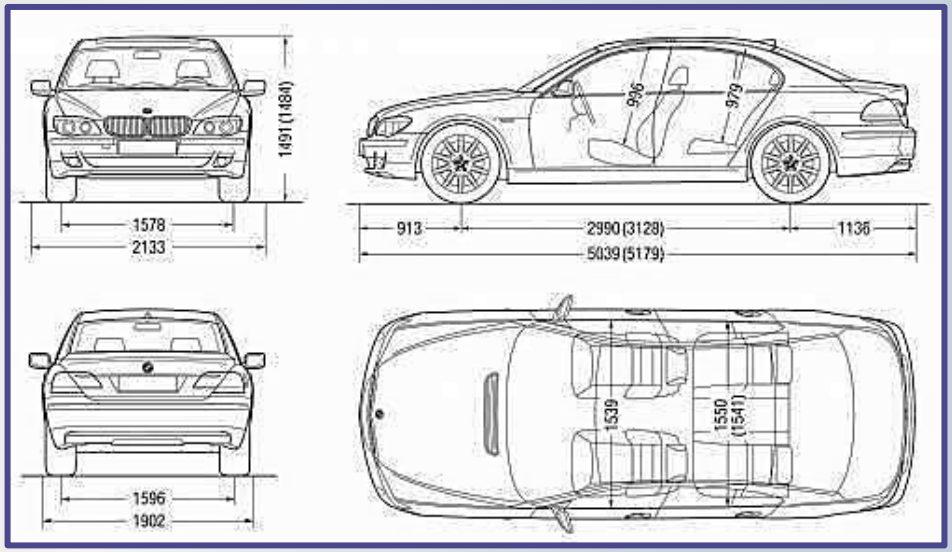
Понятие класса

Класс - шаблон, на основе которого строятся однотипные (похожие) объекты, имеющие одинаковый набор характеристик

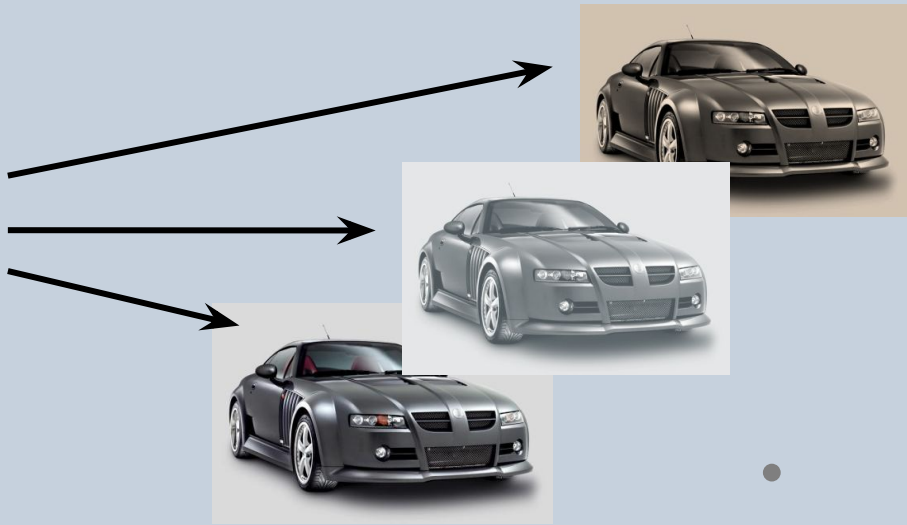
Объекты одного класса отличаются значениями своих характеристик

Отличающиеся по функционалу объекты находятся в разных классах (автомашина и ПК)

Класс



Объекты



Реализация класса и объекта

Класс – это новый тип. Объект – переменная типа класс

1. Объявление объекта

```
Класс   Объект;
```

2. Выделение памяти под объект ссылочного типа

```
Объект = new Класс();
```

Пример: создание объекта «Машина» класса «Авто»

```
Авто   машина;
машина = new   Авто();
```

Объявление объекта

Выделение памяти

Оба действия можно объединить

```
Класс   объект = new Класс();
```

```
Авто   машина = new   Авто();
```

Примеры работы с объектами

```
//Создание двух объектов одного класса
Авто  Машина1 = new  Авто();
Авто  Машина2 = new  Авто();
//Общение к свойствам (задание значений)
Машина1.Координата X = 100;
Машина2.Координата X = 200;
Машина2.Скорость = 75;
//Обращение к методам (вызов)
Машина1.Ехать(50,30);
Машина1.Врезаться(Машина2);
Машина2.Стоп();
```

Создание класса на C#

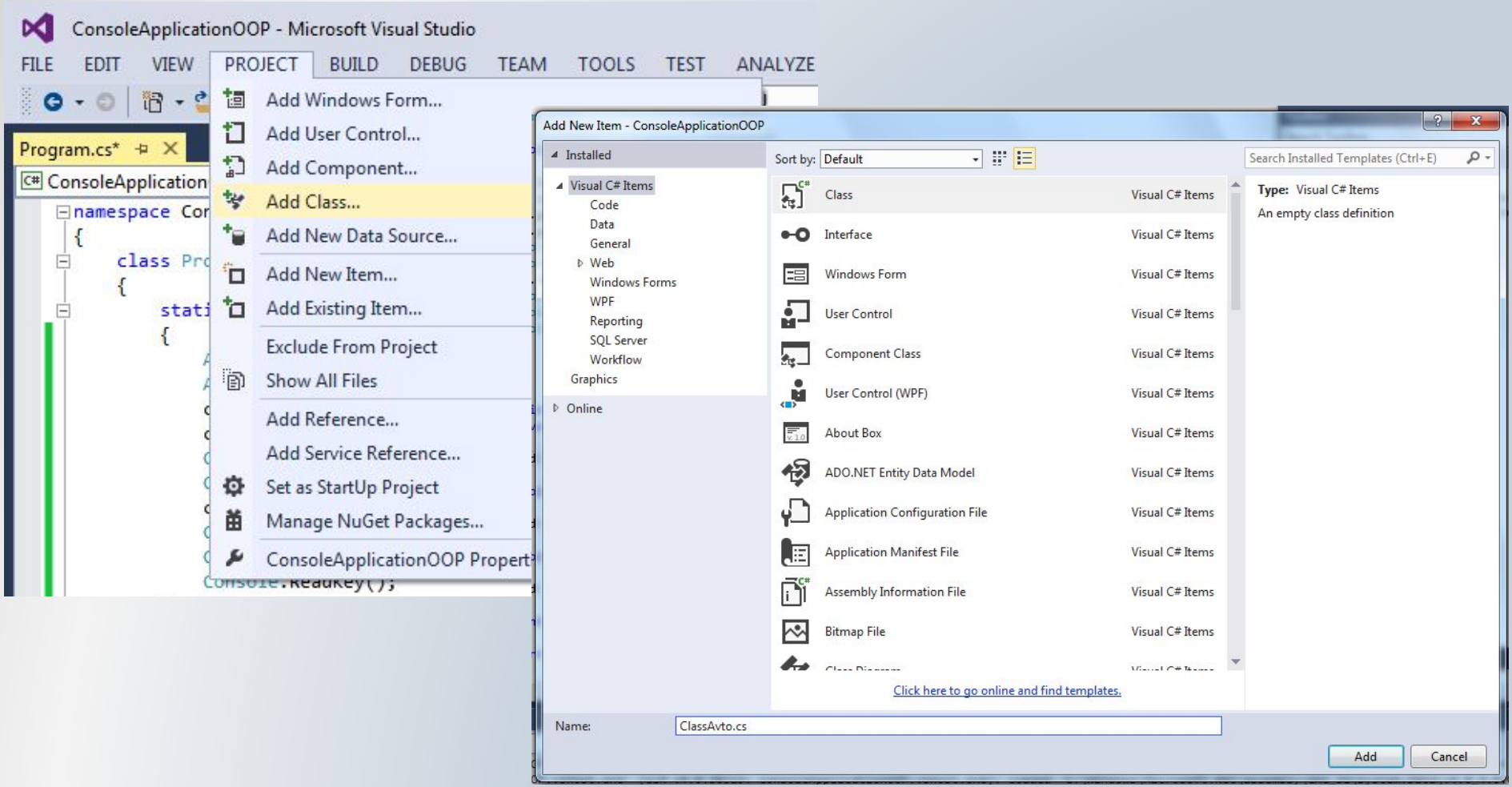
Шаблон класса C#

```
[public] class имя_класса
{
   [уровень доступа] [static] поле
   [уровень доступа] [static] метод
}
```

Если класс содержится в том же пространстве (namespace) что и class Program с Main() (один проект), то public можно не указывать. Если класс будет использоваться в других проектах (как dll-библиотека), то public обязательно

Создание файла с классом

Обычно класс размещается в отдельном файле проекта с расширением cs.



Уровень доступа

Уровень доступа определяет область кода, в пределах которой можно использовать класс и его члены

Уровни доступа:

- public – открытый, доступный везде (в классе и вне класса)
- private – закрытый, только внутри класса (по умолчанию)
- protected – защищенный, доступ только для родственных классов

В раздел **private** помещают члены класса, которые необходимо защитить, спрятать (инкапсуляция). Они доступны только внутри класса и не доступны за пределами класса

В раздел **public** помещают члены класса, которые необходимо сделать открытыми, доступными. Средство общения классов. Доступны вне класса

static:

Использование члена класса (поле или метод) без объекта.

Правила объявления класса

Правила на поля (свойства) класса

1. Поля схожи с переменными: указываются тип и имя
2. Обычно поля имею private-доступ (инкапсуляция)
3. Обычно static-поля имеют public-доступ.
4. Доступ к полям только методами этого класса.

Метод другого класса не имеет доступа к полям.

Правила объявления класса

Правила на методы класса

1. Метод раздела public доступен из любых мест программы
2. Метод раздела private доступен только в методах этого класса
3. Обычно метод имеет public-доступ, чтобы использовать его при взаимодействии с другими объектами
4. Методы могут быть перегружены в одном классе (с разным типом и числом параметров).

Пример класса на C#

```
class Passat
{
  //Поля-свойства
  private int speed;                    //Скорость машины
  //Методы
  private void stop()                   //Остановка-закрытый
     {
     speed = 0;
     }
  public void accident (Passat pas)   //Авария
     {
      this.stop();                      //Вызов метода к текущей машине
      pas.stop();                       //Вызов метода к машине-параметру
     }
  public void gaz(int g)                //Разгон
     { speed += g;   }
  public void brake(int g)              //Торможение
     { speed -= g;   }
  public void print()                   //Вывод характеристик
     {
        Console.WriteLine("Текущая скорость = " + this.speed);
     }
}
```

Описание объекта

Класс – это новый тип. Память под класс не выделяется

Объект – это переменная типа класс.
Под объект выделяется память

Шаблон объявления объекта

```
Имя_класса   Имя_объекта;
```

Шаблон создания объекта конструктором

```
Имя_объекта = new Имя_класса();
```

Можно объединить

```
Имя_класса Имя_объекта = new Имя_класса();
```

Описание объекта

Работа с объектами происходит в блоке Main()

Создание объектов типа Passat

```
class Program
{
    //Главный метод программы
    static void Main()
    {
     Passat   car; //Объявили один объект
     car = new Passat();  //Создать
     Passat[] cars = new Passat[n]; //Массив
     for (int  i = 0; i < cars.Length; i++)
       cars[i] = new Passat();
     }
}
```

Замечания

Для каждого объекта выделяется своя память под свойства.

Изменения в одном объекте не затрагивают свойства другого объекта(инкапсуляция)

Обращение к полям и методам

В Main() доступны только public-члены нового класса, т.к. Main() принадлежит другому классу (Program)

Обращение к свойству

```
Объект.Свойство = значение; //Нестатическое
Класс.Свойство = значение;  //static
```

Обращение к методу

```
Результат = Объект.Метод(параметры);
Результат = Класс.Метод(параметры);
```

Благодаря составному имени свойства и метода, можно изменять свойства конкретного объекта

Доступ к членам класса

Пример доступа к членам класса - методам

```
static void Main()
{
 …
 car.gaz(50);      //Разгон одной машины
 cars[1].gaz(50); //Разгон 2-ой машины
 cars[1].print(); //Вывести текущую скорость
 cars[2].stop();  //Ошибка – нет доступа
 car.speed = 100; //Ошибка – нет доступа
}
```

Ключевое слово this

Слово **this** означает ссылку на объект, для которого вызывается метод. Обеспечивает доступ к членам объекта из метода.

Используется, чтобы отличить доступ к полю объекта класса от параметра метода при их одинаковых именах

```
class Example
{
  private int  code;        //Закрытое поле
  public  int  Get()        //Открытый метод
   {
    return  this.code;  //Не обязательно this
   }
  public void Set(int  code)
   {
    //Обязательно this для отличия поля от параметра
    this.code = code;
   }
}
```

Практическая работа на ПК

«Классы и объекты»

Создать класс «Avto».

Свойства класса (private):

- номер авто – строка
- количество бензина в баке – целое
- расход топлива на 100 км – вещественное

Методы класса:

- заполнение информацией о машине:
  void info(string nom, float bak, float ras);
- вывод информации: void out(void);
- заправка топливом:void zapravka(float top);
- езда: void move(int km);
- остаток топлива: int ostatok(void); (private)

Оценивание практической работы

Реализацию класса разместить в отдельном файле cs. Выполнить пункт меню к проекту «Добавить класс».

На «Удовлетворительно»

Создать один объект на основе класса. Заполнить бак бензином (объем вводит пользователь). Организовать цикл езды: случайно получить расстояние, по расходу определить, доедет ли. Если да, то ехать, иначе спросить о возможной заправке. Дополнительно реализовать public-методы «Торможение» и «Разгон».

На «Хорошо»

Добавить поле private «Пробег». Разработать методы:

- расчет расстояния между начальной и конечной координатами машины (private) . Вызывать внутри метода «езда»
- расчет общего пробега с учетом пройденного расстояния (public)

На «Отлично»

Реализовать несколько машин (массив). Создать метод «Авария». Случайно задавать случайные индексы для машин с аварией.

Практическая работа на ПК

«Классы и объекты»

Оценивание работы

Пояснения

Реализацию класса разместить в отдельном файле: класс.cpp.
С помощью команды #include “класс.cpp” подключить класс к основной программе.

На оценку «Удовлетворительно»

Выполнить действия для одного объекта

На оценку «Хорошо»

Выполнить действия с массивом объектов

На оценку «Отлично»

Добавить утилиты:

- **сохранение данных об объекте в файле;**
- **восстановление данных об объекте из файла.**

Практическая работа

Постановка задачи:

Создать класс «Счет в банке». Реализовать методы класса, влияющие на значения свойств.

Поля данных класса – свойства будущего объекта:

- **№ счета (целое)**
- **ФИО владельца счета (строка)**
- **Сумма на счету (вещественное)**

Методы класса – действия с будущим объектом:

- **Открыть счет: void otk (int nom, char *name, float sum);**
- **Показать информацию о счете: void out ();**
- **Положить на счет: void dob (float sum);**
- **Снять со счета: void umen (float sum);**
- **Взять всю сумму: void obnul ();**
- **Перенести сумму с одного счета на другой**

Практическая работа на ПК

«Классы и объекты»

Создать класс «Игровой персонаж».

Свойства класса:

- название персонажа - строка
- координаты расположения - целые
- лагерь (друг/враг)- логическое
- количество жизней - целое

Методы класса:

- заполнение информацией:
  `void info (char *s, int x, int y, bool l, int kol);`
- вывод информации: `void print (void);`
- перемещение по горизонтали:`void movex (int dx);`
- перемещение по вертикали: `void movey (int dy);`
- уничтожение: `void del(void);`
- нанесение урона: `void uron (int du);`
- лечение: `void doc (int du);`
- полное восстановление: `void vost(void);`
- принадлежность лагерю: `bool lager(void);`

Оценивание практической работы

На «удовлетворительно»

Создать один объект на основе класса. Выполнить над объектом все методы через пункты меню

На «хорошо»

Создать массив объектов из 10 элементов. Отдельные поля данных заполнить случайными числами. Обработать по номеру отдельные элементы массива через пункты меню.

На «отлично»

Пользователь определяет количество объектов. Занести информацию в динамический массив объектов. Выполнить основные операции с элементами массива.