Абстрактные классы

- Абстрактные классы предназначены для представления **общих понятий**, которые предполагается конкретизировать в производных классах.
- *Абстрактный класс задает интерфейс для всей иерархии.*
- Абстрактный класс задает набор методов, которые каждый из потомков будет реализовывать по-своему.
- Методы абстрактного класса могут *иметь пустое тело* (объявляются как **abstract)**.
- Абстрактный класс может содержать и полностью определенные методы (в отличие от интерфейса).
- Если в классе есть хотя бы один абстрактный метод, весь класс должен быть описан как **abstract**.
- Если класс, производный от абстрактного, не переопределяет все абстрактные методы, он также должен описываться как абстрактный.

Применение абстрактных классов

- *Абстрактный класс служит только для порождения потомков*.
- Абстрактные классы используются:
  - при работе со структурами данных, предназначенными для хранения объектов одной иерархии
  - в качестве параметров полиморфных методов

- Методу, параметром которого является абстрактный класс, при выполнении программы можно передавать объект любого производного класса.
- Это позволяет создавать *полиморфные методы*, работающие с объектом любого типа в пределах одной иерархии.

Бесплодные (финальные) классы

- Ключевое слово **sealed** позволяет описать класс, от которого, в противоположность абстрактному, наследовать запрещается:

```
sealed class Spirit    {  ...   }
//  class Monster : Spirit { ... }          ошибка!
```

- Большинство встроенных типов данных описано как sealed. Если необходимо использовать функциональность бесплодного класса, применяется не наследование, а *вложение*, или *включение*: в классе описывается поле соответствующего типа.
- Поскольку поля класса обычно закрыты, описывают метод объемлющего класса, из которого вызывается метод включенного класса. Такой способ взаимоотношений классов известен как *модель включения-делегирования* (*об этом – далее*).

Класс object

- Корневой класс System.Object всей иерархии объектов .NET, называемый в C# object, обеспечивает всех наследников несколькими важными методами.
- Производные классы могут использовать эти методы непосредственно или переопределять их.

- Класс object используется непосредственно:
  - при описании типа параметров методов для придания им общности;
  - для хранения ссылок на объекты различного типа.

Открытые методы класса System.Object

public virtual bool Equals(object obj);

- возвращает true, если параметр и вызывающий объект ссылаются на одну и ту же область памяти

public static bool Equals(object ob1, object ob2);

- возвращает true, если оба параметра ссылаются на одну и ту же область памяти

public virtual int GetHashCode();

- формирует хэш-код объекта и возвращает число, однозначно идентифицирующее объект

public Type GetType();

- возвращает текущий полиморфный тип объекта (не тип ссылки, а тип объекта, на который она в данный момент указывает)

public static bool ReferenceEquals(object ob1, object ob2);

- возвращает true, если оба параметра ссылаются на одну и ту же область памяти

public virtual string ToString()

- возвращает для ссылочных типов полное имя класса в виде строки, для значимых — значение величины, преобразованное в строку. Этот метод переопределяют, чтобы выводить информацию о состоянии объекта.

Пример переопределения метода Equals

```
// сравнение значений, а не ссылок

public override bool Equals( object obj ) {
    if ( obj == null || GetType() != obj.GetType() ) return false;
    Monster temp = (Monster) obj;
    return  health == temp.health &&
            ammo   == temp.ammo   &&
            name   == temp.name;
}
public override int GetHashCode()
{
    return name.GetHashCode();
}
```

Рекомендации по программированию

- Главное преимущество наследования состоит в том, что **на уровне базового класса можно написать универсальный код**, с помощью которого работать также с объектами производного класса, что реализуется с помощью виртуальных методов.
- Как виртуальные должны быть описаны методы, которые выполняют во всех классах иерархии одну и ту же функцию, но, возможно, разными способами.
- Для представления общих понятий, которые предполагается конкретизировать в производных классах, используют абстрактные классы. Как правило, в абстрактном классе задается набор методов, то есть интерфейс, который каждый из потомков будет реализовывать по-своему.
- Обычные методы (не виртуальные) переопределять в производных классах не рекомендуется.

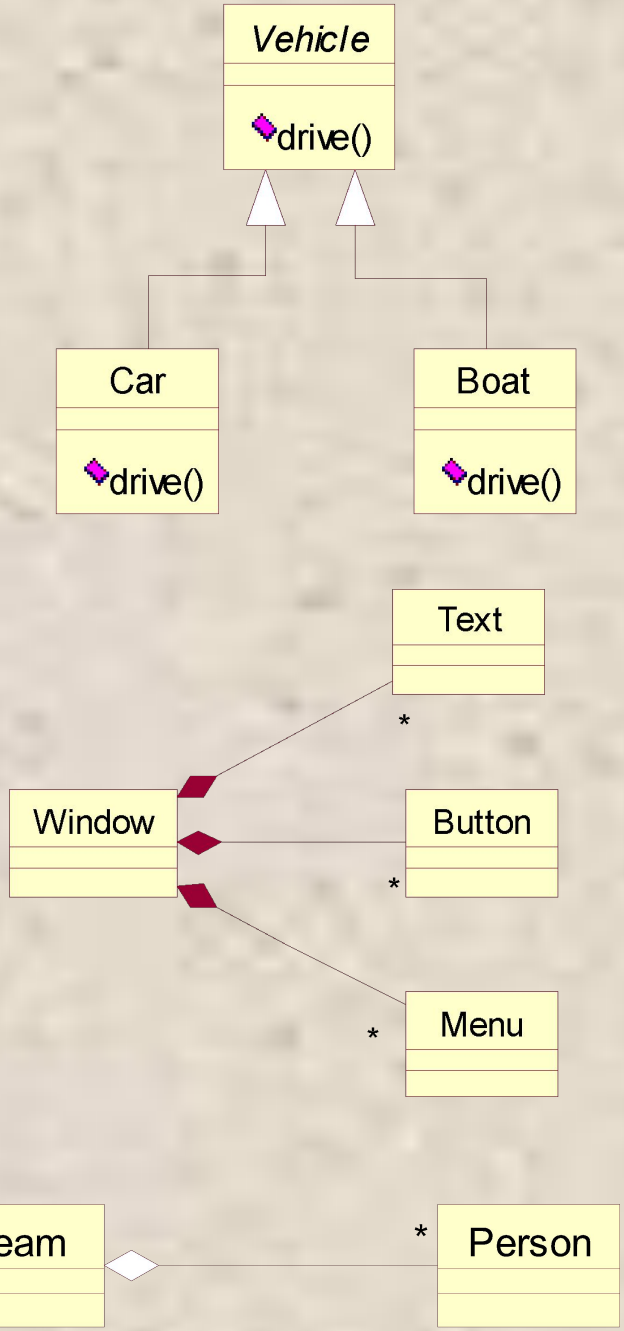
Виды взаимоотношений между классами

- Наследование
  - Специализация (Наследник является специализированной формой предка)
  - Спецификация (Дочерний класс реализует поведение, описанное в предке)
  - Конструирование или Варьирование (Наследник использует методы предка, но не является его подтипом; предок и потомок являются вариациями на одну тему – например, прямоугольник и квадрат)
  - Расширение (В потомок добавляют новые методы, расширяя поведение предка)
  - Обобщение (Потомок обобщает поведение предка)
  - Ограничение (Потомок ограничивает поведение предка)
- Вложение
  - композиция
  - агрегация

Классификация Тимоти Бадда

Наследование и вложение

- *Наследование* класса Y от класса X чаще всего означает, что Y представляет собой разновидность класса X (более конкретную, частную концепцию).
- *Вложение* является альтернативным наследованию механизмом использования одним классом другого: один класс является полем другого.
- *Вложение* представляет отношения классов «Y содержит X» или «Y реализуется посредством X» и реализуется с помощью модели «включение-делегирование».



Модель включения-делегирования

```
class Двигатель {public void Запуск() {Console.WriteLine( "вжжж!!" ); }}

class Самолет
   {     public Самолет()
            {    левый  = new Двигатель(); правый = new Двигатель(); }
         public void Запустить_двигатели()
            {     левый.Запуск();  правый.Запуск();    }
         Двигатель левый, правый;
   }
   class Class1
   {   static void Main()
      {        Самолет АН24_1 = new Самолет();
               АН24_1.Запустить_двигатели();
      }
   }
```

Результат работы программы:
вжжж!!
вжжж!!