Объектно-ориентированное программирование

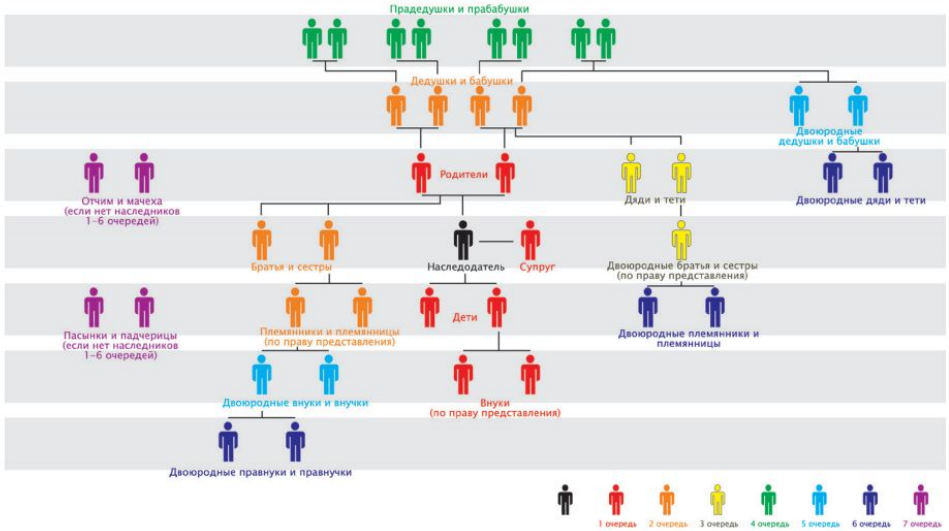
Агрегация и композиция

Агрегация и композиция

В реальном мире не существует объектов, не связанных с другими.

Агрегация и композиция

Основные отношения – наследование и ассоциация.



Агрегация и композиция

Ассоциации бывают различных видов (агрегация и композиция).

Агрегация

Некоторые объекты слабо взаимосвязаны
друг с другом, как компьютер
и его периферийное оборудование

Композиция

Некоторые объекты тесно взаимосвязаны,
как дерево и его листья

Ассоциации

Ассоциация показывает, что объекты одной сущности (класса) связаны с объектами другой сущности.

Существует пять различных типов ассоциации. Наиболее распространёнными являются *двунаправленная* и *однонаправленная*.

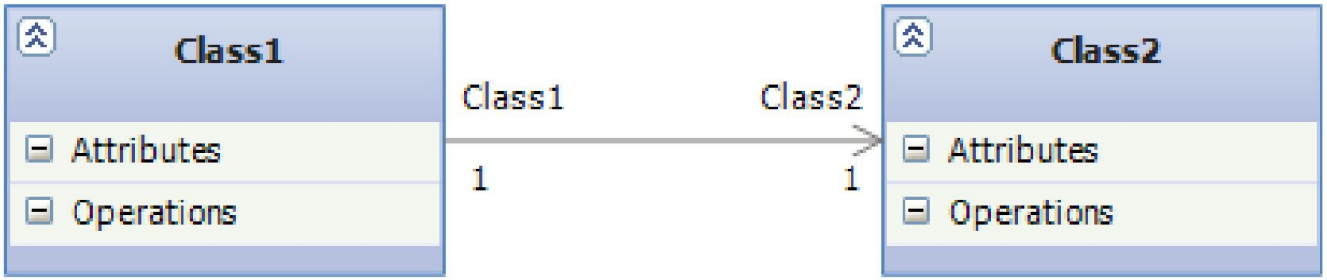
Например, классы «рейс» и «самолёт» связаны двунаправленной ассоциацией.

Ассоциации

а классы «человек» и «кофейный автомат» связаны однонаправленной.

Ассоциации

Ассоциация на диаграмме обозначается линией без стрелки или со стрелкой. Обычно в ходе дальнейшего проектирования уточняется – агрегация или композиция.



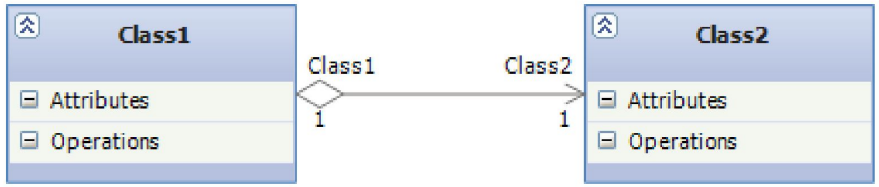
Агрегация

Агрегация — это разновидность ассоциации при отношении между целым и его частями.

Агрегация

Агрегация встречается, когда один класс является коллекцией или контейнером других. Причём по умолчанию, агрегацией называют *агрегацию по ссылке*, то есть когда время существования содержащихся классов не зависит от времени существования содержащего их класса. Если контейнер будет уничтожен, то его содержимое — нет.

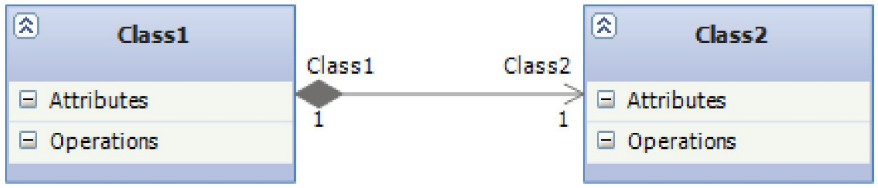
Графически агрегация представляется пустым ромбиком на блоке класса и линией, идущей от этого ромбика к содержащемуся классу.



Композиция

Композиция — более строгий вариант агрегации. Известна также как агрегация по значению.

Композиция имеет жёсткую зависимость времени существования экземпляров класса контейнера и экземпляров содержащихся классов. Если контейнер будет уничтожен, то всё его содержимое будет также уничтожено.

Графически предс  с закрашенным ром(

Различия между композицией и агрегацией

Комната является частью квартиры, следовательно здесь подходит *композиция*, потому что комната без квартиры существовать не может.

Различия между композицией и агрегацией

А, например, мебель не является неотъемлемой частью квартиры, но в то же время, квартира содержит мебель, поэтому следует использовать *агрегацию*.

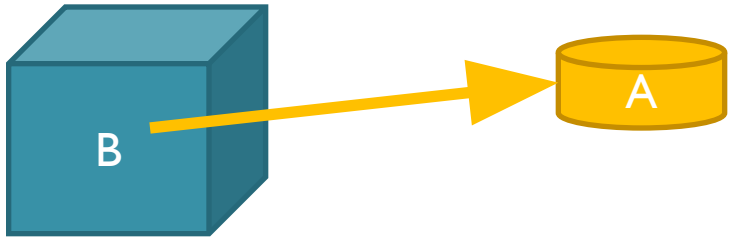
Агрегация и композиция

Допустим, существует некий класс A

```
class A
{
   ...
}
```

Агрегация

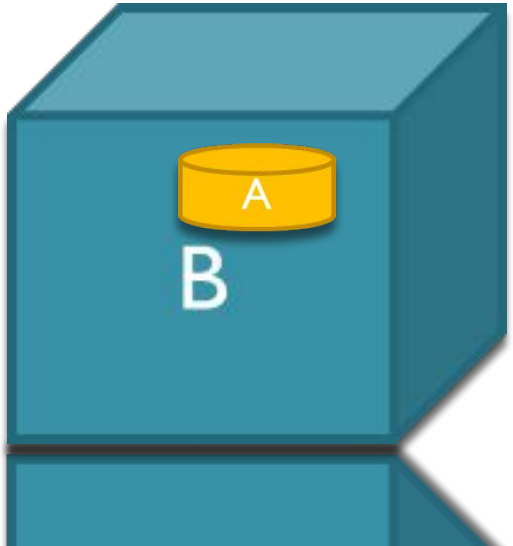
агрегация




```
class B
{
   private A _a;
   public B(A a)
   // Объект A живет где-то отдельно
   // (суть не в конструкторе)
       {
          _a = a;
       }
}
```

Композиция

```
class B
{
   private A _a = new A();
   // Объект A существует только вместе с B
}
```

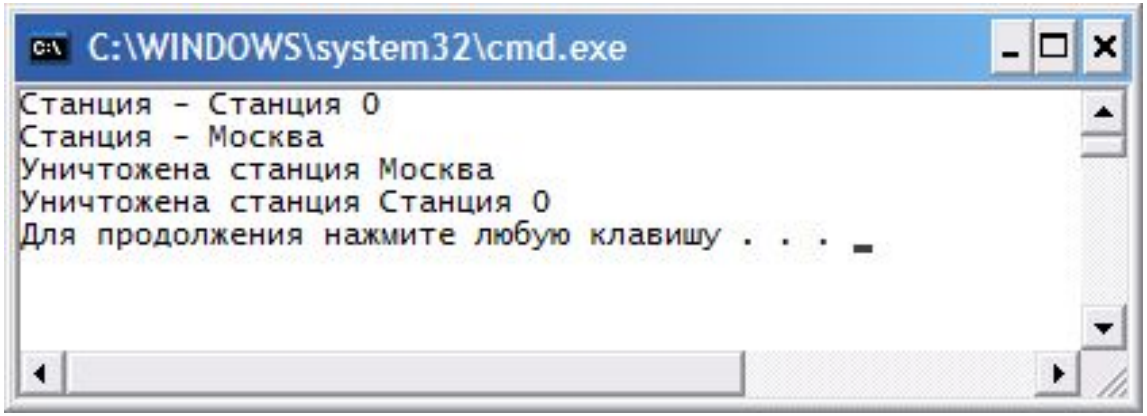


Класс CStation

```
class CStation
{
    public string name; // название станции
    public CStation()
    {
        name = "Станция 0";
    }
    public CStation(string name)
    {
        this.name = name;
    }
    public void Print()
    {
        Console.WriteLine("Станция - " + name);
    }
    ~CStation()
    {
        Console.WriteLine("Уничтожена станция " + name);
    }
}
```

Класс CStation

```
CStation s1 = new CStation();
CStation s2 = new CStation("Москва");
s1.Print();
s2.Print();
```



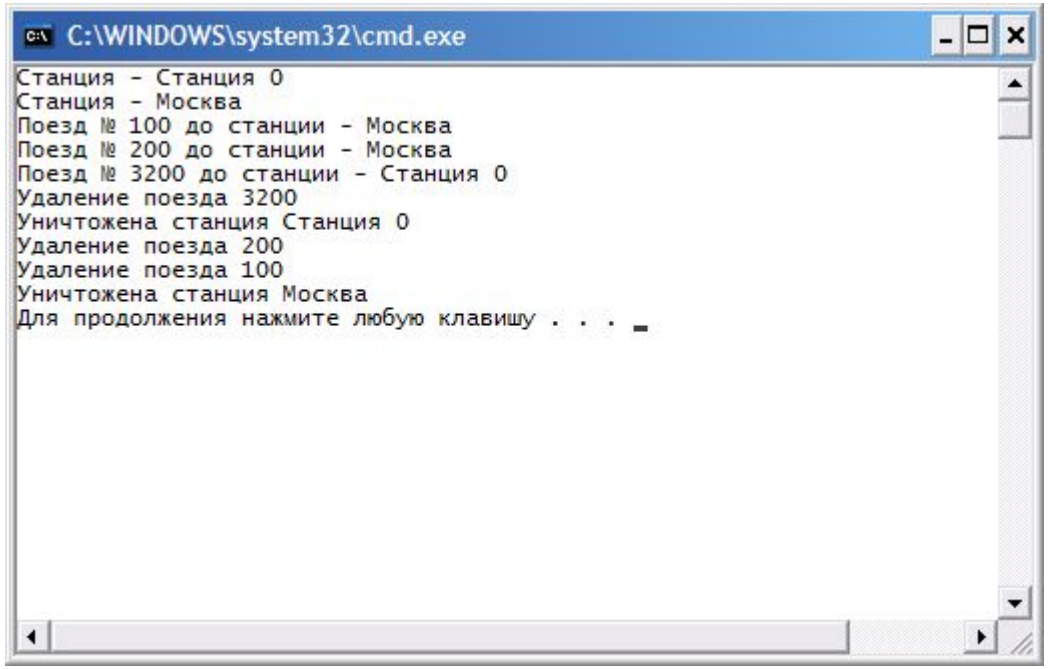
Класс CTrain

```
class CTrain
 {   public int n;//номер поезда
     public CStation st;// станция назначения
     public CTrain()
     {   n = 0;
         st = null;
     }
     public CTrain(int n, CStation st)
     {   this.n = n;
         this.st = st;
     }
     public void Print()
     {  Console.WriteLine("Поезд № " + n + " до станции - " +
st.name); }
     ~CTrain()
     {  Console.WriteLine("Удаление поезда " + n); }
 }
```

Агрегация

```
static void Main(string[] args)
{
    CStation s1 = new CStation();
    CStation s2 = new CStation("Москва");
    s1.Print();
    s2.Print();
    CTrain tr1 = new CTrain(100,s2);
    tr1.Print();
    CTrain tr2 = new CTrain(200, s2);
    tr2.Print();
    CTrain tr3 = new CTrain(3200, s1);
    tr3.Print();
}
```

Агрегация


C:\WINDOWS\system32\cmd.exe
Станция - Станция 0
Станция - Москва
Поезд № 100 до станции - Москва
Поезд № 200 до станции - Москва
Поезд № 3200 до станции - Станция 0
Удаление поезда 3200
Уничтожена станция Станция 0
Удаление поезда 200
Удаление поезда 100
Уничтожена станция Москва
Для продолжения нажмите любую клавишу . . .

Композиция

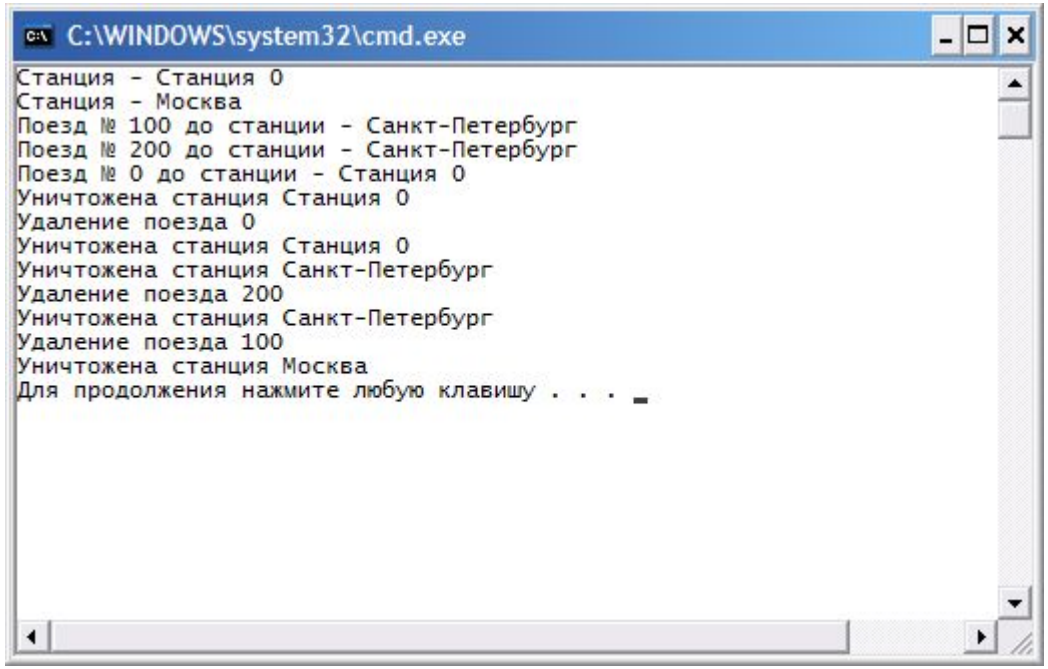
```
 class CTrain
{
    public int n;//номер поезда
    public CStation st;// станция назначения
    public CTrain()
    {
        n = 0;
        st = new CStation("Станция 0");
    }
    public CTrain(int n, string str)
    {
        this.n = n;
        this.st = new CStation(str);
    }
    public void Print()
    {
        Console.WriteLine("Поезд № " + n + " до станции - " + st.name);

    }
```

Композиция

```
static void Main(string[] args)
{
    CStation s1 = new CStation();
    CStation s2 = new CStation("Москва");
    s1.Print();
    s2.Print();
    CTrain tr1 = new CTrain(100,"Санкт-Петербург");
    tr1.Print();
    CTrain tr2 = new CTrain(200, "Санкт-Петербург");
    tr2.Print();
    CTrain tr3 = new CTrain();
    tr3.Print();
}
```

Композиция


C:\WINDOWS\system32\cmd.exe
Станция - Станция 0
Станция - Москва
Поезд № 100 до станции - Санкт-Петербург
Поезд № 200 до станции - Санкт-Петербург
Поезд № 0 до станции - Станция 0
Уничтожена станция Станция 0
Удаление поезда 0
Уничтожена станция Станция 0
Уничтожена станция Санкт-Петербург
Удаление поезда 200
Уничтожена станция Санкт-Петербург
Удаление поезда 100
Уничтожена станция Москва
Для продолжения нажмите любую клавишу . . .