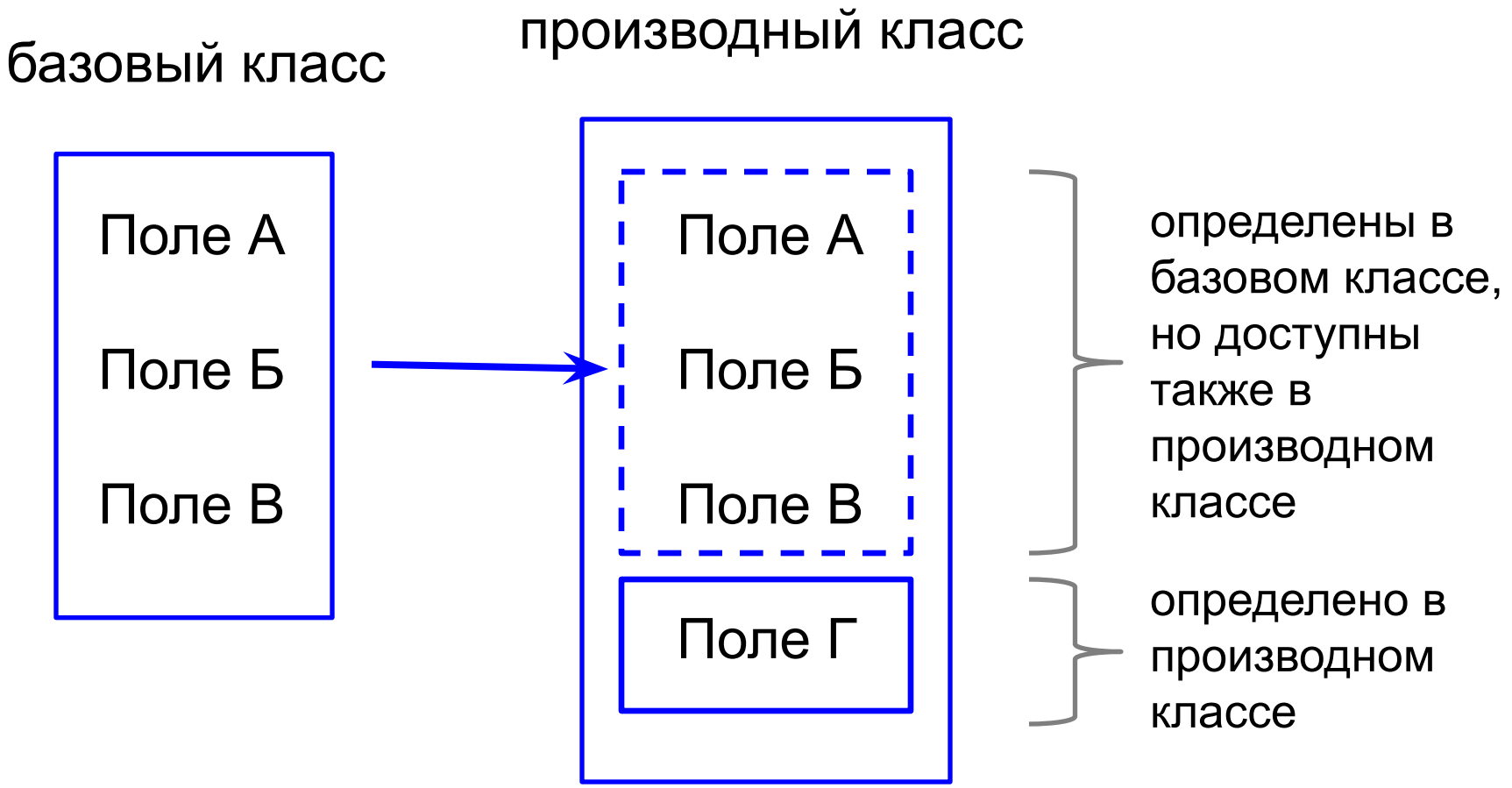
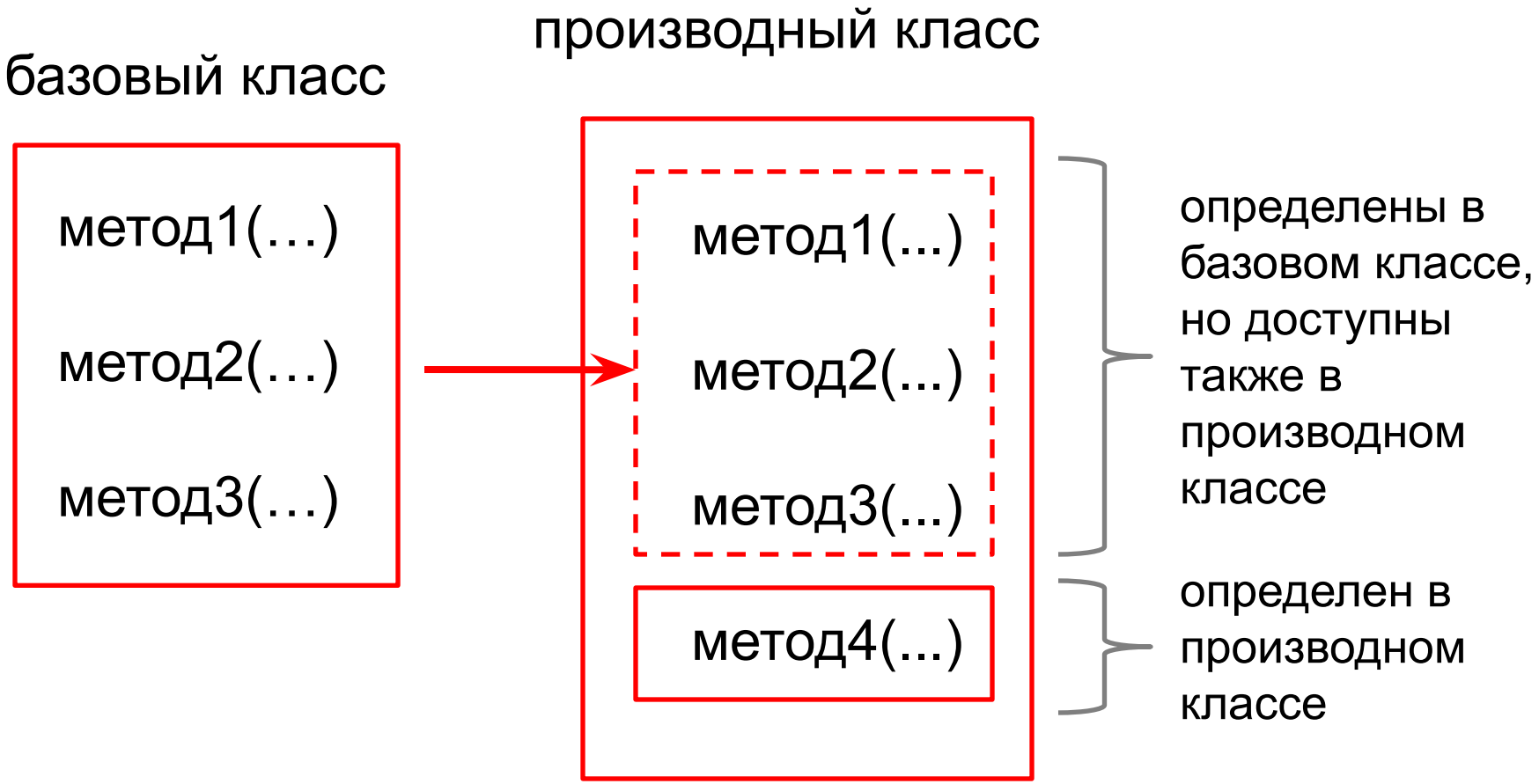
Лекция 8

Общее и частное наследование. Права доступа

Наследование свойств



Наследование поведения



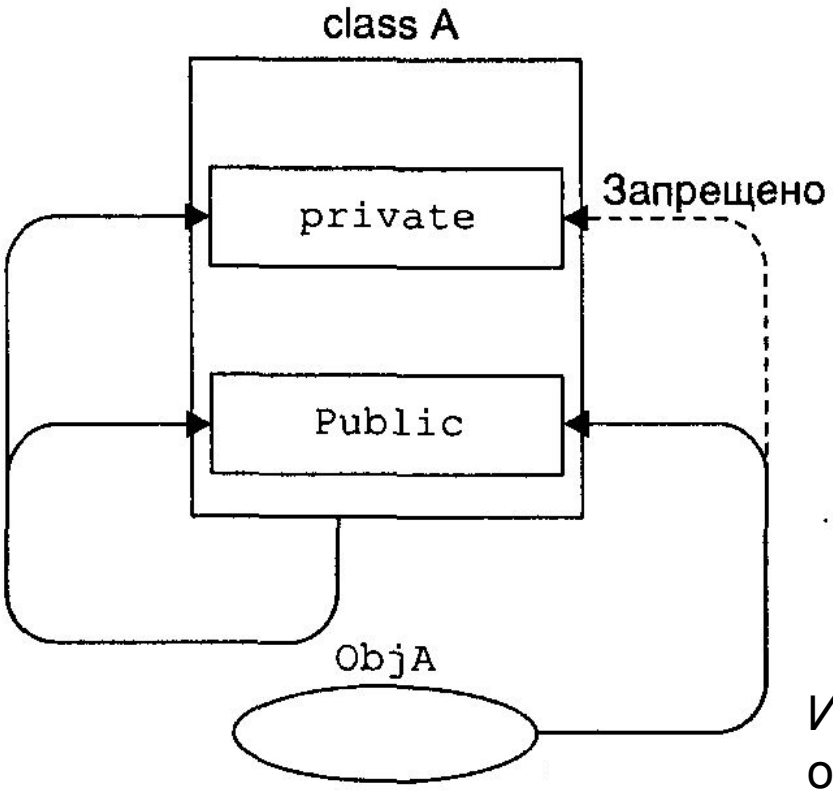
Права доступа при наследовании

Общее правило:

Методы *производного класса* имеют доступ к полям и методам *базового класса* только в случае, если в базовом классе они были объявлены как **public** или **protected**. К закрытым (**private**) данным наследник доступа не имеет.

Ситуация без наследования

Методы класса A имеют доступ ко всем полям и методам (public и private)



Извне можно обращаться только public полям и методам

Ситуация с общим (public) наследованием

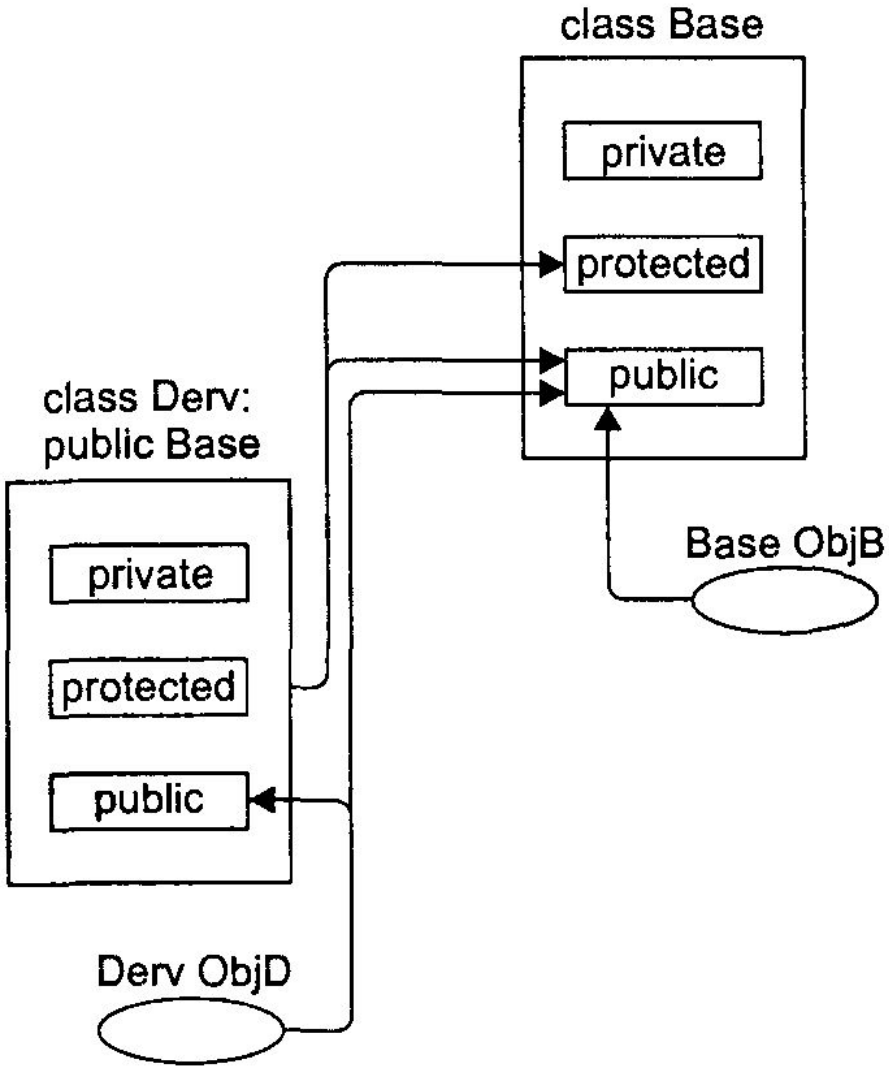


Таблица прав доступа при public-наследовании

спецификатор доступа	доступ из базового класса	доступ из производных классов	доступ из внешних функций
public	+	+	+
protected	+	+	-
private	+	-	-

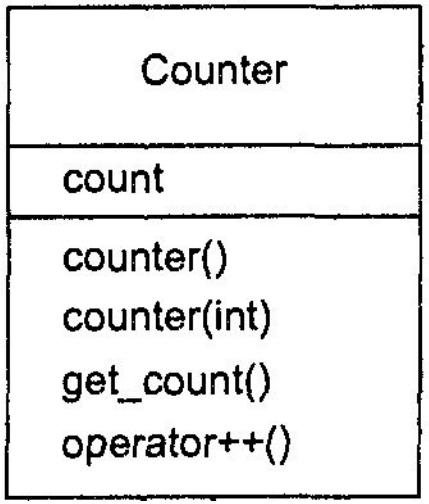
Пример общего наследования

Класс счетчика (Counter)

Использование: подсчет числа определенных событий (например, числа скачиваний файла)

Поля:
count – текущее число событий

Методы:
counter(), counter(int) – конструкторы,
operator++() – увеличение счетчика,
get_count() – запрос значения.



Базовый класс счетчика

```
class Counter
  {
  protected:
     unsigned int count;

  public:
     Counter() : count(0)
        {  }
     Counter(int c) : count(c)
        {  }
     unsigned int get_count() const
        { return count; }
     Counter operator++()
        { return Counter(++count); }
  };
```

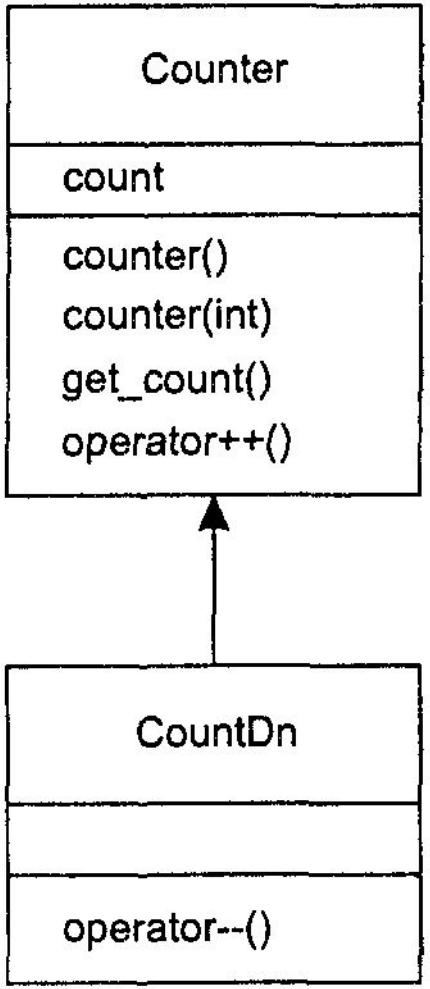
Использование базового класса

```
int main()
{
   Counter c1;   // объект класса Counter

   cout << "\nc1=" << c1.get_count();
   ++c1; ++c1; ++c1;
   cout << "\nc1=" << c1.get_count();

   cout << endl;
   return 0;
}
```

Производный класс – счетчик с уменьшением

Класс CountDn : наследник Counter

Методы:
operator--() – уменьшение счетчика.



Производный класс

```
class CountDn : public Counter
{
  public:
     Counter operator--()
     {
         return Counter(--count);
     }
};
```

Использование производного класса

```
int main()
{
   CountDn c2;

   cout << "\nc2=" << c2.get_count();
   ++c2; ++c2; ++c2;
   cout << "\nc2=" << c2.get_count();

   --c2; --c2;
   cout << "\nc2=" << c2.get_count();

   return 0;
}
```

Конструктор производного класса

```
class CountDn : public Counter
{
  public:
     CountDn() : Counter()
        { }

     CountDn(int c) : Counter(c)
        { }

     Counter operator--()
        { return Counter(--count); }
};
```

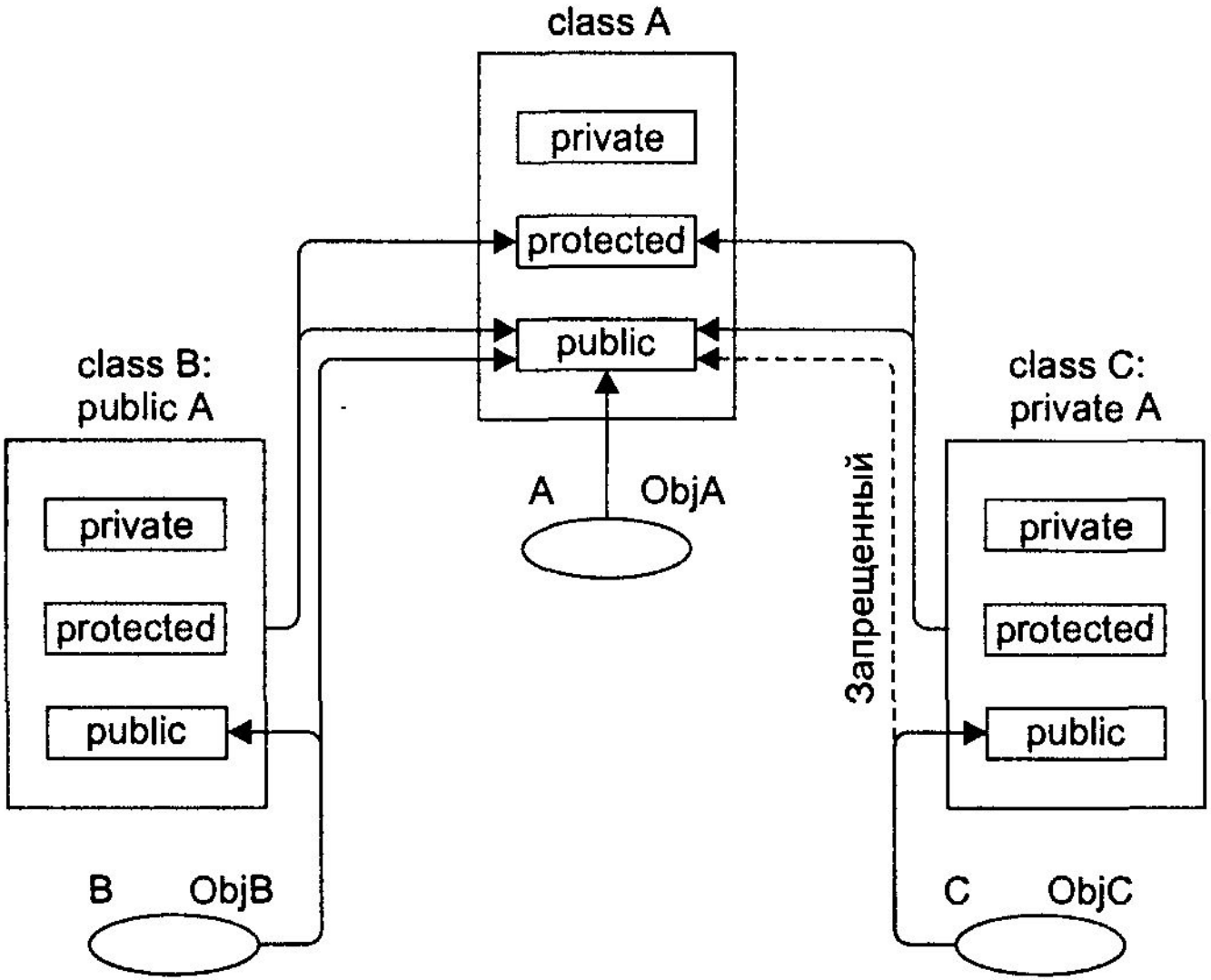
Общее и частное наследование

Альтернативой спецификатору доступа public при наследовании является спецификатор private.

```
class имя_класса1 : private имя_класса2
{
    <объявления полей и методов класса1>
}
```

При частном (private) наследовании доступ ко всем членам базового класса для объектов производного класса закрыт.

Права доступа

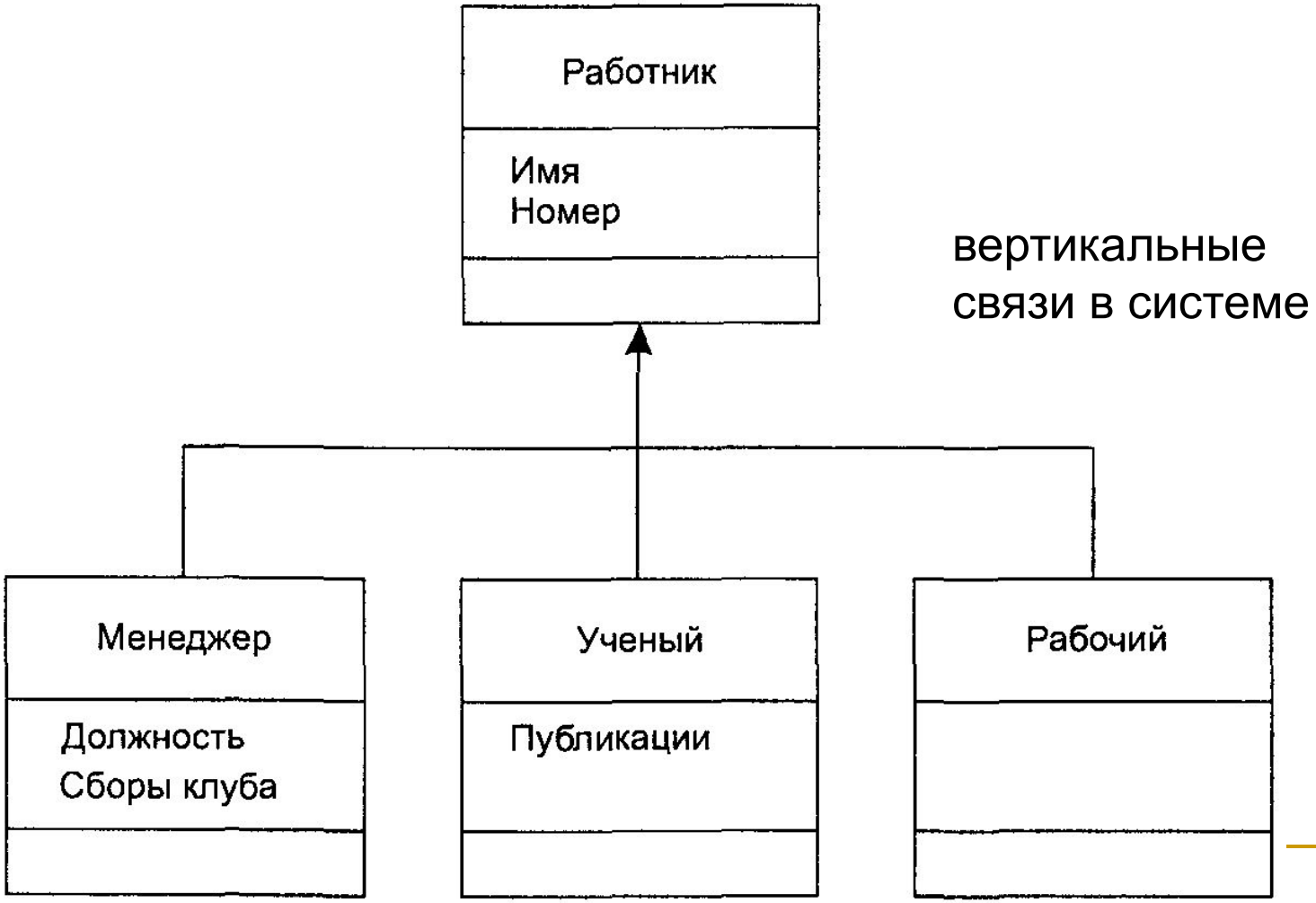


Пример с общим и частным наследованием

```
void main()
{
  int a;

  B objB;
  a = objB.privdataA;    //ошибка!
  a = objB.protdataA;    //ошибка!
  a = objB.publdataA;    //OK

  C objC;
  a = objC.privdataA;    // ошибка!
  a = objC.protdataA;    // ошибка!
  a = objC.publdataA;    // ошибка!
}
```

Иерархии классов

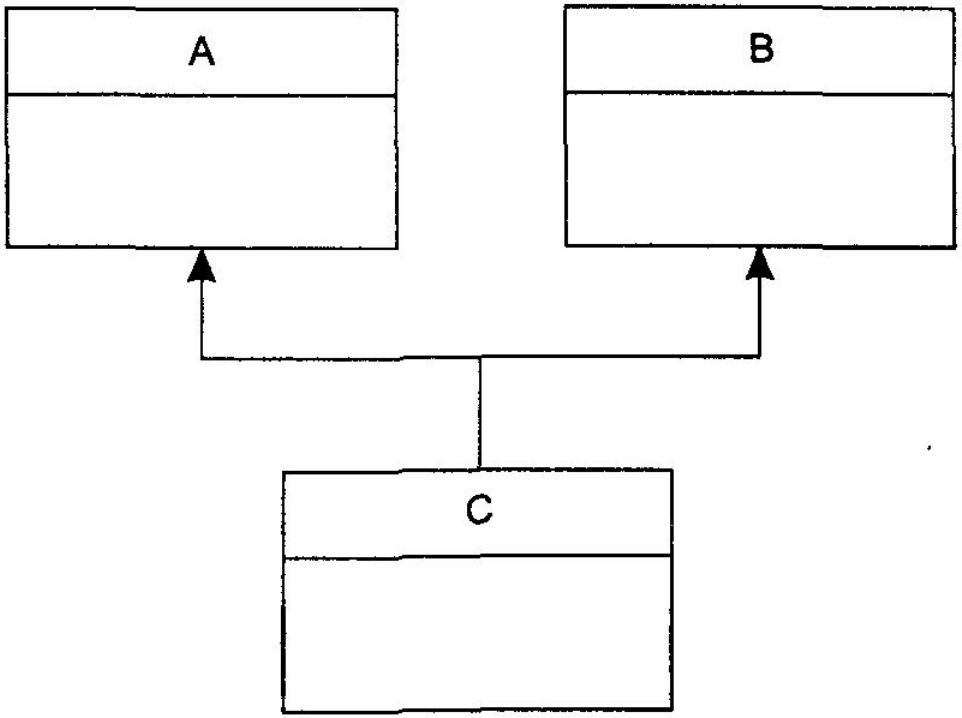
Пример: база данных сотрудников предприятия



вертикальные
связи в системе

Множественное наследование

Класс может быть производным от нескольких базовых классов. Такой случай называют **множественным наследованием**.



Синтаксис множественного наследования

```
class A
{
    <поля и методы класса A>
};

class B
{
    <поля и методы класса B>
};

class C : public A, public B
{
    <поля и методы класса C>
};
```

Пример: база данных сотрудников