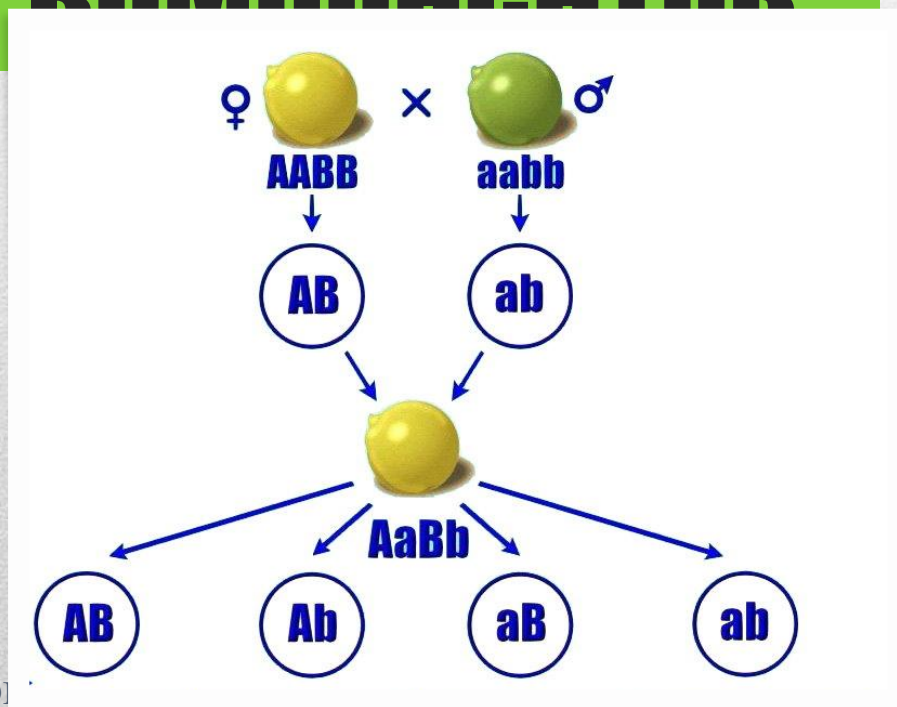


Понятие концепции наследования.

Видимость компонентов



Дисциплина: Конструирование про...

Преподаватель: Гайшун Алеся Александровна ©

План лекции:

1. Введение
2. Базовые и производные классы
3. Открытое, защищенное и закрытое наследование

Цель изучения темы:

Сформировать знания учащихся о базовой концепции ООП – **наследования**, механизме наследования классов на конкретном примере и видимости компонентов в зависимости от степени их защиты.



**существующий
класс**

*наследует
элементы*

новый класс

~~писать новые
поля данных
и методы~~

Базовый класс – это класс, на основе которого создаются другие классы.

Производный класс – это класс, который наследует базовый класс.

Производный класс служит представлением более специализированной группы объектов.



Непосредственный базовый класс является классом, который явно наследуется производным классом.

Косвенный базовый класс расположен на два или более уровней выше в иерархии наследования.

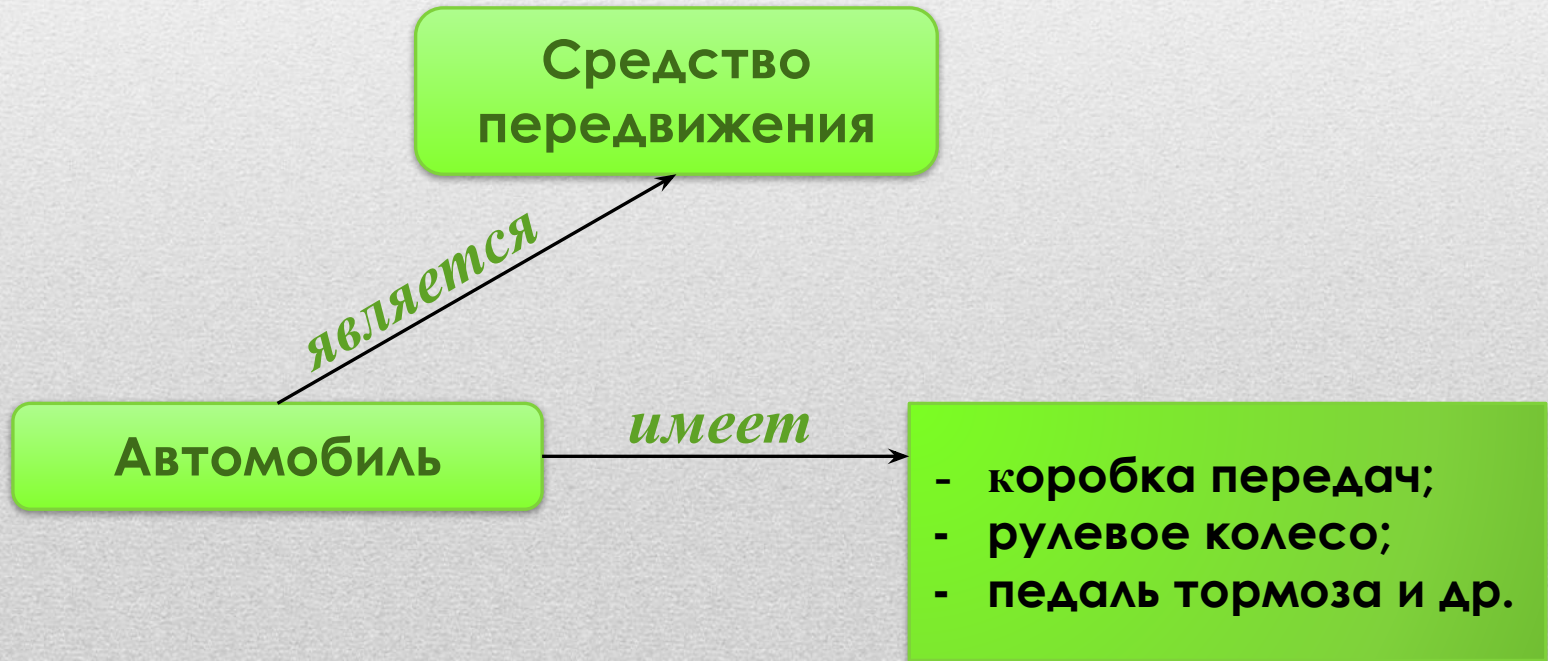
В C++ различают отношение

«является»

представляет концепцию
наследования

«имеет»

представляет
КОМПОЗИЦИЮ



```
class A
{
  public:
    int a;
    void f()
};
```

```
class B
{
  ...
};
```

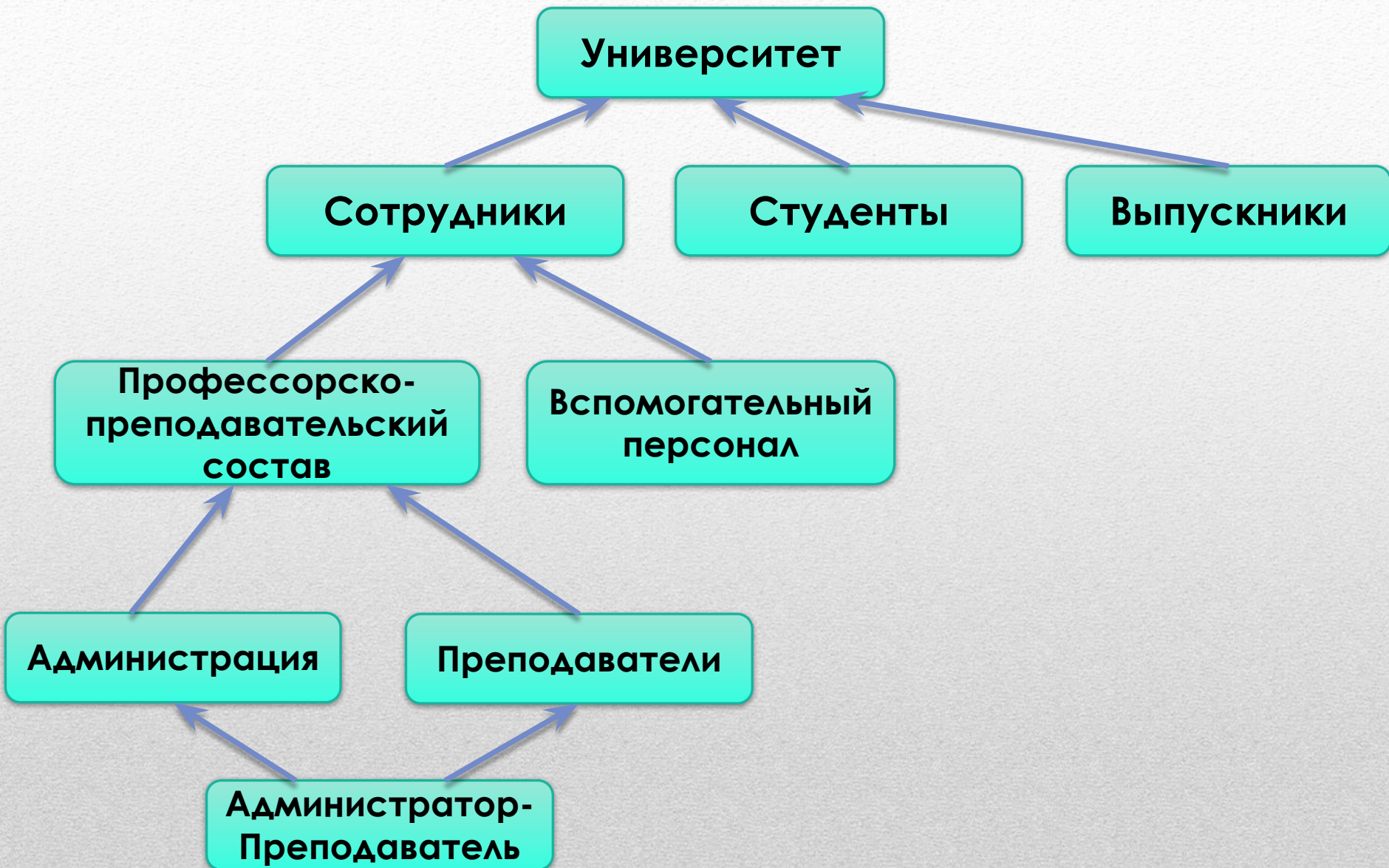
Производный класс может
обращаться к элементам
базового класса.

```
class A
{
  private:
    int a;
    void f()
};
```

```
class B
{
  ...
};
```

Элементы базового класса не
доступны элемент-функциям
производного.

Иерархия наследования для членов университетского сообщества



```
class Employee : public University
```

Спецификатор доступа элемента базового класса	Тип наследования		
	public	protected	private
public	<p>public в производном классе. Непосредственно доступен для элемент-функций, дружественных и обычных функций.</p>	<p>protected в производном классе. Непосредственно доступен для элемент-функций и дружественных функций.</p>	<p>private в производном классе. Непосредственно доступен для элемент-функций и дружественных функций.</p>
protected	<p>protected в производном классе. Непосредственно доступен для элемент-функций и дружественных функций</p>	<p>protected в производном классе. Непосредственно доступен для элемент-функций и дружественных функций</p>	<p>private в производном классе. Непосредственно доступен для элемент-функций и дружественных функций</p>
private	<p>Скрыт в производном классе. Может быть доступен для элемент-функций и дружественных функций через открытые или защищенные элемент-функции базового класса.</p>	<p>Скрыт в производном классе. Может быть доступен для элемент-функций и дружественных функций через открытые или защищенные элемент-функции базового класса.</p>	<p>Скрыт в производном классе. Может быть доступен для элемент-функций и дружественных функций через открытые или защищенные элемент-функции базового класса.</p>



Свойства

Имя

⋮

Возраст

Сила

Методы:

Строить

Кушать

Спать

Свойства:

Имя

Возраст

Сила

Методы:

Готовить

Кушать

Спать

```
#include <iostream>
using namespace std;

class Human
{
public:
    string name;
    int age;
    int force;

    void eat()
    { force+=20;
      cout<<force<<" - Now I have much more energy!"<<endl;}

    void sleep()
    { force+=15;}

    void information()
    {cout<< "My name is "<<name<<endl;
      cout<<"I am " <<age<< " years old."<<endl;}
};
```

```
class Builder : public Human
{
public:
    void build()
    {
        force-=10; }
};
```

```
class Grandmother : public Human
{
public:
    void cook()
    {
        cout<< "Hot cakes!"<<endl;}
};
```

```
int main()
```

```
{
```

```
Builder object_1;
```

```
object_1.name="Deni";
```

```
object_1.age=38;
```

```
object_1.information();
```

```
Grandmother object_2;
```

```
object_2.name="Mila";
```

```
object_2.age=68;
```

```
object_2.force=0;
```

```
object
```

```
object
```

```
return
```

```
}
```

```
My name is Deni
```

```
I am 38 years old.
```

```
My name is Mila
```

```
I am 68 years old.
```

```
20 - Now I have much more energy!
```