Объектно-ориентированное программирование

Наследование

Гришмановский Павел Валерьевич,
кафедра автоматики и компьютерных систем, Политехнический институт, СурГУ

Описание класса

Описание класса:

```
class <имя_кл> [: <список_кл_предков>]
{
   <описания>
}
[<список_экземпляров>];
```

Все классы-предки являются равноправными
Потомок наследует **все** элементы от **всех** классов-предков

Список классов-предков:

```
<мод_доступа> <имя_предка1>[, <мод_доступа> <имя_предка2> ...]
```

Один предок – нормальное наследование
Два и более предков – множественное наследование
Нет предков – нет наследования
Количество потомков не имеет значения

Виды наследования

Доступность в классе-предке (исходная)	Доступность унаследованных элементов в классе-потомке в зависимости от вида наследования (результирующая)		
	Закрытое наследование **private**	Защищенное наследование **protected**	Открытое наследование **public**
private:	–	–	–
protected:	private	protected	protected
public:	private	protected	public

Модификаторы доступа только **ограничивают** видимость унаследованных элементов в классе-потомке, но доступ **нельзя расширить** (сделать более открытыми)

Чаще всего используется открытое наследование – доступ к унаследованным элементам не изменяется

Конструкторы и деструкторы

Порядок выполнения конструктора:

1. Вызываются конструкторы для всех классов-предков в порядке их перечисления в объявлении класса
2. Вызываются конструкторы для всех собственных полей данных в порядке их описания в объявлении класса
3. Выполняется тело конструктора

- Порядок выполнения конструктора каждого поля и класса-предка – такой же
- Порядок вызова конструкторов полей и классов-предков не зависит от порядка, в котором они указаны в списке инициализации
- Если в списке инициализации не указан конструктор поля или класса-предка, то вызывается конструктор по умолчанию
- Порядок выполнения деструктора – в точности обратный

Реализация конструктора

```
[void] <имя_кл>::<имя_кл>(<список_форм_парам>)
     [: <имя>(<список_факт_парам_конс>), ...]
{
}
```

<имя> – имя поля данных или класса-предка

В списке инициализации важен вид конструкторов, но не их порядок

Класс как область действия

Приоритеты пространств:

1. Локальная область
2. Пространство класса
3. Пространства классов-предков (рекурсивно до корневого класса)
4. Пространство имен, в котором описан класс
5. Глобальное пространство

Обращение к данным и методам объекта возможно только посредством экземпляра

неявно по указателю **`this`**

Порядок поиска идентификатора при обращении к нему:

- в теле метода – п. 1-5
- в области класса – п. 2-5
- извне класса – п. 2-3

явно:

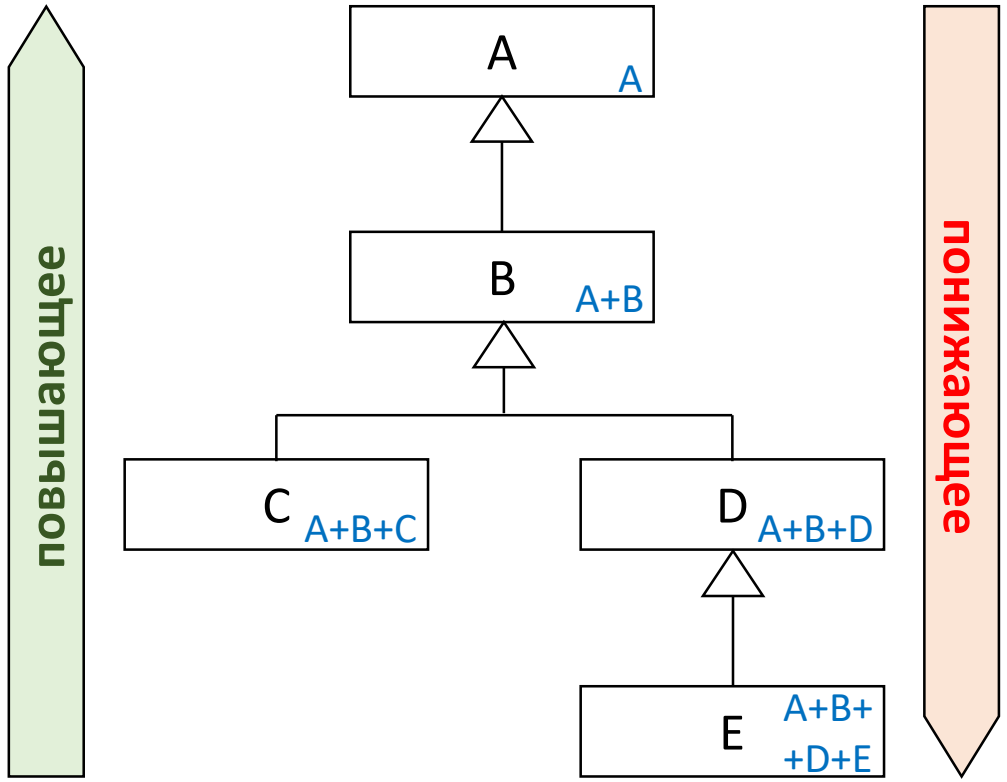
- по имени – тип точно известен
- по указателю – в соответствии с типом указателя
- по ссылке – в соответствии с типом ссылки

Любой идентификатор в классе-потомке **скрывает** все одноименные унаследованные элементы

Разрешение конфликтов:

- Операция видимости – все пространства, исключая локальное
- Специальный указатель **`this`** – только пространство класса, включая классы-предки

Повышающее и понижающее преобразования



Повышающим преобразованием называют приведение типа указателя или ссылки на объект **от потомка к предку** (вверх)

- является безопасным, выполняется автоматически (неявно)

Понижающим преобразованием называют приведение типа указателя или ссылки на объект **от предка к потомку** (вниз)

- является потенциально опасным, автоматически не выполняется (только явно)

При повышающем и понижающем преобразованиях изменяется только трактовка указателя или ссылки, но над объектом никаких действий не выполняется