ОБЪЕКТНО-ОРИЕНТИРОВАННОЕ ПРОГРАММИРОВАНИЕ

Вопросы:

1. Основные понятия объектно-ориентированного программирования
2. Определение объекта. Свойства объектов
2.1. Инкапсуляция
2.2. Наследование и переопределение.
2.3. Полиморфизм
3. Виртуальные методы. Конструкторы и Деструкторы

1. Основные понятия ООП

ОБЪЕКТНО-ОРИЕНТИРОВАННЫЙ ПОДХОД:

- Всё является объектом.
- Программа — группа объектов, указывающих друг другу, что делать, посредством сообщений.
- Каждый объект имеет собственную «память», состоящую из других объектов.
- У каждого объекта есть тип (класс).
- Все объекты определенного типа (класса) могут получать одинаковые сообщения.

В основе ООП лежит понятие объекта (object).

Объект - это тип, который включает не только поля данных объекта, но и подпрограммы для их обра ботки, называемые методами.

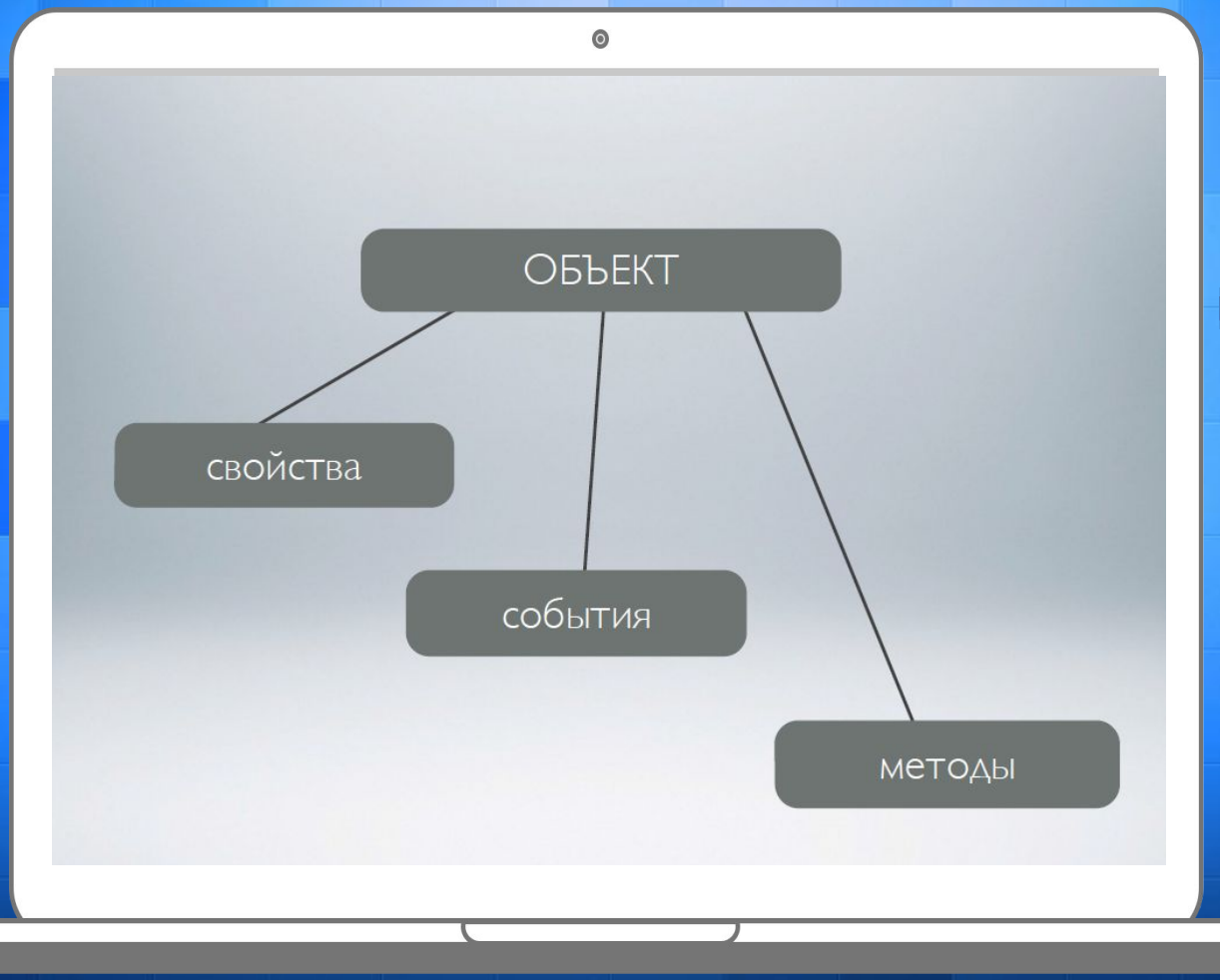
Таким образом, в объекте сосредоточены его свойства (состояния, данные) и их поведение (обработка с помощью методов). Идеи создания нового типа (объект) были заложены при введении процедурных типов параметров.

***Объект, object*.**

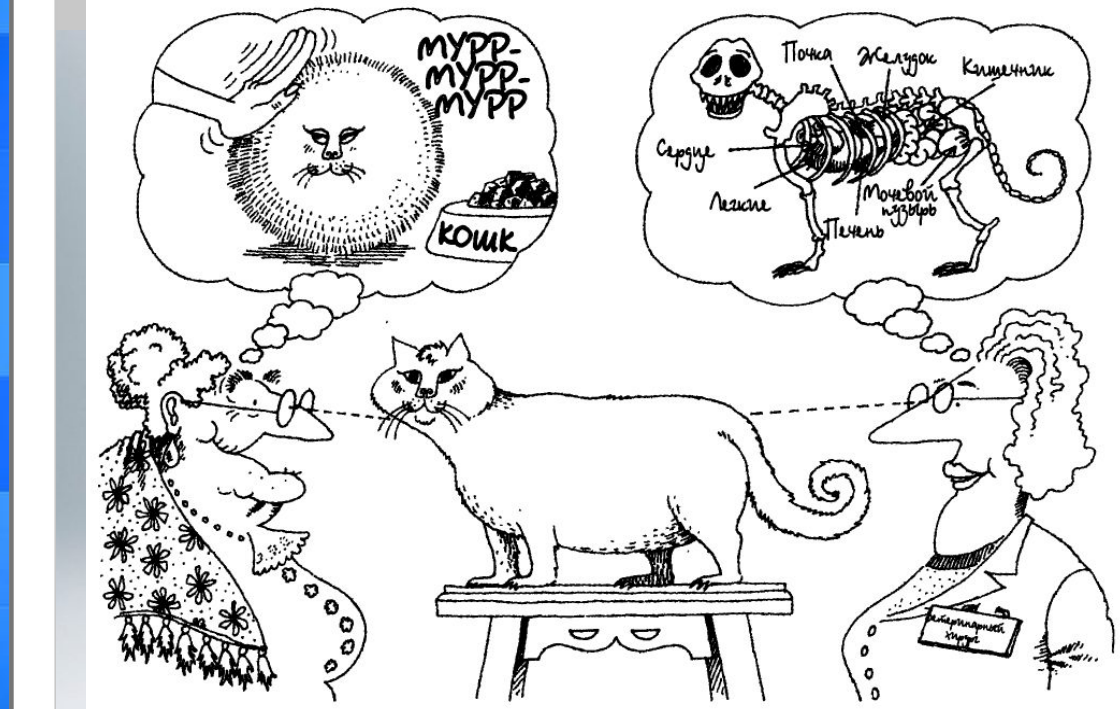
Нечто, чем можно оперировать. Объект имеет состояние, поведение и идентичность. Структура и поведение сходных объектов определены в общем для них классе. Термины "экземпляр" и "объект" взаимозаменяемы.

Состояние, state.
включает в себя перечень (обычно, статический) свойств объекта и текущие значения (обычно, динамические) этих свойств.
Событие, event.
Что-то, что может изменить состояние системы.
Сообщение, message.
Операция, которую один объект может выполнять над другим. Термины "сообщение", "метод" и "операция" обычно взаимозаменяемы.
Метод, method.
Операции, выполняемые над данным объектом, и входят в описание класса объекта
Операция, operation.
Определенное воздействие одного объекта на другой с целью вызвать соответствующую реакцию.

Структура объекта

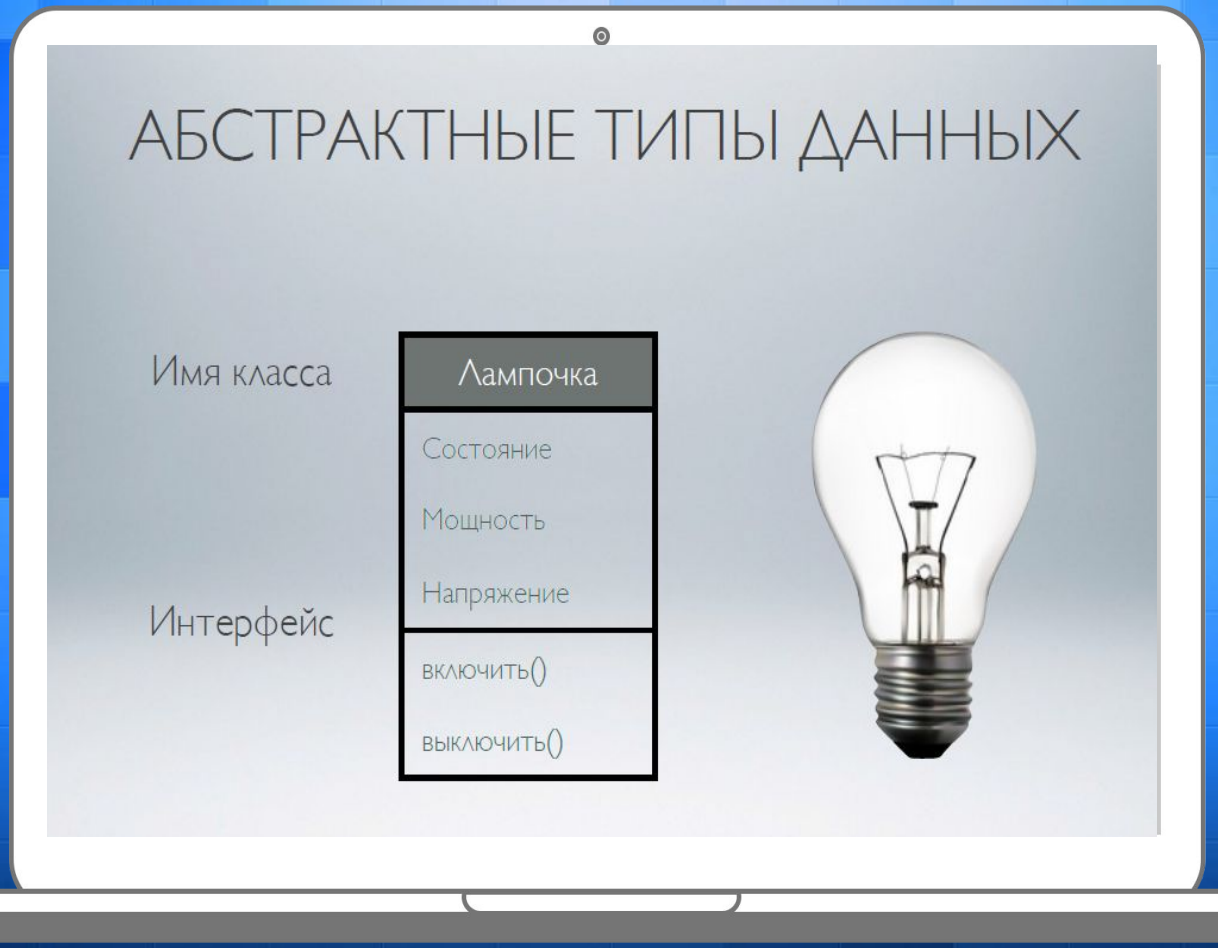


Структура класса



Абстракция концентрирует внимание на существенных свойствах объекта с точки зрения наблюдателя

UML диаграмма



ООП характеризуется тремя основными свойствами: инкапсуляцией (encapsulation), наследованием (inheritance) и полиморфизмом (polymorphism).

Инкапсуляция означает объединение в одном объекте данных и действий над ними.

Наследование - это возможность использования уже определенных объектов, что позволяет создавать иерархию объектов начиная с некоторого простого первоначального (предка) и кончая более сложными, включающими (наследующими) свойства предшествующих элементов иерархии (предков).

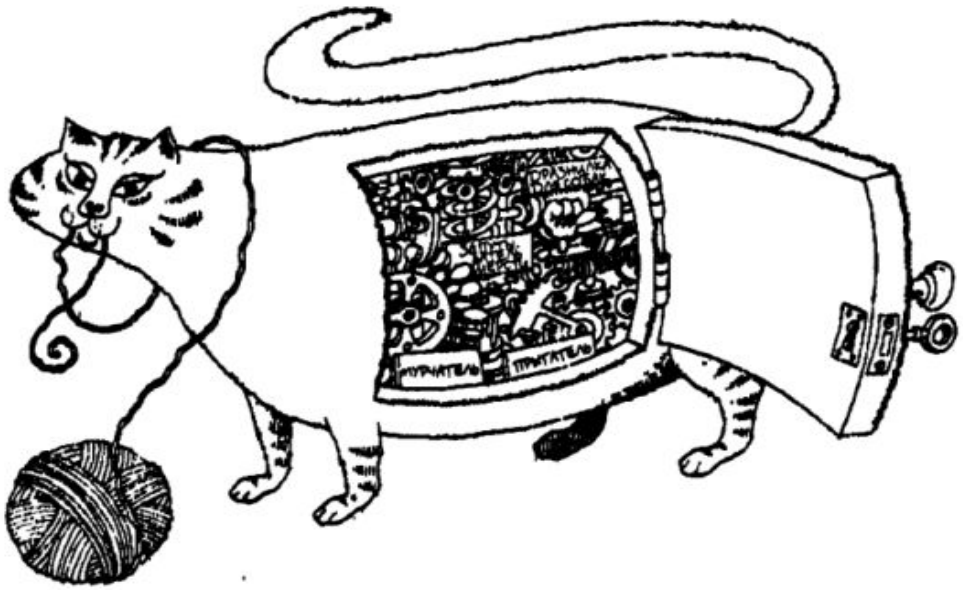
Полиморфизм - это возможность определения единого по имени действия (процедуры или функции), применимого ко всем объектам иерархии наследования; причем каждый объект иерархии может иметь особенность реализации этого действия

Особенности и применимость объектно-ориентированного программирования

1) модульность; она позволяет: -разбить программу на модули и локализовать область действия подпрограмм и переменных; - изменять локальные подпрограммы, не изменяя других программных модулей;
2) абстракция данных; абстрактный тип данных определяется на основе некоторого их представления и множества подпрограмм для обработки данных абстрактного типа;
3) динамическая связка подпрограмм программы; это позволяет не перекомпилировать всю программу при внесении изменений в отдельные модули, что увеличивает гибкость языка, позволяя вводить новые классы объектов без модификации всей программы;
4) наследование; оно позволяет создавать классы объектов, на которые может ссылаться порожденный класс, который наследует все свойства порождающего класса и может задавать дополнительные свойства и новые подпрограммы обработки данных порожденного класса.

Инкапсуляция

Основным понятием ООП и элементом программ является объект. Любая объектно-ориентированная программа состоит из двух частей:
декларативной части программы (описания объектов);
исполняемой части программы (основной программы и подпрограмм).
Исполняемая часть объектно-ориентированной программы состоит из последовательности действий, выполняемых над данными типа объект. Каждое такое действие изменяет состояние объекта и представляет собой вызов под программы, доступной объекту.
Объединение декларативных (данных) и исполняемых (процедурных) элементов при описании объекта называется инкапсуляцией.



Инкапсуляция скрывает детали реализации объекта

Объект - это такая структура, компонентами которой являются данные и методы (процедуры и функции) для их обработки. Компоненты - данные -это поля объекта, а компоненты подпрограммы - это методы. По написанию объект напоминает тип-запись. Форма объявления объекта:

TYPE Имя-объекта = OBJECT
Поля-данных;
Заголовки-методов;
END;

При описании объекта сначала определяются все данные объекта, а затем -заголовки методов их обработки - как опережающие описания подпрограмм. Далее - тексты всех методов, написанные так же, как подпрограммы в модулях.

Текст подпрограммы метода может быть в одном из программных файлов: в том же, где объявлен объект, или в другом, например в одном из модулей. При написании текста метода в заголовке подпрограммы перед ее именем обязательно надо указать имя типа объекта, которому принадлежит метод, т. е. имя метода должно быть составным в виде:

имя-типа-объекта.имя-подпрограммы;.

Конкретная переменная или константа объектного типа называется экземпляром - переменною или константою этого типа.

Вызов метода для обработки данных экземпляра состоит из составного имени в виде:

имя-экземпляра-объектного-типа.имя-метода;

Скрытые поля и методы

Часть полей и методов объектных типов можно объявить как скрытые. Это ограничивает область их видимости. Для этого используется ключевое слово **private**. Схема объявления:

```
Type
ObjectType = object
Обычные поля и методы
private
Скрытые поля и методы
end;
```

Идентификаторы полей и методов, объявленных как скрытые, известны (доступны, видимы) только в пределах программы или модуля, в которых они объявлены. Вне модуля с их описанием скрытые поля и методы неизвестны (недоступны). Сам объектный тип и его остальные компоненты видимы по обычным

Наследование и переопределение.

ООП позволяет определить новый объект как *потомок (наследник)* другого, ранее определенного типа.

Это означает, что новый тип автоматически получает все поля и методы ранее определенного типа, который в этом случае называется предком или родителем. В объявлении типа-потомка (наследника) должно быть указано в круглых скобках после *служебного слова* **object** имя родительского типа. Поля и методы предка могут появляться в телах методов наследников так, как если бы они были явно объявлены в объектах-наследниках. Это существенно упрощает запись схожих объектов.

Форма объявления объекта-потомка (наследника):

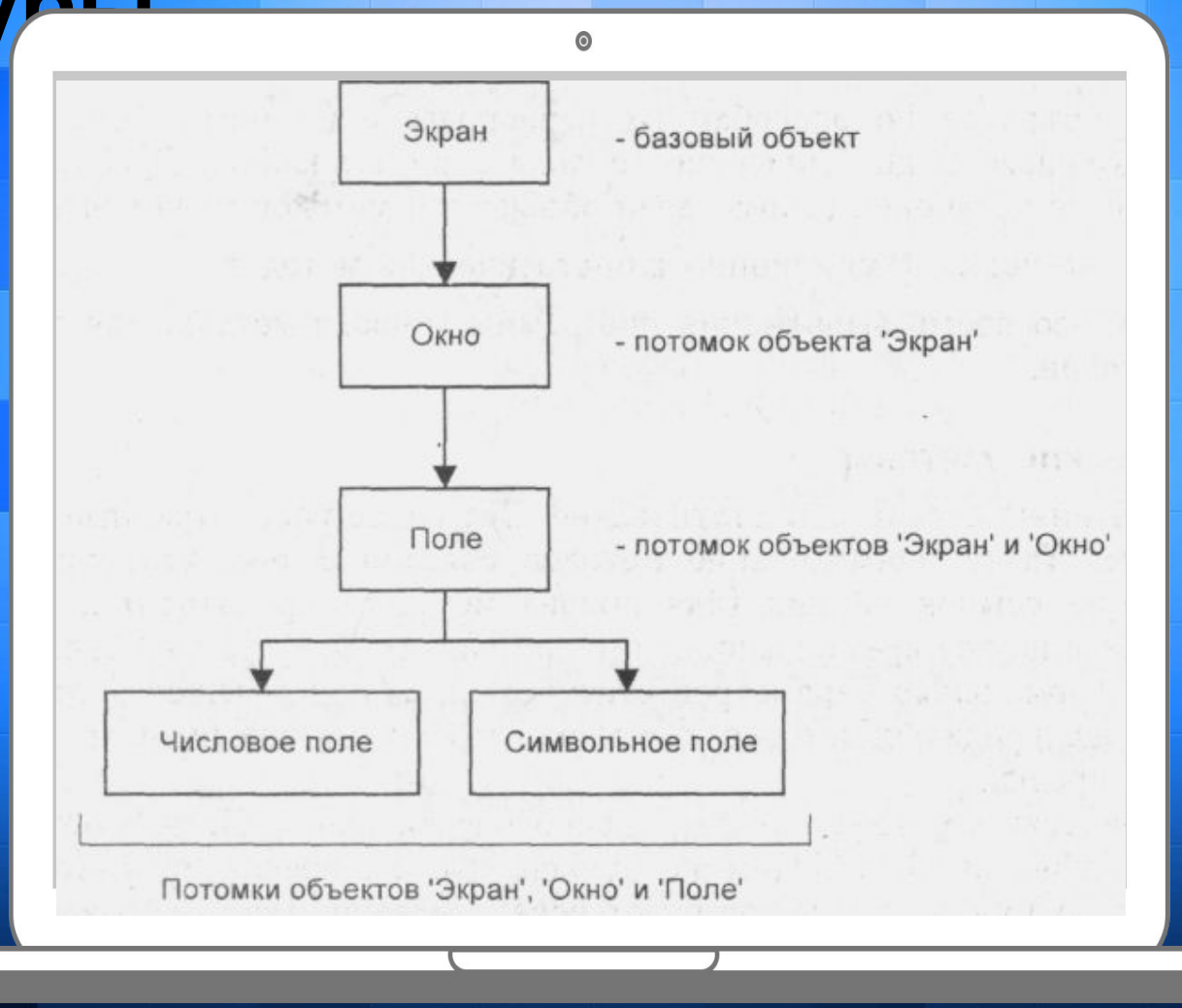
TYPE
Имя-типа-объекта-потомка =OBJECT(Имя-типа-объекта-предка)
Новые-поля-объекта-потомка;
Новые-методы-объекта-потомка;
END;

Предок у типа может быть только один. Он может быть потомком другого типа и т. д. Потомков у одного предка может быть много. Потомок наследует поля и методы всех своих предков.

Процесс наследования является транзитивным: если тип объекта TypeB -наследник типа TypeA, а тип TypeC - наследник типа TypeB, то тип объекта TypeC также является наследником TypeA:

Type A → Type B → Type C

Пример простой иерархической структуры



Полиморфизм

Полиморфизм - это возможность иметь несколько методов с одним и тем же именем для различных объектов одной иерархии, т. е. средство для развития объектов в потомках.
Свойство реализуется тем, что объект-потомок может переопределять, т. е. заменять методы предка на новые с теми же именами. Какой из методов будет выполняться при. обращении к методу с заданным именем, определяется типом объекта (предок или потомок) и используемого метода.
Методы объектов по способам их переопределения могут быть статические, виртуальные и динамические (подкласс виртуальных).
В соответствии с этим процесс установки взаимосвязи объектов и методов может быть:
ранним - во время компиляции - для статических методов;
поздним - во время выполнения программы (вызова метода) для виртуальных методов.

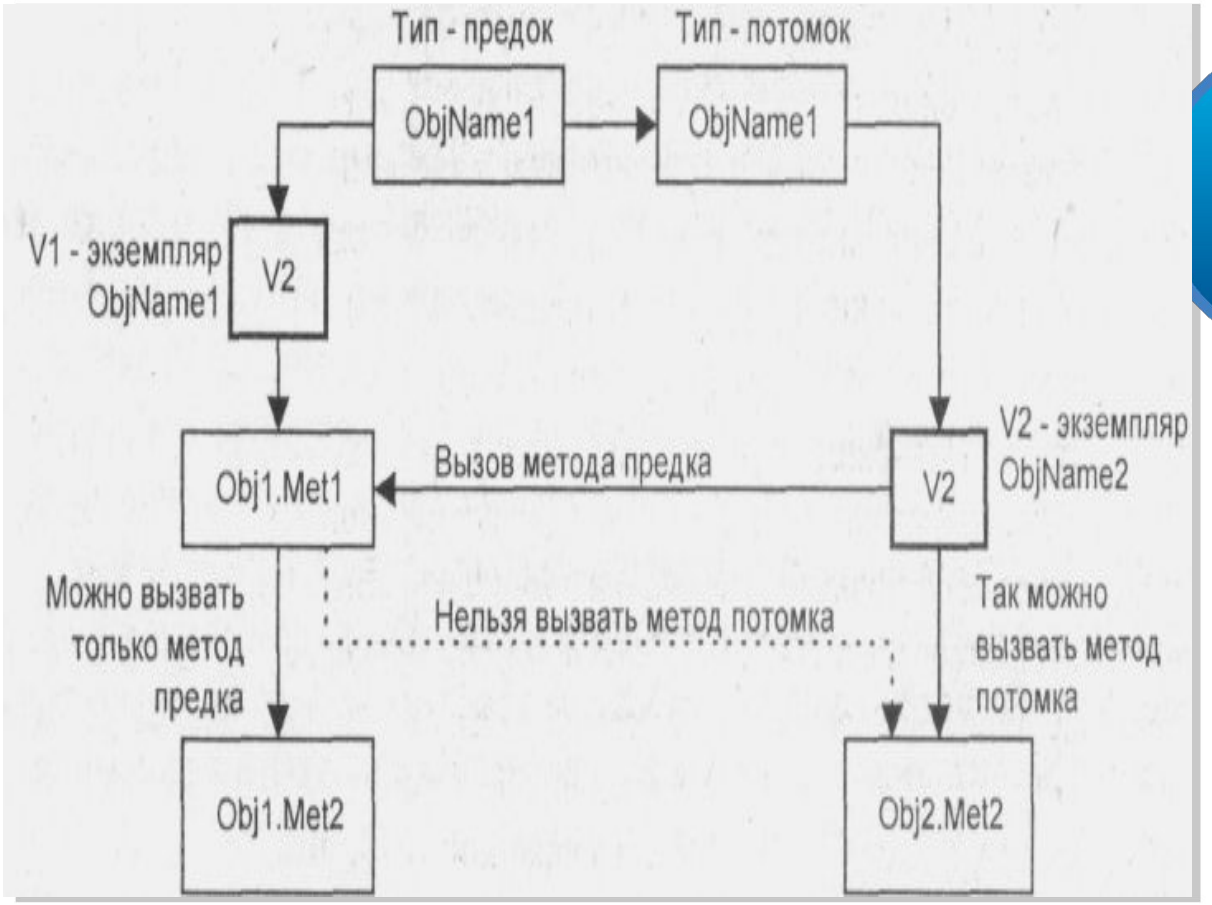
Статические методы

По умолчанию все методы статические. Все ранее рассмотренные методы -статические. Такое наименование методов связано с тем, что определение и размещение ссылок на них (для вызова методов) производится на этапе компиляции и на все время выполнения программы. Это - раннее связывание.

Список формальных параметров статических методов может быть различным у метода предка и методов потомков, переопределяющих (заменяющих) этот метод предка.

Со статическим методом связан способ наследования. Если в объявлении потомка появляется метод с тем же именем, что и у предка, то в этом типе и во всех его потомках этот метод будет переопределен. Таким образом, потомки могут использовать переопределенные ими методы предка. Предок может вызвать только свои методы; методы потомков для него недоступны. Таким образом, при использовании статических методов полиморфизм распространяется от текущего уровня иерархии вниз, к потомкам.

Схема ограничения вызова переопределенных статических методов



Виртуальные методы. Конструкторы и Деструкторы

Виртуальный (кажущийся, гипотетический) **метод** имеет спецификатор (стандартную директиву) Virtual.

Например:

Procedure Met2; Virtual;

Виртуальный метод предназначен для переопределения виртуального метода предшествующего предка. Недопустимо смешение статических и виртуальных методов при их переопределении.

Ограничение: переопределяющие виртуальные методы должны иметь точно такой же набор формальных параметров, как и самый первый виртуальный метод многоуровневой иерархии объектов.

Паскаль обеспечивает вызов (связывание) виртуального метода программы на этапе выполнения программы. Это называют поздним связыванием.

Компилятор не устанавливает связи объекта с виртуальным методом. Вместо этого он создает специальную *таблицу виртуальных методов* (ТВМ, VMT -Virtual Method Table).
Для каждого типа объекта создается своя ТВМ; каждый экземпляр объекта использует эту ТВМ, единственную для данного типа вир туальных объектов.
В каждой ТВМ содержится размер данного типа объекта в байтах. ТВМ любого объекта доступна через скрытый параметр Self, содер жащий адрес ТВМ, который передается методу при вызове.

Связывание каждого экземпляра объекта и его ТВМ осуществляется с помощью *конструктора* на этапе выполнения программы.

Это специальный метод, подобный обычной процедуре, но в заголовке **вместо PROCEDURE стоит слово CONSTRUCTOR**.

Если объектный тип содержит виртуальный метод, то он должен содержать хотя бы один конструктор. Каждый экземпляр объекта должен инициализироваться отдельным вызовом конструктора. Конструктор инициализирует экземпляр объекта и устанавливает для него значение адреса его ТВМ. Экземпляр объекта содержит только адрес ТВМ, а не саму ТВМ.

В объекте может быть сколько угодно конструкторов.

Конструктор не может быть виртуальным. Конструктор может быть только статическим и может быть переопределен. Конструкторы наследуются так же, как и другие статические методы. Из конструктора можно вызывать и виртуальные методы.

Метод конструктора может быть и пустым, так как основная информация содержится не в теле конструктора, а связана с его заголовком, содержащим слово Constructor. Например:

```
Constructor TA.INIT ;
Begin
End;
```

Конструктору принято давать имя INIT.

Динамические методы

В Паскале имеется дополнительный класс методов позднего связывания -**динамические методы**. Они являются подклассом виртуальных методов и отличаются от них только способом вызова на этапе выполнения.
Объявление динамического метода аналогично виртуальному, за исключением того, что оно должно включать индекс (номер) динамического метода, который указывается сразу за ключевым словом **Virtual**. Индекс динамического метода должен быть целочисленной константой в диапазоне от 1 до 65535 и представлять собой уникальное значение среди индексов других динамических методов данного объектного типа и его предков. Например:

Function GetSum: Real; Virtual 10; **где - 10 - ИНДЕКС.**

При использовании динамических методов создается *таблица динамических методов* (ТДМ, DMT - Dynamic Method Table), альтернативная таблице виртуальных методов.

В ней указываются только те виртуальные методы, которые переопределяются. Это экономит ОП, но требуется время для поиска в DMT объектов-предков. Поэтому производительность DMT ниже, чем VMT, так как доступ к методу через VMT - простое извлечение адреса из таблицы, а доступ к методу через DMT может привести к более длительному поиску

Удаление объектов. Деструкторы

Удаление объектов может быть с помощью процедур Dispose или с помощью деструктора. Подобно другим динамическим типам данных динамические объекты со статическими методами могут удаляться с помощью **Dispose**.

Например:

Dispose (P1);.

вопросы

1. Что такое объектно-ориентированное программирование? Каковы его особенности и область применения?
2. Назовите свойства объектов.
3. Что такое инкапсуляция? Как объявить объект?
4. Что такое наследование и переопределение, предок, потомок? Как объявить объект-потомок?
5. Что такое полиморфизм? Как переопределить статический, виртуальный методы?
6. Как удалить объект?
7. Что такое конструктор, деструктор?