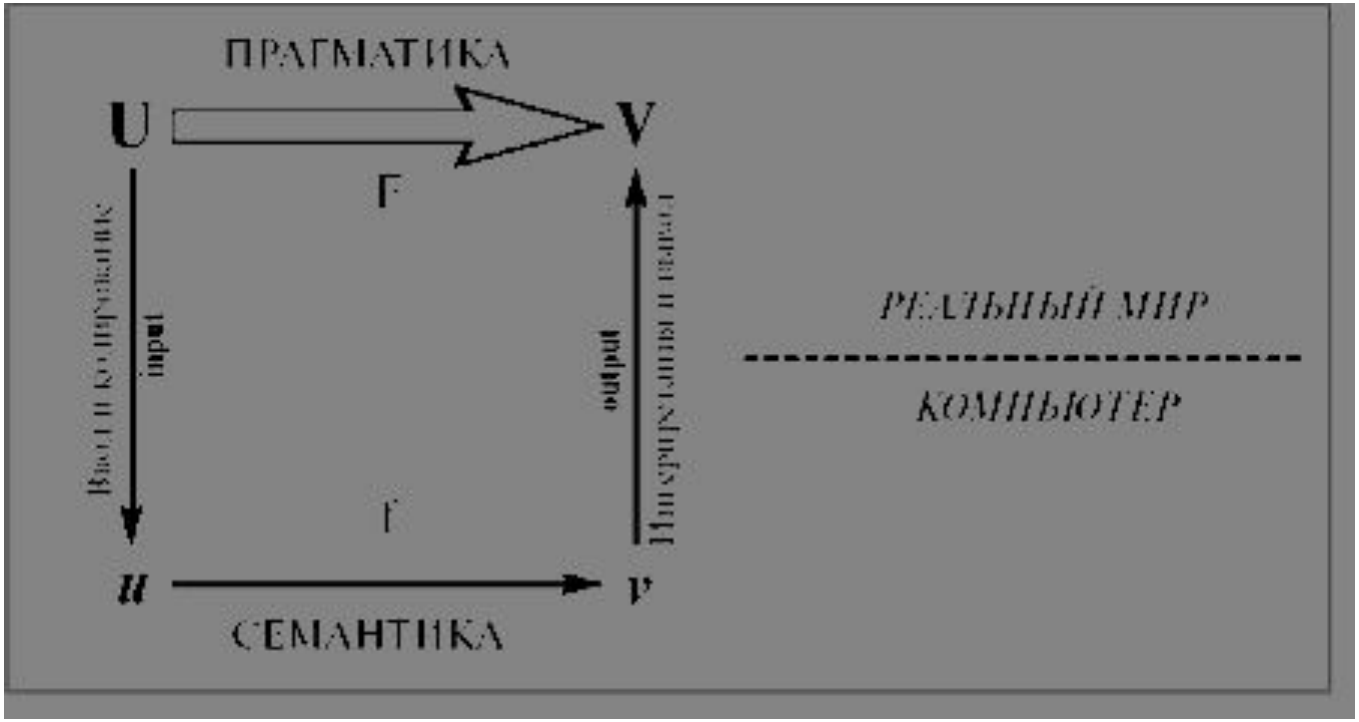
Методология объектно-ориентированного программирования

- **Объектно-ориентированный подход к проектированию основан на представлении предметной области задачи в виде множества моделей для независимой от языка разработки программной системы на основе ее прагматики.**



Семантика (смысл программы с точки зрения выполняющего ее компьютера) и прагматика (смысл программы с точки зрения пользователей)

Объектно-ориентированный подход обладает преимуществами

- уменьшение сложности программного обеспечения;
- повышение надежности программного обеспечения;
- обеспечение возможности модификации отдельных компонентов программного обеспечения без изменения остальных его компонентов;
- обеспечение возможности повторного использования отдельных компонентов программного обеспечения.

Объекты

Будем называть ***объектом*** **понятие, абстракцию или любой предмет с четко очерченными границами, имеющий смысл в контексте рассматриваемой прикладной проблемы**.

Введение *объектов* преследует две цели:

- понимание прикладной задачи (проблемы);
- введение основы для реализации на компьютере.

Объект - это мыслимая или реальная сущность, обладающая характерным поведением и отличительными характеристиками и являющаяся важной в предметной области.

Каждый объект имеет **состояние**, обладает четко определенным **поведением** и **уникальной идентичностью**.

Состояние

Состояние (state) - **совокупный результат *поведения объекта*: одно из стабильных условий, в которых *объект* может существовать, охарактеризованных количественно**; в любой момент времени состояние объекта включает в себя перечень (обычно статический) свойств объекта и текущие значения (обычно динамические) этих свойств.

Поведение

В терминологии объектно-ориентированного подхода понятия **"действие", "сообщение" и "метод" являются синонимами**. Т.е. выражения "выполнить действие над объектом", "вызвать метод объекта" и "послать сообщение объекту для выполнения какого-либо действия" эквивалентны.

Поведение (behavior) - **действия и реакции *объекта*, выраженные в терминах передачи сообщений и изменения *состояния*; видимая извне и воспроизводимая активность *объекта***.

Уникальность

Уникальность (identity) - **свойство *объекта*; то, что отличает его от других *объектов*.**

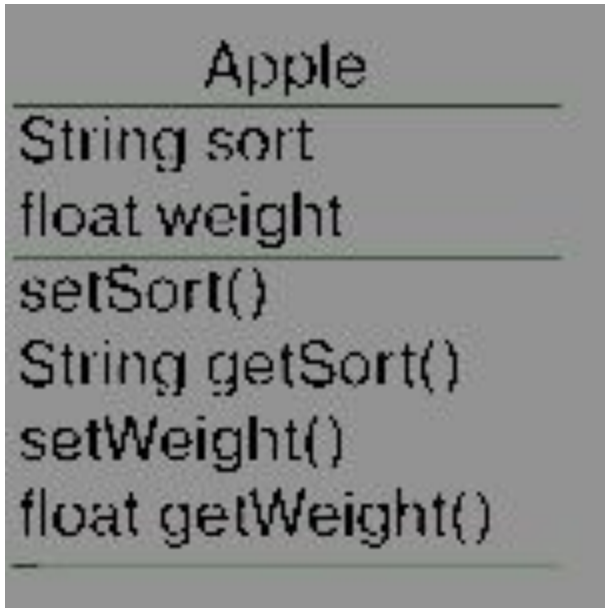
В машинном представлении под параметром уникальности объекта чаще всего понимается адрес размещения объекта в памяти.

Классы

Все объекты одного и того же класса описываются одинаковыми наборами атрибутов. Однако объединение объектов в классы определяется не наборами атрибутов, а семантикой.

Класс **- это шаблон *поведения объектов* определенного типа с заданными параметрами, определяющими *состояние***. Все экземпляры одного класса (*объекты*, порожденные от одного класса) имеют один и тот же набор свойств и общее поведение, то есть одинаково реагируют на одинаковые сообщения.

Графическое представление класса в UML



Инкапсуляция

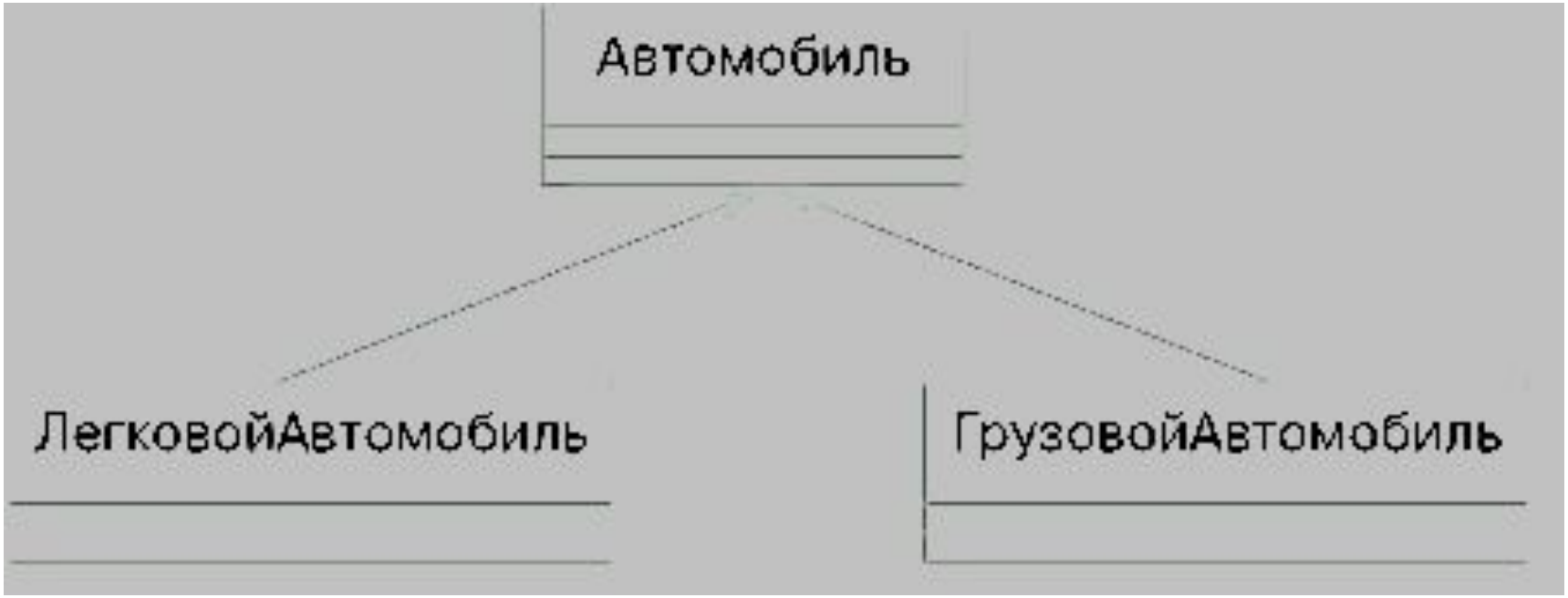
Инкапсуляция (encapsulation) - ***это сокрытие реализации*** **класса** ***и отделение его внутреннего представления от внешнего (интерфейса).***

Наследование

Наследование (inheritance) - **это отношение между классами, при котором класс использует структуру или поведение другого класса (одиночное *наследование*), или других (множественное *наследование*) *классов*.**

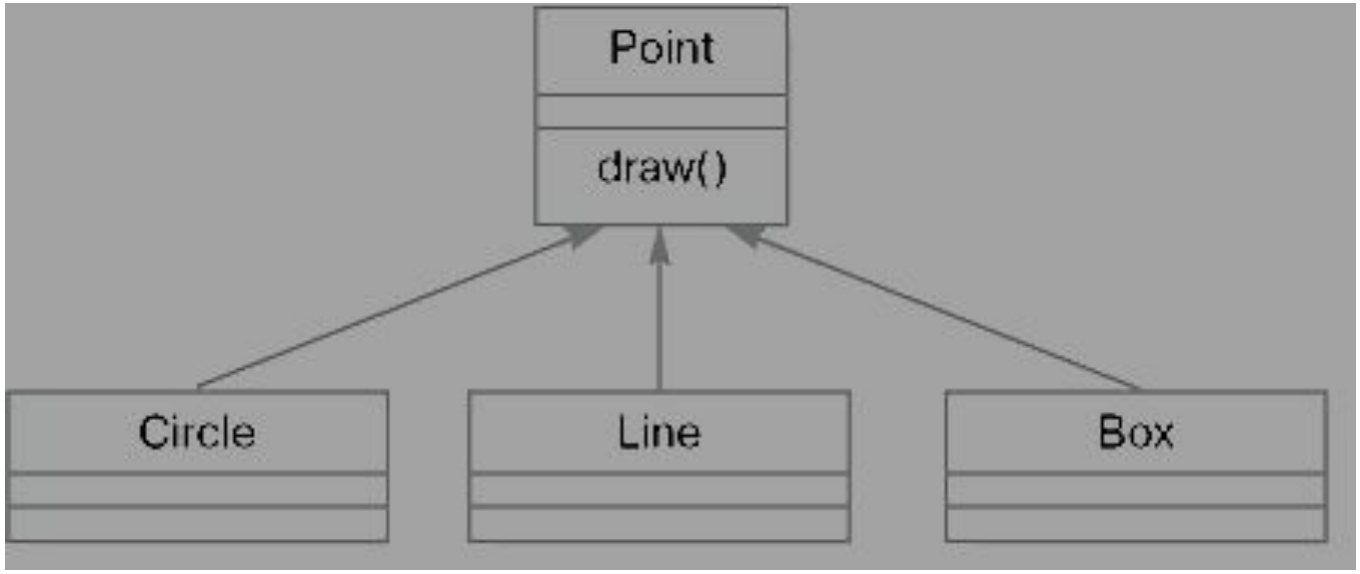
Наследование вводит иерархию "общее/частное", в которой ***подкласс*** наследует от одного или нескольких более общих ***суперклассов***. Подклассы обычно дополняют или переопределяют унаследованную структуру и поведение.

Наследование



Полиморфизм

Слово "*полиморфизм*" греческого происхождения и означает "имеющий много форм".



Полиморфизм

Для описанной выше иерархии классов, используя полиморфизм, можно написать следующий код:

```
…
Point p[] = new Point[1000];
p[0] = new Circle();
p[1] = new Box();
p[2] = new Line();
…
for(int i = 0; i < p.length;i++)
{   if(p[i]!=null) p[i].draw();}…
```

Полиморфизм

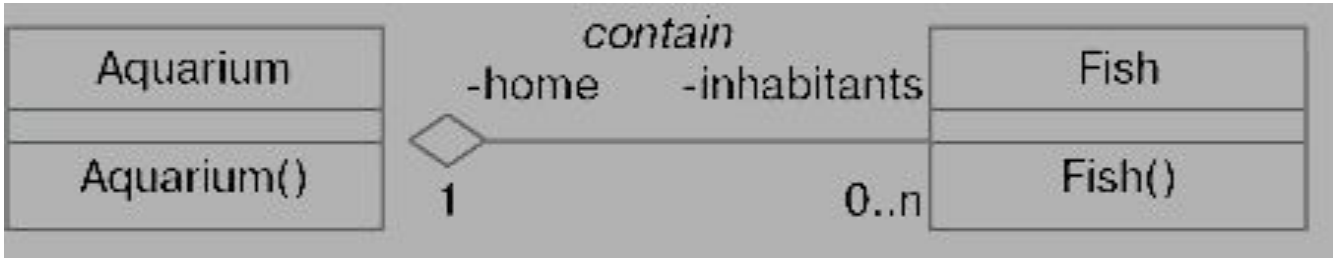
Под *полиморфизмом* в ООП понимают способность одного и того же программного текста x.M (где M - виртуальный метод) выполняться по-разному, в зависимости от того, с каким объектом связана сущность x. Полиморфизм гарантирует, что вызываемый метод M будет принадлежать классу объекта, связанному с сущностью x.

Типы отношений между классами

- **агрегация (Aggregation);**
- ***ассоциация* (Association);**
- ***наследование* (Inheritance);**
- ***метаклассы* (Metaclass).**

Агрегация

Отношение между классами типа "содержит" (contain) или "состоит из" называется агрегацией, или включением.




```
// определение класса Fish
public class Fish {
private Aquarium home;
public Fish() { }}
// определение класса Aquarium
public class Aquarium {
private Fish inhabitants[];
public Aquarium() { }}
```

Ассоциация

Объекты одного класса ссылаются на один или более объектов другого класса, то ни в ту, ни в другую сторону отношение между объектами не носит характера "владения", или контейнеризации

```
public class Programmer {
private Computer computers[];
public Programmer() { }
}
public class Computer {
private Programmer programmers[];
public Computer() { }
}
```

Метаклассы

Каждый *класс*, или описание, всегда имеет строгий **шаблон, задаваемый языком программирования или выбранной объектной моделью**.

Шаблон, задающий различные классы, называется ***метаклассом***.

Итак, *объекты* порождаются от *классов*, а *классы* - от *метакласса*.