# Java Брюс Эккель

## Введение в объекты

### Развитие абстракции

Программист – устанавливает связь между **моделью машины** (в «*пространстве решения*», которое представляет место, где реализуется решение, - например, компьютер) и **моделью задачи**, которую и нужно решать (в «*пространстве задачи*», которое является местом существования задачи – например, прикладной областью).

Элементы *пространства задачи* и их представления в *пространстве решения* называются «**объектом**»

5 основных черт языка **Smalltalk**(первого удачного объектно-ориентированного языка).  
Они также представляют академический подход к объектно-ориентированному программированию:

* **Все является объектом.** Представляйте себе объект как усовершенствованную переменную; он хранит данные, но вы можете «обращаться с запросами» к объекту, требуя у него выполнить операции над собой. Теоретически абсолютно любой компонент решаемой задачи (собака, здание, услуга и т. п.) может быть представлен в виде объекта;
* **Программа – это группа объектов, указывающих друг другу, что делать, посредством сообщений.** Чтобы обратиться с запросом к объекту, вы «посылаете ему сообщение». Более наглядно можно представить сообщение как вызов метода, принадлежащего определенному объекту;
* **Каждый объект имеет собственную «память», состоящую из других объектов.** Иными словами, вы создаете новый объект с помощью встраивания в него уже существующих объектов. Таким образом, можно сконструировать сколь угодно сложную программу, скрыв общую сложность за простотой отдельных объектов;
* **У каждого объекта есть тип.** В других терминах, каждый объект является экземпляром класса, где «класс» является аналогом слова «тип». Важнейшее отличие классов друг от друга как раз и заключается в ответе на вопрос: «Какие сообщения можно посылать объекту?»;
* **Все объекты определенного типа могут получать одинаковые сообщения.** Как мы вскоре убедимся, это очень важное обстоятельство. Так как объект типа «круг» также является объекта типа «фигура», справедливо утверждение, что «круг» заведомо способен принимать сообщения для «фигуры». А это значит, что можно писать код для фигур и быть уверенным в том, что он подойдет для всего, что попадает под понятие фигуры. Взаимозаменяемость представляет одно из самых мощных понятий ООП.

**Объект** обладает ***состоянием****,* ***поведением***и ***индивидуальностью****.*

* **Состояние –** внутренние данные;
* **Поведение –** методы;
* **Индивидуальность –** каждый объект обладает уникальным адресом в памяти.

**Полиморфизм** – способность метода(функции) обрабатывать данные разных типов.  
Кристофер Стрэчи (1967) задлелил его на 2 типа:

* **Параметрический** полиморфизм –исполнение **одного и тот же** кода для всех допустимых типов аргументов;
* **Ad hoc** полиморфизм – исполнение потенциально **разного** кода для каждого типа или подтипа аргумента. «один интерфейс – много реализаций» (Бьерн Страуструп).

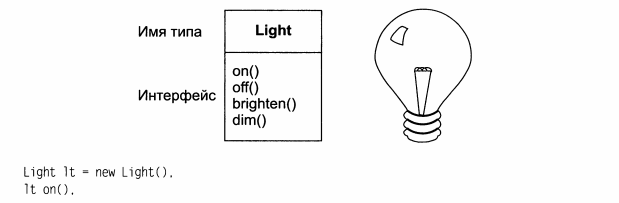
### Объект имеет интерфейс

**Класс –** абстрактный тип данных, служащий для объединение разных объектов в «классы объектов». Определяет набор объектов с идентичными характеристиками (элементы данных) и поведением (функциональность). Создание абстрактных типов данных есть фундаментальное понятие во всем объектно-ориентированном программировании.

Запросы, которые вы можете посылать объекту, определяются его *интерфейсом.*

Код, выполняющий запросы, составляет *реализацию.*

Тип содержит метод для каждого возможного запроса, и при получении определенного запроса вызывается нужный метод.



В данном примере существует *тип (класс)* с именем Light (лампа), конкретный объект типа Light с именем lt, и класс поддерживает различные запросы к объекту Light: выключить лампочку, включить, сделать ярче или притушить. Вы создаете объект Light, определяя «ссылку» на него (lt) и вызывая оператор new для создания нового экземпляра этого типа. Что-бы послать сообщение объекту, следует указать имя объекта и связать его с нужным запросом знаком точки.