Объектно-ориентированный подход основан на систематическом использовании моделей для языково-независимой разработки программной системы, на основе из ее прагматики.

*Прагматика* определяется целью разработки программной системы: для обслуживания клиентов банка, для управления работой аэропорта и т.д. В формулировке цели участвуют предметы и понятия реального мира, имеющие отношение к разрабатываемой программной системе. При объектно-ориентированном подходе эти предметы и понятия заменяются их моделями, т.е. определенными формальными конструкциями, представляющими их в программной системе.

Модель содержит не все признаки и свойства представляемого ею предмета (понятия), а только те, которые существенны для разрабатываемой программной системы. Тем самым модель "беднее", а, следовательно, проще представляемого ею предмета (понятия). Но главное даже не в этом, а в том, что модель есть формальная конструкция: формальный характер моделей позволяет определить формальные зависимости между ними и формальные операции над ними. Это упрощает как разработку и изучение (анализ) моделей, так и их реализацию на компьютере. В частности, формальный характер моделей позволяет получить формальную модель разрабатываемой программной системы как композицию формальных моделей ее компонентов.

Таким образом, объектно-ориентированный подход помогает справиться с такими сложными проблемами, как

* 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.

Объектно-ориентированная разработка программного обеспечения связана с применением объектно-ориентированных моделей при разработке программных систем и их компонентов. Говоря об объектно-ориентированной разработке, я имею в виду:

* объектно-ориентированные методологии (технологии) разработки программных систем;
* инструментальные средства, поддерживающие эти технологии.

Объектно-ориентированная разработка может начаться на самом первом этапе жизненного цикла; она не связана с языком программирования, на котором предполагается реализовать разрабатываемую программную систему: этот язык может и не быть объектно-ориентированным. На этапе разработки объекты - это некоторые формальные конструкции (например, прямоугольники с закругленными углами, с помощью которых они изображаются на схемах), никак пока не связанные с их будущей реализацией на одном из языков программирования.

Объектно-ориентированная разработка программного обеспечения связана с применением объектно-ориентированных методологий (Например, OMT). Обычно эти методологии поддерживаются инструментальными программными средствами, но и без таких средств они полезны, так как позволяют хорошо понять различные аспекты и свойства разрабатываемой программной системы, что в последующем существенно облегчает ее реализацию, тестирование, сопровождение, разработку новых версий и более существенную модификацию.

**Объектно-ориентированный язык программирования** (ОО-язык) — язык,построенный на принципах объектно-ориентированного программирования.

Неполный список объектно-ориентированных языков программирования:

|  |  |  |
| --- | --- | --- |
| * [C#](http://ru.wikipedia.org/wiki/C_Sharp) * [C++](http://ru.wikipedia.org/wiki/C%2B%2B) * [F#](http://ru.wikipedia.org/wiki/F_Sharp) * [Java](http://ru.wikipedia.org/wiki/Java) * [Delphi](http://ru.wikipedia.org/wiki/Delphi_(%D1%8F%D0%B7%D1%8B%D0%BA_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F)) * [Eiffel](http://ru.wikipedia.org/wiki/Eiffel) * [Simula](http://ru.wikipedia.org/wiki/Simula) * [D](http://ru.wikipedia.org/wiki/D_(%D1%8F%D0%B7%D1%8B%D0%BA_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F)) * [Io](http://ru.wikipedia.org/wiki/Io) * [Objective-C](http://ru.wikipedia.org/wiki/Objective-C) | * [Swift](http://ru.wikipedia.org/wiki/Swift_(%D1%8F%D0%B7%D1%8B%D0%BA_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F)) * [Object Pascal](http://ru.wikipedia.org/wiki/Object_Pascal) * [VB.NET](http://ru.wikipedia.org/wiki/Visual_Basic_.NET) * [Visual DataFlex](http://ru.wikipedia.org/wiki/Visual_DataFlex) * [Perl](http://ru.wikipedia.org/wiki/Perl) * [PowerBuilder](http://ru.wikipedia.org/wiki/PowerBuilder) * [Python](http://ru.wikipedia.org/wiki/Python) * [Scala](http://ru.wikipedia.org/wiki/Scala_(%D1%8F%D0%B7%D1%8B%D0%BA_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F)) * [ActionScript](http://ru.wikipedia.org/wiki/ActionScript) (3.0) | * [JavaScript](http://ru.wikipedia.org/wiki/JavaScript) * [JScript .NET](http://ru.wikipedia.org/wiki/JScript_.NET) * [Ruby](http://ru.wikipedia.org/wiki/Ruby) * [Smalltalk](http://ru.wikipedia.org/wiki/Smalltalk) * [Ada](http://ru.wikipedia.org/wiki/%D0%90%D0%B4%D0%B0_(%D1%8F%D0%B7%D1%8B%D0%BA_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F)) * [Xbase++](http://ru.wikipedia.org/wiki/Xbase%2B%2B) * [X++](http://ru.wikipedia.org/wiki/X%2B%2B) * [Vala](http://ru.wikipedia.org/wiki/Vala) * [PHP](http://ru.wikipedia.org/wiki/PHP) * [Cyclone](http://ru.wikipedia.org/wiki/Cyclone) |

В современных ОО языках используются механизмы:

* [*Наследование*](http://ru.wikipedia.org/wiki/%D0%9D%D0%B0%D1%81%D0%BB%D0%B5%D0%B4%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5)). Создание нового класса объектов путём добавления новых элементов ([методов](http://ru.wikipedia.org/wiki/%D0%9C%D0%B5%D1%82%D0%BE%D0%B4)). Некоторые ОО языки позволяют выполнять [множественное наследование](http://ru.wikipedia.org/wiki/%D0%9C%D0%BD%D0%BE%D0%B6%D0%B5%D1%81%D1%82%D0%B2%D0%B5%D0%BD%D0%BD%D0%BE%D0%B5_%D0%BD%D0%B0%D1%81%D0%BB%D0%B5%D0%B4%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5), то есть объединять в одном классе возможности нескольких других классов.
* [*Инкапсуляция*](http://ru.wikipedia.org/wiki/%D0%98%D0%BD%D0%BA%D0%B0%D0%BF%D1%81%D1%83%D0%BB%D1%8F%D1%86%D0%B8%D1%8F_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5)). Сокрытие деталей реализации, которое позволяет вносить изменения в части программы безболезненно для других её частей, что существенно упрощает сопровождение и модификацию ПО.
* [*Полиморфизм*](http://ru.wikipedia.org/wiki/%D0%9F%D0%BE%D0%BB%D0%B8%D0%BC%D0%BE%D1%80%D1%84%D0%B8%D0%B7%D0%BC_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5)). При полиморфизме некоторые части (методы) родительского класса заменяются новыми, реализующими специфические для данного потомка действия. Таким образом, [интерфейс классов](http://ru.wikipedia.org/w/index.php?title=%D0%98%D0%BD%D1%82%D0%B5%D1%80%D1%84%D0%B5%D0%B9%D1%81_%D0%BA%D0%BB%D0%B0%D1%81%D1%81%D0%BE%D0%B2&action=edit&redlink=1) остаётся прежним, а реализация методов с одинаковым названием и набором параметров различается. С полиморфизмом тесно связано позднее связывание.