OOP. Abstraction, inheritance, encapsulation, polymorphism. Ability to provide any example via JavaScript.

Paradigms & Principles

Парадигмы программирования

***Императивное программирование***

#### Ключевые моменты:

В этой парадигме вычисления описываются в виде инструкций, шаг за шагом изменяющих состояние программы.

*Основные понятия:*

— Инструкция

— Состояние

***Структурное программирование***

#### Ключевые моменты:

В структурном программировании мы по-прежнему оперируем состоянием и инструкциями, однако вводится понятие составной инструкции (блока), инструкций ветвления и цикла.

Благодаря этим простым изменениям возможно отказаться от оператора goto в большинстве случаев, что упрощает код.

#### Основные понятия:

— Блок  
— Цикл  
— Ветвление

## Процедурное программирование

*Ключевые моменты:*

Исходная задача разбивается на меньшие (с помощью процедур) и это происходит до тех пор, пока решение всех конкретных процедур не окажется тривиальным.

Процедура — самостоятельный участок кода, который можно выполнить как одну инструкцию.

*Основные понятия:*

— Процедура

## Модульное программирование

#### Ключевые моменты:

Программа, описанная в стиле модульного программирования — это набор модулей. Благодаря модулям впервые в программировании появилась серьезная инкапсуляция — возможно использовать какие-либо сущности внутри модуля, но не показывать их внешнему миру.

Модуль — это отдельная именованная сущность программы, которая объединяет в себе другие программные единицы, близкие по функциональности.

*Основные понятия:*

— Модуль

— Импортирование

***Функциональное программирование***

*Ключевые моменты:*

Функции не производят никаких побочных действий. Они только принимают входные значения, изменяют их и выдают результат

При этом они не изменяют никаких флагов состояния, не записывают никакой информации в глобальные переменные, не пытаются освободить занятую другими объектами память

Кроме того, становится не столь важным соблюдение последовательности выполнения операторов в программе - каждая функция может вычислить своё значение в любой момент, что позволяет легко распараллеливать программы, написанные в функциональном стиле.

Но, пожалуй, самое важное преимущество программ, написанных в функциональном стиле - это, конечно же, их логическая связность и краткость.

*Основные понятия:*

— Функция.

***Объектно-ориентированное программирование***

*Ключевые моменты:*

Объектно-ориентированное программирование (ООП) — это способ организации кода в программе путем его группировки в виде объектов, то есть отдельных элементов, которые включают информацию (значения данных) и функции. Объектно-ориентированный подход к организации программы позволяет группировать определенные фрагменты информации.

В центре ООП находится понятие объекта.

Объект — это сущность, которой можно посылать сообщения и которая может на них реагировать, используя свои данные.

Объект — это экземпляр класса. Данные объекта скрыты от остальной программы. Сокрытие данных называется инкапсуляцией.

Наличие инкапсуляции достаточно для объектности языка программирования, но ещё не означает его объектной ориентированности — для этого требуется наличие наследования.

Но даже наличие инкапсуляции и наследования не делает язык программирования в полной мере объектным с точки зрения ООП. Основные преимущества ООП проявляются только в том случае, когда в языке программирования реализован полиморфизм подтипов — возможность единообразно обрабатывать объекты с различной реализацией при условии наличия общего интерфейса.

*Основные понятия:*

— Класс

— Объект

— Абстракция

— Инкапсуляция

— Наследование

— Полиморфизм

*Прототипное программирование* — стиль объектно-ориентированного программирования, при котором отсутствует понятие класса, а наследование производится путём клонирования существующего экземпляра объекта — прототипа.

Принципы объектно-ориентированного программирования***.***

## *Абстракция* — в объектно-ориентированном программировании это придание объекту характеристик, которые отличают его от всех других объектов.

## Абстракция – процесс выделения только существенных характеристик объекта.

## Абстракция близка к инкапсуляции и сокрытию данных

## Объект должен быть назван так, чтоб было понятно что он из себя представляет, имел все нужные свойства и не содержал лишних.

## *Инкапсуляция* — принцип ООП, позволяющий не задумываться о сложности реализации используемого программного компонента (что у него внутри?), а взаимодействовать с ним посредством предоставляемого интерфейса (публичных методов и членов), а также объединить и защитить жизненно важные для компонента данные. При этом предоставляется только спецификация (интерфейс) объекта. Взаимодействовать с объектом только через этот интерфейс.

## *Наследование* — позволяет описать новый класс на основе уже существующего (родительского), при этом свойства и функциональность родительского класса заимствуются новым классом. Класс-наследник реализует спецификацию уже существующего класса (базовый класс). Это позволяет обращаться с объектами класса-наследника точно так же, как с объектами базового класса. Абстрактный класс — это класс, содержащий хотя бы один абстрактный метод, он описан в программе, имеет поля, методы и не может использоваться для непосредственного создания объекта. То есть от абстрактного класса можно только наследовать. Объекты создаются только на основе производных классов, наследованных от абстрактного. Например, абстрактным классом может быть базовый класс «сотрудник вуза», от которого наследуются классы «аспирант», «профессор» и т. д. Так как производные классы имеют общие поля и функции (например, поле «год рождения»), то эти члены класса могут быть описаны в базовом классе. В программе создаются объекты на основе классов «аспирант», «профессор», но нет смысла создавать объект на основе класса «сотрудник вуза».

## *Полиморфизм* — возможность объектов с одинаковой спецификацией иметь различную реализацию. Например, реализация класса может быть изменена в процессе наследования. Кратко смысл полиморфизма можно выразить фразой: «Один интерфейс, множество реализаций». Полиморфизм позволяет писать более абстрактные программы и повысить коэффициент повторного использования кода.

## *Композиция* —есть класс-контейнер, который включает в себя вызовы других классов. В результате получается, что создавая объект класса-контейнера, мы одновременно создаем и объекты включенных в него классов. Следует понимать, что "композиция" и "наследование" - достаточно разные свойства реальных и виртуальных систем. Наследование предполагает принадлежность к какой-то общности (похожесть), а композиция — формирование целого из частей. При создании объекта, принадлежащего классу-контейнеру, автоматически создаются объекты-части, из которых он как бы состоит. Свойства и методы объектов частей определяются в их классах.

## 

## Links

## http://habrahabr.ru/post/223253/