38. Объектно-ориентированные методы анализа и проектирования ПО. Виртуальные функции и абстрактные классы, перегрузка функций и методов. Шаблоны классов.

Методы анализа и проектирования ПО:
http://citforum.ru/programming/application/program/1.shtml
Виртуальные функции:
http://ru.wikipedia.org/wiki/%D0%92%D0%B8%D1%80%D1%82%D1%83%D0%B0%D0%BB%D1%8C%D0%BD%D1%8B%D0%B9_%D0%BC%D0%B5%D1%82%D0%BE%D0%B4
Абстрактный класс:
http://ru.wikipedia.org/wiki/%D0%90%D0%B1%D1%81%D1%82%D1%80%D0%B0%D0%BA%D1%82%D0%BD%D1%8B%D0%B9_%D0%BA%D0%BB%D0%B0%D1%81%D1%81
Перегрузка:
http://ru.wikipedia.org/wiki/%D0%9F%D0%B5%D1%80%D0%B5%D0%B3%D1%80%D1%83%D0%B7%D0%BA%D0%B0_%D0%BF%D1%80%D0%BE%D1%86%D0%B5%D0%B4%D1%83%D1%80_%D0%B8_%D1%84%D1%83%D0%BD%D0%BA%D1%86%D0%B8%D0%B9
Шаблоны классов (c++ - template, c# - generics):
http://programmersclub.ru/30/
http://ru.wikipedia.org/wiki/%D0%A8%D0%B0%D0%B1%D0%BB%D0%BE%D0%BD%D1%8B_C++

Концептуальной основой **объектно-ориентированного анализа и проектирования ПО** (ООАП) является объектная модель. Большинство современных методов ООАП основаны на использовании языка UML. Унифицированный язык моделирования UML (Unified Modeling Language) представляет собой язык для определения, представления, проектирования и документирования программных систем, организационно-экономических систем, технических систем и других систем различной природы. UML содержит стандартный набор диаграмм и нотаций самых разнообразных видов.

Главное достоинство объектно-ориентированного подхода (ООП): объектно-ориентированные системы более открыты и легче поддаются внесению изменений, поскольку их конструкция базируется на устойчивых формах. Это дает возможность системе развиваться постепенно и не приводит к полной ее переработке даже в случае существенных изменений исходных требований.

Отметим также ряд следующих преимуществ ООП:

- объектная декомпозиция дает возможность создавать программные системы меньшего размера путем использования общих механизмов, обеспечивающих необходимую экономию выразительных средств. Использование ООП существенно повышает уровень унификации разработки и пригодность для повторного использования не только ПО, но и проектов, что в конце концов ведет к сборочному созданию ПО. Системы зачастую получаются более компактными, чем их не объектно-ориентированные эквиваленты, что означает не только уменьшение объема программного кода, но и удешевление проекта за счет использования предыдущих разработок;
- объектная декомпозиция уменьшает риск создания сложных систем ПО, так как она предполагает эволюционный путь развития системы на базе

относительно небольших подсистем. Процесс интеграции системы растягивается на все время разработки, а не превращается в единовременное событие;
- объектная модель вполне естественна, поскольку в первую очередь ориентирована на человеческое восприятие мира, а не на компьютерную реализацию;
- объектная модель позволяет в полной мере использовать выразительные возможности объектных и объектно-ориентированных языков программирования.

К недостаткам ООП относятся некоторое снижение производительности функционирования ПО (которое, однако, по мере роста производительности компьютеров становится все менее заметным) и высокие начальные затраты. Объектная декомпозиция существенно отличается от функциональной, поэтому переход на новую технологию связан как с преодолением психологических трудностей, так и дополнительными финансовыми затратами. При переходе от структурного подхода к объектному, как при всякой смене технологии, необходимо вкладывать деньги в приобретение новых инструментальных средств. Здесь следует учесть расходы на обучение методу, инструментальным средствам и языку программирования. Для некоторых организаций эти обстоятельства могут стать серьезными препятствиями.

Объектно-ориентированный подход не дает немедленной отдачи. Эффект от его применения начинает сказываться после разработки двух-трех проектов и накопления повторно используемых компонентов, отражающих типовые проектные решения в данной области. Переход организации на объектно-ориентированную технологию - это смена мировоззрения, а не просто изучение новых CASE-средств и языков программирования.

Виртуальный метод (виртуальная функция) — в объектно-ориентированном программировании метод (функция) класса, который может быть переопределён в классах-наследниках так, что конкретная реализация метода для вызова будет определяться во время исполнения. Таким образом, программисту необязательно знать точный тип объекта для работы с ним через виртуальные методы: достаточно лишь знать, что объект принадлежит классу или наследнику класса, в котором метод объявлен.

Виртуальные методы — один из важнейших приёмов реализации полиморфизма. Они позволяют создавать общий код, который может работать как с объектами базового класса, так и с объектами любого его класса-наследника. При этом базовый класс определяет способ работы с объектами и любые его наследники могут предоставлять конкретную реализацию этого способа. В некоторых языках программирования, например в Java, нет понятия виртуального метода, данное понятие следует применять лишь для языков, в которых методы родительского класса не могут быть переопределены по умолчанию, а только с помощью некоторых вспомогательных ключевых слов. В некоторых же (как, например, в Python), все методы — виртуальные.

Базовый класс может и не предоставлять реализации виртуального метода, а только декларировать его существование. Такие методы без реализации называются «чистыми виртуальными» (перевод англ. pure virtual) или абстрактными. Класс, содержащий хотя бы один такой метод, тоже будет абстрактным. Объект такого класса создать

нельзя (в некоторых языках допускается, но вызов абстрактного метода приведёт к ошибке). Наследники абстрактного класса должны предоставить реализацию для всех его абстрактных методов, иначе они, в свою очередь, будут абстрактными классами.

Абстрактный класс в объектно-ориентированном программировании — базовый класс, который не предполагает создания экземпляров. Абстрактные классы реализуют на практике один из принципов ООП — полиморфизм. Абстрактный класс может содержать (и не содержать[1]) абстрактные методы и свойства. Абстрактный метод не реализуется для класса, в котором объявлен, однако должен быть реализован для его неабстрактных потомков. Абстрактные классы представляют собой наиболее общие абстракции, то есть имеющие наибольший объём и наименьшее содержание.

Абстрактный метод (или чистый виртуальный метод (pure virtual method - часто неверно переводится как чисто виртуальный метод)) — в объектно-ориентированном программировании, метод класса, реализация для которого отсутствует. Класс, содержащий абстрактные методы, также принято называть абстрактным (там же и пример). Абстрактные методы зачастую путают с виртуальными. Абстрактный метод подлежит определению в классах-наследниках, поэтому его можно отнести к виртуальным, но не каждый виртуальный метод является абстрактным.

Перегрузка процедур и функций — возможность использования одноимённых подпрограмм: процедур или функций в языках программирования. В большинстве ранних языков программирования, для упрощения процесса трансляции существовало ограничение, согласно которому одновременно в программе не может быть доступно более одной процедуры с одним и тем же именем. В соответствии этому ограничению, все подпрограммы, видимые в данной точке программы, должны иметь различные имена. Для того, чтобы иметь возможность использовать несколько вариантов подпрограммы с одним и тем же именем, но с разным числом аргументов или другими типами аргументов (то есть с разной сигнатурой) и вводится перегрузка подпрограмм. Такая перегрузка возможна в рамках процедурной парадигмы, без применения объектно-ориентированного программирования.

Перегружаемые функции имеют одинаковое имя, но разное количество или типы аргументов. Это разновидность статического полиморфизма, при которой вопрос о том, какую из функций вызвать, решается по списку ее аргументов. Этот подход применяется в статически типизированных языках, которые проверяют типы аргументов при вызове функции. Перегруженная функция фактически представляет собой несколько разных функций, и выбор подходящей происходит на этапе компиляции. Перегрузку функций не следует путать с формами полиморфизма, где правильный метод выбирается во время выполнения, например, посредством виртуальных функций, а не статически.

Шаблоны позволяют определить семейство функций или классов, которые могут работать с различными типами данных. Шаблоны используются в случаях дублирования одного и того же кода для нескольких типов. Например, можно использовать шаблоны функций для создания набора функций, которые применяют один и тот же алгоритм к различным типам данных. Кроме того, шаблоны классов можно использовать для разработки набора типобезопасных классов.