37. Объектно-ориентированные методы анализа и проектирования ПО. Основные принципы построения объектной модели. Основные элементы объектной модели. Инкапсуляция, наследование, полиморфизм.

Методы анализа и проектирования ПО:
http://citforum.ru/programming/application/program/1.shtml
Кратко о объектно-ориентированном проектировании (пункты 1.3-1.4), подробнее пункт 2.1 (в конце), совсем подробно глава 6.
http://www.helloworld.ru/texts/comp/other/oop/index.htm
Основные принципы построения объектной модели:
http://wiki.clan.su/publ/osnovnye_principy_postroenija_obektnoj_modeli_osnovnye_ehlementy_obektnoj_modeli/1-1-0-40
Основные элементы объектной модели (pdf – 162 страница):
http://vernikov.ru/media/K2/item_attachments/vendorov.pdf
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)
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)
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)

http://habrahabr.ru/post/37576/

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.

Основные принципы построения объектной модели.

Объектно-ориентированная технология основывается на так называемой объектной модели. Основными ее принципами является:

- абстрагирование
- инкапсуляция
- модульность
- иерархичность
- типизация
- параллелизм
- сохраняемость.

Каждый из этих принципов собственно не новый, но в объектной модели они впервые применены в совокупности. Первые четыре понятия является

обязательными в том понимании, что без каждого из них модель не будет объектно-ориентированной. Другие являются дополнительными, имея в виду, что они полезны в объектной модели, но не обязательные.

Абстрагирование — это выделение наиболее важных, существенных характеристик некоторого объекта, которые отличают его от всех других видов объектов и, таким образом, четко определяют его концептуальные границы с точки зрения дальнейшего рассмотрения и анализа, и игнорирование менее важных или незначительных деталей. Абстрагирование позволяет управлять сложностью системы, концентрируясь на существенных свойствах объекта. Абстрагирование концентрирует внимание на внешних особенностях объекта и позволяет отделить самые существенные особенности его поведения от деталей их реализации. Выбор правильного набора абстракций для заданной предметной области представляет собой главную задачу объектно-ориентированного проектирования. Абстракция зависит от предметной области и точки зрения - то, что важно в одном контексте, может быть не важно в другом. Объекты и классы — основные абстракции предметной области.

Инкапсуляция — физическая локализация свойств и поведения в рамках единственной абстракции (рассматриваемой как «черный ящик»), скрывающая их реализацию за общедоступным интерфейсом. Инкапсуляция — это процесс отделения друг от друга отдельных элементов объекта, определяющих его устройство и поведение. Инкапсуляция служит для того, чтобы изолировать интерфейс объекта, отражающий его внешнее поведение, от внутренней реализации объекта. Объектный подход предполагает, что собственные ресурсы, которыми могут манипулировать только операции самого объекта, скрыты от внешней среды. Абстрагирование и инкапсуляция являются взаимодополняющими: абстрагирование фокусирует внимание на внешних особенностях объекта, а инкапсуляция (или иначе ограничение доступа) не позволяет объектам-пользователям различать внутреннее устройство объекта.

По-другому инкапсуляцию можно описать, сказав, что приложение разделяется на небольшие фрагменты связанной функциональности.
Еще одним преимуществом инкапсуляции является ограничение последствий изменений, вносимых в систему. Инкапсуляция подобна понятию сокрытия информации (information hiding). Это возможность скрывать многочисленные детали объекта от внешнего мира. Внешний мир объекта ~ это все, что находится вне его, включая остальную часть системы. Сокрытие информации предоставляет то же преимущество, что и инкапсуляция, — гибкость.

Модульность — это свойство системы, связанное с возможностью ее декомпозиции на ряд внутренне сильно сцепленных, но слабо связанных между собой подсистем (модулей). Модульность снижает сложность системы, позволяя выполнять независимую разработку отдельных модулей. Инкапсуляция и модульность создают барьеры между абстракциями.

Иерархия — это ранжированная или упорядоченная система абстракций, расположение их по уровням. Основными видами иерархических структур применительно к сложным системам являются структура классов (иерархия по номенклатуре) и структура объектов (иерархия по составу). Примерами иерархии классов являются простое и множественное наследование (один класс использует структурную или функциональную часть соответственно одного или нескольких других классов), а иерархии объектов - агрегация.

Основные элементы объектной модели.

К основным понятиям объектно-ориентированного подхода (элементам объектной модели) относятся:
• объект;
• класс;
• атрибут;
• операция;
• полиморфизм (интерфейс);
• компонент;
• связи.

Наследование

Наследование позволяет создавать новые классы, которые повторно используют, расширяют и изменяют поведение, определенное в других классах. Класс, члены которого наследуются, называется *базовым классом*, а класс, который наследует эти члены, называется *производным классом*. Производный класс может иметь только один непосредственный базовый класс. Однако наследование является транзитивным. Если ClassC является производным от ClassB, и ClassB является производным от ClassA, ClassC наследует члены, объявленные в ClassB и ClassA.

Концептуально, производный класс является специализацией базового класса. Например, при наличии базового класса Animal, возможно наличие одного производного класса, который называется Mammal, и еще одного производного класса, который называется Reptile. Mammal является Animal и Reptile является Animal, но каждый производный класс представляет разные специализации базового класса.

При определении класса для наследования от другого класса, производный класс явно получает все члены базового класса, за исключением его конструкторов и деструкторов. Производный класс может таким образом повторно использовать код в базовом классе без необходимости в его повторной реализации. В производном классе можно добавить больше членов. Таким образом, производный класс расширяет функциональность базового класса.

Полиморфизм

Полиморфизм (polymorphism) - это свойство, которое позволяет одно и то же имя использовать для решения двух или более схожих, но технически разных задач. Целью полиморфизма, применительно к объектно-ориентированному программированию, является использование одного имени для задания общих для класса действий. Выполнение каждого конкретного действия будет определяться типом данных.
Это понятие имеет два различающихся аспекта.

1. Во время выполнения объекты производного класса могут рассматриваться как объекты базового класса в таких местах как параметры метода и коллекции или массивы. При этом объявленный тип объекта больше не идентичен его типу времени выполнения.

2. Базовые классы могут определять и реализовывать виртуальные *методы*, а производные классы могут переопределять их. Это означает, что они предоставляют свои собственные определения и реализацию. Во время выполнения, когда клиентский код вызывает метод, среда CLR ищет тип времени выполнения объекта и вызывает это переопределение виртуального метода. Таким образом, в исходном коде можно вызвать метод в базовом классе и вызвать выполнение метода с версией производного класса.

Полиморфизм может применяться также и к операторам. Фактически во всех языках программирования ограниченно применяется полиморфизм, например, в арифметических операторах. Так, в Си, символ + используется для складывания целых, длинных целых, символьных переменных и чисел с плавающей точкой. В этом случае компилятор автоматически определяет, какой тип арифметики требуется. В C++ вы можете применить эту концепцию и к другим, заданным вами, типам данных. Такой тип полиморфизма называется перегрузкой операторов (operator overloading).

Ключевым в понимании полиморфизма является то, что он позволяет вам манипулировать объектами различной степени сложности путём создания общего для них стандартного интерфейса для реализации похожих действий.