

# Словарь терминов: OO.A&D&P

Гради Буч

## **CRC-КАРТОЧКИ, CRC CARDS:**

CRC - Class/Responsibilities/Collaborators, Класс/Ответственности/Сотрудники; простое, но достаточно эффективное средство мозгового штурма при выявлении ключевых абстракций и механизмов.

## **АБСТРАКТНАЯ ОПЕРАЦИЯ, ABSTRACT OPERATION:**

Объявленная, но не реализованная операция в абстрактном классе. В C++ абстрактные операции объявляются как чисто виртуальные функции-члены.

## **АБСТРАКТНЫЙ КЛАСС, ABSTRACT CLASS:**

Класс, который не может иметь экземпляров. Абстрактный класс пишется в предположении, что его конкретные подклассы дополняют его структуру и поведение, скорее всего, реализовав абстрактные операции.

## **АБСТРАКЦИЯ, ABSTRACTION:**

Существенные характеристики объекта, которые отличают его от всех других объектов и четко определяют его концептуальные границы для наблюдателя. Абстрагирование - процесс выявления абстракций. Один из основных элементов объектной модели.

## **АГЕНТ, AGENT:**

Объект, который подвергается воздействию со стороны и сам воздействует на другие объекты. Обычно агенты создаются для выполнения некоторой работы по поручению актеров или других агентов.

## **АКТЕР, ACTOR:**

Объект, воздействующий на другие объекты, но сам не подвергающийся воздействию с их стороны. В некоторых контекстах то же самое, что активный объект.

## **АКТИВНЫЙ ОБЪЕКТ, ACTIVE OBJECT:**

Объект, которому выделен свой поток управления.

## **АЛГОРИТМИЧЕСКАЯ ДЕКОМПОЗИЦИЯ, ALGORITHMIC DECOMPOSITION:**

Процесс разделения системы на части, каждая из которых отражает этап общего процесса. Применение структурного подхода к проектированию приводит к алгоритмической декомпозиции, которая фокусируется на потоке управления в системе.

**АРХИТЕКТУРА МОДУЛЕЙ, MODULE ARCHITECTURE:**

Граф, вершины которого соответствуют модулям, а ребра - отношениям модулей между собой. Архитектура модулей системы представляется совокупностью диаграмм модулей.

**АРХИТЕКТУРА ПРОЦЕССОВ, PROCESS ARCHITECTURE:**

Граф, вершины которого соответствуют процессорам и устройствам, а ребра - соединениям между ними. Для описания архитектуры процессов системы используются диаграммы процессов.

**АРХИТЕКТУРА, ARCHITECTURE:**

Логическая и физическая структура системы, сформированная всеми стратегическими и тактическими проектными решениями.

**АССОЦИАЦИЯ, ASSOCIATION:**

Отношение, означающее некоторую смысловую связь между классами.

**АТТРИБУТ, ATTRIBUTE:**

Часть составного объекта (агрегата).

**БАЗОВЫЙ КЛАСС, BASE CLASS:**

Наиболее общий класс в какой-либо структуре классов. В большинстве приложений есть несколько таких корневых классов. В некоторых языках программирования определяется всеобщий базовый класс, который является суперклассом для всех остальных классов.

**БЛОКИРУЮЩИЙ ОБЪЕКТ, BLOCKING OBJECT:**

Пассивный объект, способный работать в многопоточном окружении. Вызов операции блокирующего объекта блокирует клиента на все время операции.

**ВИДИМОСТЬ, VISIBILITY:**

Способность одной абстракции видеть другую и, таким образом, ссылаться на ее ресурсы извне. Абстракции видимы друг другу, только если они находятся в одном пространстве имен. Контроль экспорта может еще более ограничить доступ к видимым абстракциям.

**ВИРТУАЛЬНАЯ ФУНКЦИЯ, VIRTUAL FUNCTION:**

Какая-либо операция над объектом. Виртуальная функция может быть переопределена в подклассах, следовательно, ее реализация определяется всем множеством методов, объявленных во всех классах дерева наследования. Термины «обобщенная функция» и «виртуальная функция» взаимозаменяемы.

**ВРЕМЕННАЯ СЛОЖНОСТЬ, TIME COMPLEXITY:**

Относительное или абсолютное время, за которое выполняется операция.

**ДЕЙСТВИЕ, ACTION:**

Некое происшествие в системе, требующее, с практической точки зрения, нулевого времени для своего завершения. Действием может быть вызов операции, запуск другого события, начало или остановка деятельности.

**ДЕЛЕГИРОВАНИЕ, DELEGATION:**

При делегировании один объект, ответственный за операцию, передает выполнение этой операции другому объекту.

**ДЕСТРУКТОР, DESTRUCTOR:**

Операция класса, которая освобождает состояние объекта и/или уничтожает сам объект.

**ДЕЯТЕЛЬНОСТЬ, ACTIVITY:**

Операция, выполнение которой требует некоторого времени.

**ДИАГРАММА ВЗАИМОДЕЙСТВИЙ, INTERACTION DIAGRAM:**

Часть системы обозначения объектно-ориентированного проектирования используется для демонстрации выполнения какого-либо сценария в контексте диаграммы объектов.

**ДИАГРАММА КЛАССОВ, CLASS DIAGRAM:**

Часть системы обозначений объектно-ориентированного проектирования используется, чтобы наглядно показать классы и их взаимоотношения в логическом проекте системы. Может представлять всю структуру классов или ее часть.

**ДИАГРАММА МОДУЛЕЙ, MODULE DIAGRAM:**

Часть системы обозначений объектно-ориентированного проектирования используется для демонстрации разбиения классов и объектов по модулям в физическом проекте системы. Диаграмма модулей отображает архитектуру модулей системы.

**ДИАГРАММА ОБЪЕКТОВ, OBJECT DIAGRAM:**

Часть системы обозначений объектно-ориентированного проектирования используется, чтобы наглядно показать объекты и отношения между ними в логическом проекте системы. Может отражать всю объектную структуру или часть ее; обычно иллюстрирует смысл механизмов в логическом проекте. Отдельная диаграмма объектов - моментальный снимок из жизни системы.

**ДИАГРАММА ПЕРЕХОДОВ И СОСТОЯНИЙ, STATE TRANSITION DIAGRAM:**

Часть обозначений объектно-ориентированного проектирования; используется для отображения пространства состояний данного класса, событий, которые вызывают переход из одного состояния в другое, и действий, возникающих в результате смены состояния.

**ДИАГРАММА ПРОЦЕССОВ, PROCESS DIAGRAM:**

Часть системы обозначений объектно-ориентированного проектирования используется, чтобы наглядно показать, как процессы размещены по процессорам в физическом проекте системы. Диаграмма процессов отражает архитектуру процессов.

**ДИНАМИЧЕСКОЕ СВЯЗЫВАНИЕ, DYNAMIC BINDING:**

Связывание означает установление соответствия имени (например, объявленной переменной) с классом. Динамическое связывание происходит при выполнении программы в тот момент, когда создается объект, обозначенный именем.

**ДРУГ, FRIEND:**

Класс или операция, имеющие доступ к закрытым операциям или данным некоторого класса. Только сам класс может называть своих друзей.

**ЗАКРЫТАЯ ЧАСТЬ, PRIVATE:**

Часть интерфейса какого-либо класса, объекта или модуля, закрытая (невидимая) для других классов, объектов и модулей.

**ЗАЩИЩЕННАЯ ЧАСТЬ, PROTECTED:**

Часть интерфейса какого-либо класса, объекта или модуля, невидимая для всех других классов, объектов и модулей за исключением подклассов.

**ИДЕНТИЧНОСТЬ, IDENTITY:**

Природа объекта; то, что отличает его от других объектов.

**ИДИОМА, IDIOM:**

Выражение, общепринятое в каком-либо языке программирования или культуре какого-либо приложения, отражающее общепринятый способ использования данного языка.

**ИЕРАРХИЯ, HIERARCHY:**

Подчинение или упорядочение абстракций. Две типичных иерархии в сложной системе - структура классов (включая иерархию «общее/частное») и структура объектов (включая иерархию «целое/часть»); иерархии можно также обнаружить в архитектурах модулей и процессов.

**ИНВАРИАНТ, INVARIANT:**

Логическое выражение некоторого условия, истинность которого необходимо соблюдать.

**ИНКАПСУЛЯЦИЯ, ENCAPSULATION:**

Процесс разделения элементов абстракции, которые образуют ее структуру и поведение. Служит для отделения внешних обязательств объекта от его реализации.

**ИНСТАНЦИРОВАНИЕ, INSTANTIATION:**

Подстановка параметров шаблона обобщенного или параметризованного класса; в результате создается конкретный класс, который может иметь экземпляры.

**ИНТЕРФЕЙС, INTERFACE:**

Внешний вид класса, объекта или модуля, выделяющий его существенные черты и не показывающий внутреннего устройства и секретов поведения.

**ИСКЛЮЧЕНИЕ, EXCEPTION:**

Возбуждение исключения показывает, что некоторый логический инвариант не соблюдается. В C++ мы возбуждаем исключение, чтобы избежать неправомерное исполнение операций и дать знать о возникшей проблеме другим объектам, которые могут перехватить исключение и принять меры.

**ИСПОЛЬЗОВАТЬ, USE:**

Ссылаться на абстракцию извне.

**ИТЕРАТОР, ITERATOR:**

Операция, позволяющая навещать части некоторого объекта.

**КАТЕГОРИЯ КЛАССОВ, CLASS CATEGORY:**

Логически полный набор классов, одни из которых видимы для других категорий классов, а другие - нет. Классы в категории сотрудничают для предоставления некоторого набора услуг.

**КЛАСС, CLASS:**

Множество объектов с общей структурой и поведением. Термины «класс» и «тип» в большинстве случаев (но не всегда) взаимозаменяемы. Понятие класса отличается от понятия типа тем, что концентрируется на классификации по структуре и поведению.

**КЛАСС-КОНТЕЙНЕР, CONTAINER CLASS:**

Класс, экземпляры которого представляют собой коллекции других объектов. Контейнер может быть однородным (коллекции включают экземпляры только одного класса) либо неоднородным (коллекции включают экземпляры разных классов, имеющих обычно общий суперкласс). В C++ контейнеры обычно определяются как параметризованные классы с параметром, обозначающим класс объектов коллекции.

**КЛИЕНТ, CLIENT:**

Объект, который пользуется услугами другого объекта либо выполняя операции над последним, либо через доступ к его состоянию.

**КЛЮЧ, KEY:**

Атрибут, значение которого однозначно идентифицирует объект.

**КЛЮЧЕВАЯ АБСТРАКЦИЯ, KEY ABSTRACTION:**

Класс или объект, являющийся частью словаря предметной области.

**КОНКРЕТНЫЙ КЛАСС, CONCRETE CLASS:**

Класс, реализация которого завершена и который, поэтому, может иметь экземпляры.

**КОНСТРУКТОР, CONSTRUCTOR:**

Операция, создающая объект и/или инициализирующая его состояние.

**МЕТАКЛАСС, METACLASS:**

Класс класса; класс, экземпляры которого сами являются классами.

**МЕТОД, МЕТОД:**

Операция над объектом, определенная как часть описания класса. Не любая операция является методом, но все методы - операции. Термины «метод», «сообщение» и «операция» обычно взаимозаменяемы. В некоторых языках методы существуют сами по себе и могут переопределяться подклассами; в других языках метод не может быть переопределен, - он служит как часть реализации обобщенных или виртуальных функций, которые можно переопределять в подклассах.

**МЕХАНИЗМ, MECHANISM:**

Структура, посредством которой объекты сотрудничают друг с другом, осуществляя поведение, которое соответствует требованиям системы.

**МОДИФИКАТОР, MODIFIER:**

Операция, изменяющая состояние объекта.

**МОДУЛЬ, MODULE:**

Единица кода, служащая строительным блоком физической структуры системы программный блок, который содержит объявления, выраженные в соответствии с требованиями языка и образующие физическую реализацию части или всех классов и объектов логического проекта системы. Как правило, модуль состоит из интерфейсной части и реализации.

**МОДУЛЬНОСТЬ, MODULARITY:**

Свойство системы, которая была разделена на связанные и слабо зацепленные между собой модули.

**МОНОМОРФИЗМ, MONOMORPHISM:**

Положение теории типов, согласно которому имена (например, переменных) могут обозначать только объекты одного и того же класса.

**МОЩНОСТЬ, CARDINALITY:**

Число экземпляров класса: число экземпляров, участвующих в связи классов.

**НАСЛЕДОВАНИЕ, INHERITANCE:**

Отношение между классами, при котором класс использует структуру или поведение другого (одиночное наследование) или других (множественное наследование) классов. Наследование вводит иерархию «общее/частное» в которой подкласс наследует от одного или нескольких более общих суперклассов. Подклассы обычно дополняют или переопределяют унаследованную структуру и поведение.

**ОБОБЩЕННАЯ ФУНКЦИЯ, GENERIC FUNCTION:**

Какая-либо операция над объектом. Обобщенная функция класса может быть переопределена в подклассах; следовательно, ее реализация определяется всем множеством методов, объявленных во всех классах дерева наследования. Термины «обобщенная функция» и «виртуальная функция» взаимозаменяемы.

**ОБОБЩЕННЫЙ КЛАСС, GENERIC CLASS:**

Класс, служащий шаблоном для создания других классов: шаблон параметризуется другими классами, объектами и/или операциями. Обобщенный класс до создания объектов должен быть инстанцирован. Обобщенные классы используются как контейнерные классы. Термины «обобщенный класс» и «параметризованный класс» взаимозаменяемы.

**ОБРАТНЫЙ ИНЖИНИРИНГ, REVERSE-ENGINEERING:**

Восстановление логической или физической модели системы по коду. Противопоставляется прямому инжинирингу.

**ОБЪЕКТ, ОБЪЕСТ:**

Нечто, чем можно оперировать. Объект имеет состояние, поведение и идентичность. Структура и поведение сходных объектов определены в общем для них классе. Термины «экземпляр» и «объект» взаимозаменяемы.

**ОБЪЕКТНАЯ МОДЕЛЬ, ОБЪЕСТ MODEL:**

Совокупность основополагающих принципов, лежащих в основе объектно-ориентированного проектирования; парадигма программирования, основанная на принципах абстрагирования, инкапсуляции, модульности, иерархичности, типизации, параллелизма и устойчивости.

**ОБЪЕКТНОЕ ПРОГРАММИРОВАНИЕ, ОБЪЕСТ-BASED PROGRAMMING:**

Метод программирования, основанный на представлении программы как совокупности объектов, каждый из которых является экземпляром некоторого типа. Типы образуют иерархию, но не наследственную. В таких программах типы рассматриваются как статические, а объекты имеют более динамическую природу, которую ограничивают статическое связывание и мономорфизм.

**ОБЪЕКТНО-ОРИЕНТИРОВАННАЯ ДЕКОМПОЗИЦИЯ, ОБЪЕСТ-ORIENTED DECOMPOSITION:**

Процесс разбиения системы на части, соответствующие классам и объектам предметной области. Практическое применение методов объектно-ориентированного проектирования приводит к объектно-ориентированной декомпозиции, при которой мы рассматриваем мир как совокупность объектов, согласованно действующих для обеспечения требуемого поведения.

**ОБЪЕКТНО-ОРИЕНТИРОВАННОЕ ПРОГРАММИРОВАНИЕ, ОБЪЕСТ-ORIENTED PROGRAMMING (ООП):**

Методология реализации, при которой программа организуется, как совокупность сотрудничающих объектов, каждый из которых является экземпляром какого-либо класса, а классы образуют иерархию наследования. При этом классы обычно статичны, а объекты очень динамичны, что поощряется динамическим связыванием и полиморфизмом.

**ОБЪЕКТНО-ОРИЕНТИРОВАННОЕ ПРОЕКТИРОВАНИЕ, OBJECT-ORIENTED DESIGN (OOD):**

Методология проектирования, соединяющая процесс объектно-ориентированной декомпозиции и систему обозначений для представления логической и физической, статической и динамической моделей проектируемой системы. Система обозначений состоит из диаграмм классов, объектов, модулей и процессов.

**ОБЪЕКТНО-ОРИЕНТИРОВАННЫЙ АНАЛИЗ, OBJECT-ORIENTED ANALYSIS:**

Метод анализа, согласно которому требования рассматриваются с точки зрения классов и объектов, составляющих словарь предметной области.

**ОБЪЕКТ-ЧЛЕН, MEMBER OBJECT:**

Часть состояния объекта. В совокупности объекты-члены полностью определяют структуру объекта. Термины «переменная экземпляра», «поле», «объект-член» и «слот» взаимозаменяемы.

**ОГРАНИЧЕНИЕ, CONSTRAINT:**

Выражение некоторого смыслового условия, которое должно выполняться.

**ОПЕРАЦИЯ КЛАССА, CLASS OPERATION:**

Операция, например, конструктор или деструктор, общая для всего класса и не принадлежащая конкретному объекту.

**ОПЕРАЦИЯ, OPERATION:**

Нечто, проделываемое одним объектом над другим, чтобы вызвать реакцию. Все операции, которые можно выполнить над каким-либо объектом, сосредоточены в свободных подпрограммах и функциях-членах (методах). Термины «операция», «метод» и «сообщение» взаимозаменяемы.

**ОТВЕТСТВЕННОСТЬ, RESPONSIBILITY:**

Поведение, за которое ответственен объект.

**ОТКРЫТАЯ ЧАСТЬ, PUBLIC:**

Часть интерфейса какого-либо класса, объекта или модуля, открытая (видимая) для всех классов, объектов и модулей.

**ПАРАЛЛЕЛИЗМ, CONCURRENCY:**

Свойство, отличающее активные объекты от неактивных.

**ПАРАЛЛЕЛЬНЫЙ ОБЪЕКТ, CONCURRENT OBJECT:**

Активный объект, способный работать в многопоточной среде.

**ПАРАМЕТРИЗОВАННЫЙ КЛАСС, PARAMETERIZED CLASS:**

Класс, служащий шаблоном для других классов; шаблон параметризуется другими классами, объектами и/или операциями. Параметризованный класс должен быть инстанцирован до создания объектов. Параметризованные классы используются как контейнеры. Термины «обобщенный класс» и «параметризованный класс» взаимозаменяемы.

**ПАССИВНЫЙ ОБЪЕКТ, PASSIVE OBJECT:**

Объект, не имеющий собственного потока управления.

**ПЕРЕМЕННАЯ КЛАССА, CLASS VARIABLE:**

Часть состояния класса. Совокупность всех переменных класса образует его структуру. Переменные класса совместно используются всеми его экземплярами. В C++ переменная класса объявляется как статический член.

**ПЕРЕМЕННАЯ ЭКЗЕМПЛЯРА, INSTANCE VARIABLE:**

Часть состояния объекта. В совокупности переменные экземпляра полностью определяют структуру объекта. Термины «переменная экземпляра», «поле», «объект-член» и «слот» взаимозаменяемы.

**ПЕРЕХОД, TRANSITION:**

Переход из одного состояния в другое.

**ПОВЕДЕНИЕ, BEHAVIOR:**

Действия и реакции объекта, выраженные в терминах передачи сообщений и изменения состояния; видимая извне и воспроизводимая активность объекта.

**ПОДКЛАСС, SUBCLASS:**

Класс, наследующий от одного или нескольких классов (которые называются его непосредственными суперклассами).

**ПОДСИСТЕМА, SUBSYSTEM:**

Совокупность модулей, часть которых видима для других подсистем, а часть - скрыта.

**ПОЛЕ, FIELD:**

Часть состояния объекта; совокупность полей объекта образуют его структуру. Термины «поле», «переменная экземпляра», «объект-член» и «слот» означают одно и то же.

**ПОЛИМОРФИЗМ, POLYMORPHISM:**

Положение теории типов, согласно которому имена (например, переменных) могут обозначать объекты разных (но имеющих общего родителя) классов. Следовательно, любой объект, обозначаемый полиморфным именем, может по-своему реагировать на некий общий набор операций.

**ПОСЛЕДОВАТЕЛЬНОЕ ПРОЕКТИРОВАНИЕ, ROUND-TRIP GESTALT DESIGN:**

Стиль проектирования, который подчеркивает последовательность и итеративность в развитии системы: посредством уточнения различных, хотя и согласованных логических и физических представлений системы в целом; объектно-ориентированное проектирование основывается на последовательном проектировании, что является выражением взаимозависимости общей картины проекта и его деталей.

**ПОСЛЕДОВАТЕЛЬНЫЙ ОБЪЕКТ, SEQUENTIAL OBJECT:**

Пассивный объект, рассчитанный на работу в однопоточном окружении.

**ПОСТУСЛОВИЕ, POSTCONDITION:**

Инвариант, соблюдаемый на выходе из операции.

**ПОТОК УПРАВЛЕНИЯ, THREAD OF CONTROL:**

Отдельный процесс. Запуск потока управления приводит к возникновению независимой динамической деятельности в системе; данная система может иметь несколько одновременно выполняемых потоков, некоторые из которых могут динамически возникать и уничтожаться. Многопроцессорные системы допускают истинную многопоточность. в то время как на однопроцессорных компьютерах возможна только иллюзия многопоточности. (Термин «thread of control» переводится также «нить управления». В данном издании принят перевод «поток управления» как более распространенный. Отметим, что в некоторых случаях автор использует термин «flow of control», который переведен также. - *Примеч. ред.*)

**ПРЕДУСЛОВИЕ, PRECONDITION:**

Инвариант, предполагаемый на входе в операцию.

**ПРИМЕСЬ, MIXIN:**

Класс, реализующий какое-либо четко выделенное поведение; используется для уточнения поведения других классов посредством наследования; поведение примеси обычно ортогонально поведению класса, с которым она смешивается.

**ПРОСТРАНСТВЕННАЯ СЛОЖНОСТЬ, SPACE COMPLEXITY:**

Относительный или абсолютный объем памяти, занимаемый объектом.

**ПРОСТРАНСТВО СОСТОЯНИЙ, STATE SPACE:**

Перечислимое множество всех возможных состояний объекта. Пространство состояний программы содержит неопределенное, но конечное число состояний (не обязательно желаемых или ожидаемых).

**ПРОТОКОЛ, PROTOCOL:**

Способы, которыми объекты могут действовать и реагировать; полное статическое и динамическое представление объекта; протокол объекта определяет допустимое поведение объекта.

**ПРОЦЕСС, PROCESS:**

Запуск одного потока управления.

**ПРОЦЕССОР, PROCESSOR:**

Часть аппаратного обеспечения, имеющая вычислительные ресурсы.

**ПРЯМОЙ ИНЖИНИРИНГ, FORWARD-ENGINEERING:**

Создание исполнимого кода по логической или физической модели. Противопоставляется обратному инжинирингу.

**РАЗДЕЛ, PARTITION:**

Категории классов или подсистемы, составляющие часть данного уровня абстракции.

**РЕАКТИВНАЯ СИСТЕМА, REACTIVE SYSTEM:**

Система, движимая событиями. Поведение такой системы не определяется простым отображением «вход-выход».

**РЕАЛИЗАЦИЯ, IMPLEMENTATION:**

Внутреннее представление класса, объекта или модуля, включая секреты его поведения.

**РОЛЬ, ROLE:**

Способность или цель, с которой класс или объект участвует в отношениях с другими; некоторая четко выделяемая черта поведения объекта в определенный момент времени; роль - это лицо, которое объект являет миру в данный момент.

**СВОБОДНАЯ ПОДПРОГРАММА, FREE SUBPROGRAM:**

Процедура или функция, которая выполняется как непримитивная операция над объектом или объектами одного и того же или различных классов. Свободная подпрограмма - это любая подпрограмма, которая не является методом какого-либо класса.

**СВЯЗЬ, LINK:**

Связь между объектами, экземпляр ассоциации.

**СЕЛЕКТОР, SELECTOR:**

Операция, имеющая доступ к состоянию объекта, но не изменяющая его.

**СЕРВЕР, SERVER:**

Объект, который никогда не воздействует на другие объекты, но используется ими; объект, предоставляющий некоторые услуги.

**СИГНАТУРА, SIGNATURE:**

Полная спецификация операции с указанием типов аргументов и возвращаемого значения.

**СИЛЬНО ТИПИЗИРОВАННЫЙ, STRONGLY TYPED:**

Свойство языка программирования, в соответствии с которым во всех выражениях гарантируется согласованность типов.

**СИНХРОНИЗАЦИЯ, SYNCHRONIZATION:**

Семантика параллельности операции. Операция может быть простой (присутствует только один поток управления); синхронной (рандеву двух потоков); односторонняя (рандеву, при котором одному из потоков приходится ждать); по истечении времени (рандеву, в котором один процесс ждет другого определенное время); асинхронной (два процесса независимы друг от друга).

**СИСТЕМА РЕАЛЬНОГО ВРЕМЕНИ, REAL-TIME SYSTEM:**

Система, в которой некоторые существенные процессы должны укладываться в отведенное время. Система «жесткого» реального времени должна быть детерминированной; запаздывание с реакцией грозит катастрофой.

**СКРЫТИЕ ИНФОРМАЦИИ, INFORMATION HIDING:**

Процесс скрытия всех секретов объекта, которые ничего не добавляют к его существенным характеристикам; обычно скрывают структуру объекта и реализацию его методов.

**СЛОВАРЬ ДАННЫХ, DATA DICTIONARY:**

Полный перечень всех классов в системе.

**СЛОЙ, LAYER:**

Совокупность категорий классов или подсистем одного уровня абстракции.

**СЛОТ, SLOT:**

Часть состояния объекта; совокупность слотов образуют структуру объекта. Термины «поле», «переменная экземпляра», «объект-член» и «слот» означают одно и то же.

**СОБЫТИЕ, EVENT:**

Что-то, что может изменить состояние системы.

**СООБЩЕНИЕ, MESSAGE:**

Операция, которую один объект может выполнять над другим. Термины «сообщение», «метод» и «операция» обычно взаимозаменяемы.

**СОСТАВНОЙ ОБЪЕКТ (АГРЕГАТ), AGGREGATE OBJECT:**

Объект, состоящий из других объектов (его частей).

**СОСТОЯНИЕ, STATE:**

Совокупный результат поведения объекта: одно из стабильных условий, в которых объект может существовать, охарактеризованных количественно; в любой конкретный момент времени состояние объекта включает в себя перечень (обычно, статический) свойств объекта и текущие значения (обычно, динамические) этих свойств.

**СОТРУДНИЧЕСТВО, COLLABORATION:**

Процесс, в котором несколько объектов сотрудничают для обеспечения требуемого поведения верхнего уровня.

**СОХРАНЯЕМОСТЬ, PERSISTENCE:**

Способность объекта существовать во времени, переживая породивший его процесс, и (или) в пространстве, перемещаясь из одного адресного пространства в другое.

**СРЕДА РАЗРАБОТКИ, FRAMEWORK:**

Набор классов, предоставляющих некоторые базовые услуги в определенной области. Таким образом, среда разработки экспортирует классы и механизмы, которые клиенты могут использовать или адаптировать.

**СТАТИЧЕСКОЕ СВЯЗЫВАНИЕ, STATIC BINDING:**

Связывание означает установление соответствия имени (например, объявленной переменной) классу. Статическое связывание происходит при объявлении имени (во время компиляции), до того, как объект будет создан.

**СТРАЖ, GUARD:**

Логическое выражение, применяемое к событию; если выражение истинно, то событие происходит и система изменяет состояние.

**СТРАТЕГИЧЕСКОЕ ПРОЕКТНОЕ РЕШЕНИЕ, STRATEGIC DESIGN DECISION:**

Проектные решения, которые имеют решающее влияние на архитектуру.

**СТРУКТУРА КЛАССОВ, CLASS STRUCTURE:**

Граф, вершины которого соответствуют классам, а ребра - отношениям классов. Структура классов для конкретной системы представляется в виде совокупности диаграмм классов.

**СТРУКТУРА ОБЪЕКТОВ, OBJECT STRUCTURE:**

Граф, вершины которого соответствуют объектам, а ребра - отношениям объектов. Для отражения структуры объектов или ее части используются диаграммы объектов.

**СТРУКТУРА, STRUCTURE:**

Конкретное представление состояния объекта. Каждый объект имеет собственное состояние, независимое от других объектов, хотя все объекты одного класса имеют одинаковое представление состояния.

**СТРУКТУРНОЕ ПРОЕКТИРОВАНИЕ, STRUCTURED DESIGN:**

Метод проектирования, основанный на алгоритмической декомпозиции.

**СУПЕРКЛАСС, SUPERCLASS:**

Класс, которому наследуют другие классы (называемые непосредственными подклассами).

**СЦЕНАРИЙ, SCENARIO:**

Последовательность событий, выражающая некий аспект поведения системы.

**ТАКТИЧЕСКОЕ ПРОЕКТНОЕ РЕШЕНИЕ, TACTICAL DESIGN DECISION:**

Проектное решение, имеющее ограниченное значение для архитектуры.

**ТИП, TYPE:**

Определение области допустимых значений, которые может принимать объект, и множества операций, которые могут выполняться над объектом. Термины «класс» и «тип» обычно (но не всегда) взаимозаменяемы; тип отличается от класса тем, что фокусируется на поддержке общего протокола.

**ТИПИЗАЦИЯ, TYPING:**

Механизмы, препятствующие замене объектов одного типа на другой или, в крайнем случае, жестко ограничивающие такую замену.

**ТРАНСФОРМАЦИОННАЯ СИСТЕМА, TRANSFORMATIONAL SYSTEM:**

Система, поведение которой определяется в терминах отображения «вход-выход».

**УПРАВЛЕНИЕ ДОСТУПОМ, ACCESS CONTROL:**

Механизм доступа к данным и операциям класса. В C++ открытые элементы доступны всем, защищенные элементы доступны подклассам, так называемым друзьям класса и файлам реализации, закрытые элементы доступны реализации и друзьям класса. Наконец, элементы с доступом на уровне реализации доступны только в файле реализации класса.

**УРОВЕНЬ АБСТРАКЦИИ, LEVEL OF ABSTRACTION:**

Относительное упорядочение абстракций по структурам классов, объектов, модулей или процессов. В терминах иерархии «часть/целое» объект находится на более высоком уровне абстракции, чем другие, если он строится на основе этих объектов: в терминах иерархии «общее/частное», высокоуровневые абстракции носят более обобщенный характер, чем низкоуровневые.

**УСЛУГА, SERVICE:**

Поведение, обеспечиваемое некоторой частью системы.

**УСТРОЙСТВО, DEVICE:**

Часть аппаратуры, не имеющая собственных вычислительных ресурсов.

**УТВЕРЖДЕНИЕ, ASSERTION:**

Логическое выражение некоторого условия, истинность которого необходимо обеспечить.

**УТИЛИТА КЛАССА, CLASS UTILITY:**

Совокупность свободных подпрограмм. На C++ - класс, который состоит только из статических членов и/или функций-членов.

**ФУНКЦИОНАЛЬНАЯ ТОЧКА, FUNCTION POINT:**

В контексте анализа требований к системе - отдельное поведение, видимое извне и поддающееся проверке.

**ФУНКЦИЯ, FUNCTION:**

Некоторое преобразование «вход-выход», вытекающее из поведения объекта.

**ФУНКЦИЯ-ЧЛЕН, MEMBER FUNCTION:**

Операция над объектом, определенная как часть описания класса. Все функции-члены - операции, но не все операции - функции-члены. Термины «функции-члены» и «методы» взаимозаменяемы. В некоторых языках функции-члены существуют сами по себе и могут переопределяться подклассами; в других языках функция-член не может быть переопределена, - она служит как часть реализации обобщенных или виртуальных функций, которые можно переопределять в подклассах.

**ЭКЗЕМПЛЯР, INSTANCE:**

Нечто, чем можно оперировать. Экземпляр имеет состояние, поведение и идентичность. Структура и поведение всех экземпляров класса определяются этим классом. Термины «объект» и «экземпляр» взаимозаменяемы.