Словарь терминов: объектно-ориентированный анализ (OOA) и проектирование (OOD)

##	Термин	Содержание
1	**CRC-карточки, CRC cards**	CRC - Class / Responsibilities / Collaborators, Класс / Ответственности / Сотрудники; простое, но достаточно эффективное средство мозгового штурма при выявлении ключевых абстракций и механизмов, см. пп. 37, 58
2	**абстрактный класс, abstract class**	Класс, который не может иметь экземпляров. Абстрактный класс пишется в предположении, что его конкретные подклассы дополнят его структуру и поведение, см. п. 24
3	**ассоциация, association**	Отношение, означающее некоторую смысловую связь между классами, **см. п. 51**
4	**атрибут, attribute**	Часть составного объекта (агрегата)
5	**базовый класс, base class**	Наиболее общий класс в какой-либо структуре классов. В большинстве приложений есть несколько таких корневых классов. В языке программирования C# определяется всеобщий базовый класс Object, который является суперклассом для всех остальных классов
6	**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.
7	**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
8	**делегирование, delegation**	При делегировании один объект, ответственный за операцию, передает выполнение этой операции другому объекту
9	**деструктор, destructor**	Операция класса, которая освобождает состояние объекта и/или уничтожает сам объект, **см. пп. 25, 32**
10	**диаграмма взаимодействий, interaction diagram**	Часть системы обозначений объектно-ориентированного проектирования; используется для демонстрации выполнения какого-либо сценария в контексте диаграммы объектов, **см. п. 13**
11	**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, **см. п. 23**
12	**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, **см. п. 27**
13	**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, **см. п. 32**
14	**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
15	**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
16	**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 / см. статическое связывание, п. 60 /
17	**закрытая часть, private**	Часть интерфейса какого-либо класса, объекта или модуля, закрытая (невидимая) для других классов, объектов и модулей, см. пп. 18, 38
18	**защищенная часть, protected**	Часть интерфейса какого-либо класса, объекта или модуля, невидимая для всех других классов, объектов и модулей за исключением подклассов, см. пп. 17, 38
19	**иерархия, hierarchy**	Подчинение или упорядочение абстракций. Две типичных иерархии в сложной системе - структура классов (включая иерархию "общее / частное" - наследование) и структура объектов (включая иерархию "целое / часть" - ассоциация); иерархии можно также обнаружить в

		архитектурах модулей и процессов
20	**инвариант,** **invariant**	Логическое выражение некоторого условия, истинность которого необходимо соблюдать
21	**инкапсуляция,** **encapsulation**	Процесс разделения элементов абстракции, которые образуют ее структуру и поведение. Служит для отделения внешних обязательств объекта от его реализации, см. пп. 31, 43
22	**интерфейс**	Внешний вид класса, объекта или модуля, выделяющий его существенные черты и не показывающий внутреннего устройства и секретов поведения, см. п. 69
23	**класс, см. п. 65** **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, см. пп. 2, 5, 24
24	**конкретный класс,** **concrete class**	Класс, реализация которого завершена и который, поэтому, может иметь экземпляры, см. п. 2
25	**конструктор,** **constructor**	Операция, создающая объект и / или инициализирующая его состояние , см. пп. 9, 32
26	**метод,** **method**	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, см. пп. 57, 67, **53**
27	**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, см. пп. 22, 50
28	**модульность,** **modularity**	Свойство системы, которая была разделена на связные и слабо зацепленные между собой модули, см. п. 27
29	**мономорфизм,** **monomorphism**	Положение теории типов, согласно которому имена (например, переменных) могут обозначать только объекты одного и того же класса см. п. 43
30	**мощность,** **cardinality**	Число экземпляров класса: число экземпляров, участвующих в связи классов, см. п. 68
31	**наследование,** **inheritance**	Отношение между классами, при котором класс использует структуру или поведение другого (одиночное наследование) или других (множественное наследование, в C# не поддерживается, см. п. 69) 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, см. пп. 21, 43
32	**объект,** **object**	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, см. п. 68
33	**объектная модель,** **object model**	Совокупность основополагающих принципов, лежащих в основе объектно-ориентированного проектирования; парадигма программирования, основанная на принципах абстрагирования, инкапсуляции, модульности, иерархичности, типизации, параллелизма и устойчивости
34	**объектно-ориентированное программирование,** **object-oriented programming (OOP)**	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, см. пп. 16, 43
35	**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 **, см. пп. 10 – 15, сравнить с п. 63**
36	**объектно-ориентированный анализ,** **object-oriented analysis (OOA)**	Метод анализа, согласно которому требования рассматриваются с точки зрения классов и объектов, составляющих **словарь** предметной области, см. пп. 23, 32
37	**ответственность,** **responsibility**	Поведение, за которое ответственен объект, см. п. 32
38	**открытая часть,**	Часть интерфейса какого-либо класса, объекта или модуля, открытая (видимая) для всех

	public	классов, объектов и модулей, см. пп. 17, 18, 22
39	**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 (static), см. п. 40
40	**переменная экземпляра,** **instance variable**	Часть состояния **объекта**. В совокупности переменные экземпляра полностью определяют структуру объекта. Термины "переменная экземпляра", "поле", "объект-член" и "слот" взаимозаменяемы, см. п. 40
41	**поведение,** **behavior**	Действия и реакции объекта, выраженные в терминах передачи сообщений и изменения состояния; видимая извне и воспроизводимая активность объекта, см. п. 32
42	**поле,** **field**	Часть состояния объекта; совокупность полей объекта образуют его структуру. Термины "поле", "переменная экземпляра", "объект-член" и "слот" означают одно и то же
43	**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, см. пп. 21, 31, **29**
44	**постусловие,** **postcondition**	Инвариант, соблюдаемый на выходе из операции, см. п. 20, 46
45	**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
46	**предусловие,** **precondition**	Инвариант, предполагаемый на входе в операцию, см. п. 20 , 44
47	**протокол,** **protocol**	Способы, которыми объекты могут действовать и реагировать; полное статическое и динамическое представление объекта; протокол объекта определяет допустимое поведение объекта
48	**процесс,** **process**	Запуск одного потока управления
49	**процессор,** **processor**	Часть аппаратного обеспечения, имеющая вычислительные ресурсы
50	**реализация,** **implementation**	Внутреннее представление класса, объекта или модуля, включая секреты его поведения
51	**связь,** **link**	Связь между объектами, экземпляр ассоциации, **см. пп. 3, 32**
52	**сервер,** **server**	Объект, который никогда не воздействует на другие объекты, но используется ими; объект, предоставляющий некоторые услуги
53	**сигнатура,** **signature**	Имя метода со списком его параметров, см. п. 26
54	**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
55	**словарь данных,** **data dictionary**	Полный перечень всех классов в системе
56	**событие,** **event**	Что-то, что может изменить состояние системы, z.B., стандартное событи **Click**
57	**сообщение,** **message**	Операция, которую один объект может выполнять над другим. Термины "сообщение", "метод" и "операция" обычно взаимозаменяемы, см. п. 26
58	**сотрудничество,** **collaboration**	Процесс, в котором несколько объектов сотрудничают для обеспечения требуемого поведения верхнего уровня
59	**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
60	**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. /см. динамическое связывание, п. 16/
61	**структура классов,** **class structure**	Граф, вершины которого соответствуют классам, а ребра - отношениям классов. Структура классов для конкретной системы представляется в виде совокупности диаграмм классов,

		см. п. 23
62	**структура объектов,** **object structure**	Граф, вершины которого соответствуют объектам, а ребра - отношениям объектов. Для отражения структуры объектов или ее части используются диаграммы объектов, см. п. 32
63	**структурное проектирование,** **structured design**	Метод проектирования, основанный на алгоритмической декомпозиции**, сравнить с п. 35**
64	**сценарий,** **scenario**	Последовательность шагов, реализующих прецедент. Прецедент – это случай (факт) взаимодействия пользователя системы с самой системой
65	**тип, , см. п. 23** **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 **см. п. 47**
66	**уровень абстракции,** **level of abstraction**	Относительное упорядочение абстракций по структурам классов, объектов, модулей или процессов. В терминах иерархии "часть / целое" (ассоциация) объект находится на более высоком уровне абстракции, чем другие, если он строится на основе этих объектов: в терминах иерархии "общее / частное" (наследование), высокоуровневые абстракции носят более обобщенный характер, чем низкоуровневые
67	**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, см. п. 26
68	**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, см. п. 32
69	**interface**	Interface синтаксически подобен абстрактным классам. Interface может определять сигнатуры методов, свойств, индексаторов и событий. Interface в принципе **не предусматривает** какой-либо реализации. Interface определяет **что** должно быть сделано, но не уточняет **как**. Interface может реализовать любое количество классов. Одни класс может реализовать любое количество interface, см. п. 22