|  |  |
| --- | --- |
| **Abstract Window Toolkit - AWT** | Стандартный пакет графических компонент, реализованных с использованием специфических платформенных методов. Данные компоненты поддерживают лишь то подмножество функциональных возможностей, которое присуще всем платформам. В значительной степени вытеснен набором компонент Project Swing (см. Swing-компоненты). |
| **abstract** | Ключевое слово языка программирования Java, используемое в определении класса для указания на невозможность создания его экземпляров, но при этом доступного для наследования другими классами. Абстрактный класс может содержать нереализованные (абстрактные) методы, которые должны быть реализованы в его подклассах. |
| **абстрактный класс (abstract class)** | Класс, который содержит один или более абстрактных методов, вследствие чего нельзя создавать экземпляры данного класса. Абстрактные классы определены таким образом, чтобы другие классы могли расширять и конкретизировать их, реализуя абстрактные методы. |
| **абстрактный метод (abstract method)** | Метод, не имеющий реализации. |
| **контроль доступа (access control)** | Технические средства, посредством которых ограничиваются множества пользователей или программ, взаимодействующих с ресурсами, с целью повышения целостности, конфиденциальности и доступности. |
| **АЦИН (ACID - Atomicity, Consistency, Isolation and Durability)** | Акроним четырех свойств, обеспечиваемых транзакциями: атомарность, целостность, изоляция и надежность. |
| **активация (activation)** | Процесс передачи корпоративных компонент (enterprise beans) из вторичного устройства хранения данных в память. |
| **список фактических параметров (actual parameter list)** | Аргументы, определенные в вызове метода (см. список формальных параметров). |
| **альфа-фактор (alpha value)** | Значение, указывающее яркость (или интенсивность) пикселя. |
| **интерфейс прикладного программирования (API - Application Programming Interface)** | Спецификация, предназначенная для пользователей и описывающая методы доступа к свойствам и состоянию объектов и классов. |
| **апплет (applet)** | Компонент, который обычно исполняется в Web-браузере или в любой другой программе просмотра апплетов. |
| **контейнер апплетов (applet container)** | Контейнер, включающий в себя поддержку модели программирования апплетов. |
| **устройства (appliances)** | Сетевые устройства, такие как принтеры, терминалы с поддержкой технологии JavaTM и клиенты, управляемые посредством использования Java Management API (JMAPI). |
| **компоновщик приложения (application assembler)** | Человек, объединяющий компоненты и модули в большие модули разработки. |
| **клиентское приложение (application client)** | Клиентская программа первого уровня, исполняемая на собственной виртуальной Java-машине. |
| **контейнер клиентского приложения (application client container)** | Контейнер, поддерживающий клиентские приложения и обеспечивающий интегрированное представление API платформы J2EE. |
| **модуль клиентского приложения (application client module)** | Программный модуль, состоящий из одного или более классов и описания клиентского приложения. |
| **поставщик программных компонент (Application Component Provider)** | Поставщик, который предоставляет классы Java, реализующие методы компонентов, описания JSP-страниц и необходимые дескрипторы. |
| **модель программирования приложения (Application Programming Model - APM)** | Модель прикладного программирования, которая определяет, как использовать и объединять возможности платформы J2EE для создания прикладных решений в предметной области предприятия. |
| **аргумент (argument)** | Элемент данных, указанный в вызове метода. Аргумент может быть константой, переменной или выражением. |
| **массив (array)** | Совокупность элементов данных одного типа, в которой позиция каждого элемента однозначно определена целым числом (индексом массива). |
| **Американский Стандартный Код Обмена Информацией (American standard code for information interchange - ASCII)** | Стандартно, на один символ отводится 7 бит. См. также Уникод. |
| **атомарный (atomic)** | Операция, выполняемая как единый неделимый акт. |
| **аутентификация (authentication)** | Процесс, посредством которого один объект показывает другому, что он действует от имени определенной идентификационной записи. Платформа J2EE нуждается в трех видах аутентификации: обычной (basic), связанной с формой (form-based) и взаимной (mutual). |
| **авторизация (authorization)** | См. управление доступом. |
| **ограничение авторизации (authorization constraint)** | Набор ролевых имен, предназначенных для защиты и разрешающих доступ к Web-ресурсам. |
| **базовая (открытая) аутентификация (basic authentication)** | Метод проверки Web-сервером имени пользователя и пароля, полученных при помощи механизма аутентификации, встроенного в Web-клиент. |
| **bean-компонент (bean)** | Программный компонент многократного использования. Bean-компоненты могут быть объединены для создания приложения. |
| **механизм сохранения, управляемый bean-компонентом (bean-managed persistence)** | Механизм, при котором передача данных между переменными экземпляров bean-компонента и основным администратором ресурсов управляется bean-компонентом. |
| **транзакция, управляемая bean-компонентом (bean-managed transaction)** | Корпоративный компонент (enterprise bean) определяет границы транзакции. |
| **бинарный оператор (binary operator)** | Знак операции, имеющий два аргумента. |
| **бит (bit)** | Минимальная единица информации в компьютере. Может принимать значения 0 или 1. |
| **побитовый оператор (bitwise operator)** | Знак операции, воздействующий на операнды, как на набор битов (0 и 1). Например, бинарные логические операции (&, |, ^), бинарные операции сдвига (<<, >>, >>>) и унарная операция дополнения (~). |
| **блок (block)** | Любой код на языке программирования JavaTM, заключенный между двумя фигурными скобками. Например, {x = 1;}. |
| **булевский (boolean)** | Относится к выражению или переменной, которые могут принимать только два значения: true ("истина") и false ("ложь"). В языке программирования JavaTM существует тип boolean, а также литеральные константы true и false. |
| **ограничительная область (bounding box)** | Прямоугольник наименьшего размера, содержащий указанную геометрическую фигуру. Для растровых объектов включает все заданные пиксели. |
| **break** | Ключевое слово языка программирования JavaTM. Оператор break передаёт управление строке, следующей за текущей структурой. Если за break следует метка, то программа продолжает исполнение, начиная с "помеченного" оператора. |
| **бизнес-логика (business logic)** | Код, реализующий функциональность приложения. В модели Enterprise Java Beans эта логика реализуется при помощи методов корпоративного компонента (enterprise bean). |
| **бизнес-метод (business-method)** | Метод корпоративного компонента (enterprise bean), реализующий бизнес-логику или правила приложения. |
| **байт (byte)** | Последовательность из восьми битов. В языке программирования JavaTM определен соответствующий тип byte. |
| **байт-код (bytecode)** | Машинно-независимый код, генерируемый Java-компилятором и исполняемый Java-интерпретатором. |
| **методы обратной связи (callback methods)** | Метод компонента, вызываемый контейнером для уведомления компонента о важных событиях во время его жизненного цикла. |
| **вызывающий оператор (caller)** | См. администратор вызывающего оператора |
| **администратор вызывающего оператора (caller principal)** | Администратор, который идентифицирует объект, вызывающий метод корпоративного компонента (enterprise bean). |
| **case** | Ключевое слово языка программирования Java (оператор switch), которое определяет набор инструкций, исполняющихся в случае, если значение переключающего выражения совпадает со значением константы (разметки выбирающего предложения), указанной в конструкции case. |
| **преобразование типа (casting)** | Явное преобразование одного типа данных в другой. |
| **catch** | Ключевое слово языка программирования JavaTM, используемое для объявления блока инструкций, которые должны быть выполнены в случае исключительной ситуации или ошибки выполнения, возникающей в предшествующем блоке try. |
| **char** | Ключевое слово языка программирования JavaTM, используемое для объявления переменной символьного типа. |
| **класс (class)** | Тип в языке программирования JavaTM, определяющий реализацию особого вида объекта. Описание класса определяет экземпляр класса, его переменные и методы. Так же определяются интерфейсы и суперклассы. По умолчанию суперклассом любого класса является Object. |
| **метод класса (типа) (class method)** | Метод, который вызывается безотносительно ссылки на конкретный объект. Методы класса влияют на класс в целом, а не на конкретный его экземпляр. Также носит название статического метода. См. также метод экземпляра. |
| **путь к классам (classpath)** | Переменная среды окружения, которая сообщает виртуальной машине JavaTM и приложениям Java (например, утилитам, расположенным в директории JDKTM 1.1.X\bin), где находятся библиотеки классов, включая пользовательские библиотеки. Свойство виртуальной машины (JVM), которое может быть задано с помощью переменной окружения, либо другими способами, например, с помощью опций JVM. |
| **переменная класса (типа) (class variable)** | Переменная, относящаяся к некоторому классу в целом, а не к отдельным экземплярам класса. Переменные класса являются элементами определения класса. Также носит название статического поля. См. также переменная экземпляра. |
| **клиент (client)** | В модели соединения "клиент-сервер" клиент - это процесс, который удаленно обращается к ресурсам вычислительного сервера. |
| **адрес основного кода (codebase)** | При работе с атрибутом code в тэге APPLET указывает полный путь к файлу главного класса апплета: code определяет имя файла, а codebase - URL директории, содержащей файл. |
| **комментарий (comment)** | Поясняющий текст в программе, который игнорируется компилятором. В приложениях JavaTM комментарии выделяются символами // или /\*…\*/. |
| **фиксация (транзакции) (commit)** | Момент транзакции, когда все изменения, проведенные в процессе транзакции, фиксируются в базе данных. |
| **единица компиляции (compilation unit)** | Минимальная единица исходного кода, которая может быть откомпилирована. В текущей реализации JavaTM единица компиляции - это последовательность определений интерфейсов и классов, которой могут предшествовать объявление пакета и объявления импорта. |
| **компилятор (compiler)** | Программа, транслирующая исходный код приложения в код, исполняемый компьютером. JavaTM-компилятор транслирует исходный код, написанный на языке Java, в машинно-независимый код (байт-код) для виртуальной машины Java. См. также интерпретатор. |
| **компонент (component)** | Программный модуль, поддерживаемый контейнером. Компоненты конфигурируемы на стадии разработки. Платформа J2EE определяет четыре вида компонент: корпоративные компоненты (enterprise beans), Web-компоненты, апплеты и клиентские приложения. |
| **контракт компонента (component contract)** | Набор условий, регулирующих отношения между компонентом и его контейнером. Контракт включает: управление жизненным циклом компонента, контекстный интерфейс, используемый образцом компоненты для получения информации о контейнере или использования возможностей контейнера, и списка функциональных возможностей, которые должен поддерживать каждый контейнер для данной компоненты. |
| **окружение компонента (component environment)** | Набор требований, определяемых Поставщиком Программных Компонент (Application Component Provider), которые должны быть доступны компоненту J2EE. Записи окружения декларативно определены в описании компонента. Каждый компонент указывает и получает доступ к значениям конфигурации компонента, используя контекст java:comp/env JNDI. Эти значения могут быть объектами, от которых зависит компонент, такими как JDBC DataSource или простыми значениями, такими как налоговая ставка. |
| **компоновка (compositing)** | Процесс наложения одного изображения на другое, с целью получения единого изображения. |
| **соединение (connection)** | См. менеджер соединений. |
| **мастер соединения (connection factory)** | См. менеджер ресурсов мастера соединения. |
| **коннектор (connector)** | Стандартный механизм расширения контейнеров для обеспечения взаимодействия с управленческими информационными системами (Executive Information Systems - EISs). Коннектор специфичен для каждой EIS и состоит из адаптера ресурсов и средств разработки приложений для взаимодействия с EIS. Адаптер ресурсов подключен к контейнеру посредством контрактов системного уровня, определенных в архитектуре коннектора. |
| **архитектура коннектора (connector architecture)** | Архитектура, предназначенная для интеграции серверов J2EE с управленческими информационными системами (Executive Information Systems - EISs). Архитектура состоит из двух частей: адаптер ресурсов производителя EIS и сервер J2EE, поддерживающий этот адаптер. Данная архитектура определяет набор контрактов, которые должны поддерживаться адаптером ресурсов для подключения к J2EE-серверу. Например, транзакции, обеспечение безопасности, управление ресурсами. |
| **конструктор (constructor)** | Метод особого рода, создающий объект и инициализирующий его поля. В языке программирования Java имя конструктора совпадает с именем класса. Конструкторы вызываются системой при создании экземпляра объекта (исполнении конструкции new). |
| **const** | Зарезервированное ключевое слово языка JavaTM. Однако, не используется текущими версиями Java. |
| **контейнер (container)** | Сущность, обеспечивающая управление, безопасность, разработку и сервисы выполнения компонент. Кроме того, каждый тип контейнера (EJB, Web, JSP, сервлет, апплет или приложение-клиент) также предоставляет свои специфические сервисы. |
| **персистенция (сохраняемость), управляемая контейнером (container-managed persistence)** | Механизм, при котором передача данных между экземпляром корпоративной компоненты и менеджером расположенных ниже ресурсов управляется контейнером корпоративной компоненты (enterprise bean). |
| **транзакция, управляемая контейнером (container-managed transaction)** | Транзакция, границы которой определяются EJB-контейнером. Экземпляр корпоративной компоненты (enterprise bean) должен использовать транзакции, управляемые контейнерами. |
| **контекстный атрибут (context attribute)** | Объект, встроенный в контекст, ассоциированный с сервлетом. |
| **continue** | Ключевое слово языка программирования Java, которое обозначает оператор, завершающий текущую итерацию цикла и, если условие повторения выполняется, начинающий исполнение следующей. Если за данным ключевым словом следует метка, continue возобновляет исполнение, начиная с помеченного оператора (то есть начинается следующая итерация цикла, помеченного данной меткой). |
| **диалоговый режим (conversational state)** | Значения полей сессии bean-компонента плюс транзитивное замыкание объектов, доступных из полей bean-компонента. Транзитивное замыкание bean-компонента определяется в терминах протокола сериализации языка Java, которые сохранены посредством сериализации экземпляра bean-компонента. |
| **Common Object Request Broker Architecture - CORBA** | Технология построения распределенных объектных приложений, специфицируемая группой по развитию стандартов объектного программирования (Object Management Group - OMG). |
| **базовый класс (core class)** | Публичный класс (или интерфейс), являющийся стандартным членом JavaTM Platform. Обязательным свойством таких классов является их доступность в любой операционной системе, поддерживающей среду Java. Программой, "полностью" написанной на языке Java, называется программа, использующая только такие классы, и, следовательно, обладающая свойством независимости от платформы. См. также 100% Pure JavaTM. |
| **базовые пакеты (core packages)** | Необходимый набор программных интерфейсов приложений (Application Programming Interfaces - APIs) платформы Java, который поддерживается в любой реализации. |
| **метод create (create method)** | Метод, определенный в "домашнем" интерфейсе и вызываемый клиентом, для создания корпоративной компоненты (enterprise bean). |
| **полномочия (credentials)** | Учетная запись с параметрами доступа пользователя, сформированными после его успешной аутентификации. |
| **критическая секция (critical section)** | Фрагмент программы, в котором поток выполняет действия над общим ресурсом, доступ к которому должен быть монопольным. |
| **пакет проверки совместимости (Compatibility Test Suite - CTS)** | Набор программ, предназначенных для проверки соответствия продуктов J2EE спецификации платформы J2EE. |
| **объявление (declaration)** | Выражение, связывающее идентификатор с атрибутами (типом). При необходимости осуществляется выделение памяти (для данных) или выполнение (для методов). См. также описание. |
| **default** | Ключевое слово языка программирования JavaTM, при необходимости используемое после всех условий case в блоке switch. Если ни одно из проверяемых значений не совпадает со значением оператора switch, то выполняются инструкции, следующие после ключевого слова default. |
| **описание (definition)** | Объявление, осуществляющее резервирование памяти (для данных) или выполнение (для методов). См. также объявление. |
| **делегирование (delegation)** | Передача (делегирование) функций - способность объекта или потока внутри объекта выполнять под именем клиента запросы к другим удаленным объектам. |
| **администратор размещения (deployer)** | Человек, устанавливающий модули и приложения J2EE в операционной системе. |
| **размещение (deployment)** | Процесс установки программного обеспечения в операционную среду. |
| **дескриптор размещения (deployment descriptor)** | XML-файл, поставляемый с каждым модулем и приложением и описывающий процесс их установки. Дескриптор размещения управляет инструментами для установки модуля или приложения с какими-либо специфичными опциями контейнера, а также описывает особые требования, предъявляемые к конфигурации. |
| **антирекомендация (deprecation)** | Относится к классам, интерфейсам, конструкторам, методам или полям, которые рекомендуется больше не использовать, и которые могут быть исключены из последующих версий. |
| **"является потомком" (derived from)** | Класс X "является потомком" класса Y, если класс X расширяет (наследует) класс Y. См. также подкласс, суперкласс. |
| **распределенный (distributed)** | Работающий более чем в одном адресном пространстве. |
| **распределенное приложение (distributed application)** | Приложение, которое составлено из различных компонент, выполняющихся в различных средах, обычно, на разных платформах, соединенных посредством сети. Стандартные распределенные приложения: двухзвенное (клиент/сервер), трехзвенное (клиент/промежуточное программное обеспечение/сервер), n-звенное (клиент/множественное промежуточное программное обеспечение/сервер). |
| **do** | Ключевое слово языка программирования JavaTM, используемое для объявления цикла, повторяющего блок инструкций. Условие выхода из цикла определяется ключевым словом while в конце итерации. |
| **объектная модель документа (Document Object Model - DOM)** | Дерево объектов и интерфейсы для реализации обхода вершин дерева и написания его XML-версии согласно спецификации W3C. |
| **double** | Ключевое слово языка программирования JavaTM, используемое для определения переменной типа double. |
| **двойная точность (double precision)** | Согласно спецификации языка программирования JavaTM, число с плавающей точкой, занимающее 64 бита данных. См. также одинарная точность. |
| **определение типа документа (Document Type Definition - DTD)** | Описание структуры и свойств XML-файлов. |
| **EJB-контейнер (EJB-container)** | Контейнер, содержащий корпоративные компоненты (EJB). |
| **поставщик EJB-контейнера (EJB Container Provider)** | Производитель, который предоставляет EJB-контейнер. |
| **EJB-контекст (EJB-context)** | Объект, позволяющий корпоративному компоненту (enterprise bean) запускать сервисы контейнера и получать информацию о программе, вызвавшей клиентский метод. |
| **"домашний" объект EJB (EJB home object)** | Объект, обеспечивающий жизненный цикл операций (создания, удаления, поиска) для корпоративного компонента (enterprise bean). Класс для "домашнего" объекта EJB порожден инструментами разработки контейнера. "Домашний" объект EJB реализует собственный интерфейс корпоративного компонента (enterprise bean). Клиент ссылается на "домашний" объект EJB, чтобы выполнить операции жизненного цикла EJB-объекта. Для определения местонахождения EJB home object клиент использует JNDI интерфейс. |
| **EJB-.jar файл (EJB .jar file)** | Архив JAR, который содержит EJB-модуль. |
| **EJB-модуль (EJB module)** | Программный модуль, который состоит из одного или более корпоративных компонентов (enterprise beans) и дескриптора установки EJB. |
| **EJB-объект (EJB object)** | Объект, чей класс реализуется удаленным интерфейсом корпоративного компонента (enterprise bean). Клиент никогда не ссылается непосредственно на экземпляр корпоративного компонента, а только на объект EJB. Класс для объекта EJB порожден инструментами разработки, входящими в состав контейнера. |
| **EJB-сервер (EJB-server)** | Программное обеспечение, предоставляющее службы EJB-контейнеру. Например, EJB-контейнер обычно зависит от менеджера транзакций, который является частью EJB-сервера, предназначенной для двухфазного завершения распределенных транзакций параллельно со всеми участвующими в ней менеджерами ресурсов. Архитектура J2EE предполагает, что EJB-сервер, содержащий EJB-контейнер, поставляется тем же производителем, поэтому не существует соглашения между двумя этими сущностями. EJB-сервер может содержать один и более контейнеров EJB. |
| **поставщик EJB-сервера (EJB Server Provider)** | Производитель, который поставляет EJB-сервер. |
| **ресурс управленческой информационной системы (ресурс УИС) (EIS resource)** | Сущность, предоставляющая функциональные возможности УИС клиентам системы. Например, запись или множество записей в системе базы данных, бизнес-объект в ERP-системе (системе, предназначенная для планирования и управления ресурсами предприятий), программа транзакции в системе обработки транзакций. |
| **else** | Ключевое слово языка программирования JavaTM, используемое для выполнения блока инструкций в том случае, если проверка условия в операторе if дала отрицательный результат (false). |
| **технология EmbeddedJavaTM (Embedded JavaTM technology)** | Технология, которая впервые появилась в рамках Java 2 Platform. Распространение этой технологии ограничено лицензионным соглашением, которое позволяет владельцу лицензии использовать определенные Java-технологии для создания и внедрения встроенных приложений. |
| **инкапсуляция (encapsulation)** | Локализация (упрятывание) части данных в пределах класса. Поскольку объекты инкапсулируют данные и реализацию, пользователь может рассматривать объект как черный ящик, предоставляющий услуги. Переменные и методы экземпляров класса могут добавляться, удаляться или изменяться, но до тех пор, пока услуги, предоставляемые объектом, не изменяются, нет необходимости переписывать код, использующий данный объект. См. также переменная экземпляра класса и метод экземпляра класса. |
| **корпоративный компонент (enterprise bean)** | Компонент, реализующий бизнес-задачу или бизнес-объект, либо сущность или сессионный компонент. |
| **управленческая информационная система - УИС (Enterprise Information System - EIS)** | Приложения, которые содержат существующую систему управления информацией всей компании. Эти приложения обеспечивают информационную инфраструктуру предприятия. УИС предлагает определенный набор услуг своим клиентам. Эти услуги представлены клиентам как локальные и/или удаленные интерфейсы. Примеры УИС: ERP-система (система, предназначенная для планирования и управления ресурсами предприятий), универсальная система обработки транзакций, существующая система баз данных. |
| **поставщик корпоративных компонент (Enterprise Bean Provider)** | Прикладной программист, разрабатывающий классы корпоративных компонентов (enterprise beans), удаленные и локальные интерфейсы, файлы дескрипторов установки и упаковывает их в EJB-.jar файл. |
| **Enterprise JavaBeansTM - EJB** | Архитектура, предназначенная для развития и установки объектно-ориентированных, распределенных, корпоративных приложений. Приложения, написанные с использованием архитектуры Enterprise JavaBeansTM, являются масштабируемыми, многопользовательскими и безопасными. |
| **компонент управления данными (entity bean)** | Корпоративный компонент (enterprise bean), который отображает постоянные данные, хранящиеся в базе данных. Сущность идентифицируется первичным ключом. Если контейнер, которому принадлежит сущность, разрушается, сущность, ее первичный ключ и все удаленные ссылки сохраняются. |
| **ebXML** | ebXML основан на стандарте XML (Extensive Markup Language) и предназначен для поставки модульного набора спецификаций, позволяющего предприятиям различного масштаба и географического положения вести дела посредством Internet. ebXML снабжает компании и организации стандартным методом обмена деловой информацией, ведения торговых отношений, передачи данных, а также определения и регистрации бизнес процессов. |
| **исключительная ситуация (exception)** | Ситуация, возникающая во время работы программы вследствие неудачного выполнения операции, или запрограммированная с использованием оператора throw. Если в программе не предусмотрена реакция на возникшую ситуацию, дальнейшее исполнение программы невозможно. Язык программирования JavaTM поддерживает обработку исключительных ситуаций при помощи ключевых слов try, catch и throw. См. также обработчик исключительных ситуаций. |
| **обработчик исключительных ситуаций (exception handler)** | Блок кода, реагирующий на определенный тип исключительных ситуаций. Если исключительная ситуация произошла вследствие ошибки, после которой программа может возобновить работу, то программа продолжает выполнение после обработчика исключительных ситуаций. |
| **исполняемый код (executable context)** | Программа, которая выполняется из HTML-файла. См. также апплет. |
| **"расширяет" (extends)** | Класс X может "расширять" класс Y с целью добавления функциональности (при помощи добавления полей и методов классу Y или переопределения методов класса Y). В таком случае говорят, что класс X является подклассом класса Y. Один интерфейс "расширяет" другой при помощи добавления методов. См. также "произошел из". |
| **метод поиска (finder method)** | Метод, определенный в "домашнем" интерфейсе и вызываемый клиентом для нахождения сущности. |
| **первая поставка пользователям (First Customer Ship - FCS)** | Дата выпуска/передачи продукта пользователю. |
| **поле (field)** | Элемент класса. Пока не определено обратное, поле не является статическим. |
| **терминальный - final** | Ключевое слово языка программирования JavaTM. Объявление класса с модификатором final означает, что класс нельзя расширить или наследовать. Final-методы не могут переопределяться, final-переменные не могут изменять свое начальное значение. |
| **finally** | Ключевое слово языка программирования JavaTM, задающее блок операций, которые будут выполнены независимо от того, произошла исключительная ситуация или ошибка в блоке, определенном с ключевым словом try, или нет. |
| **float** | Ключевое слово языка программирования JavaTM, используемое для определения числа с плавающей точкой. |
| **for** | Ключевое слово языка программирования JavaTM, используемое для объявления цикла, повторяющего блок инструкций. Программист определяет выполняемые операции, условия выхода из цикла, а также некоторые начальные действия перед началом первой итерации (инициализацию служебных переменных и пр.). |
| **аутентификация, связанная с формой (form-based authentication)** | Аутентификация, при которой Web-сервер выдает специализированную форму для входа в систему. |
| **FTP - протокол (File Transfer Protocol - FTP)** | Internet-протокол, основанный на TCP\IP. Предназначен для передачи файлов между хост-компьютерами. См. также TCP\IP. |
| **список формальных параметров (formal parameter list)** | Параметры, заданные в описании метода. См. также список фактических параметров. |
| **"сборка мусора" (garbage collection)** | Автоматическое обнаружение и освобождение памяти, которая больше не используется. Система управления и слежения за правильностью выполнения программы (JavaTMruntime system) выполняет "сборку мусора" таким образом, чтобы программисту не требовалось напрямую вызывать методы освобождения памяти. При необходимости программист может запустить сборщик мусора явно, путем вызова специального метода. В этом случае сборка мусора будет выполняться синхронно. |
| **goto** | Зарезервированное слово языка программирования JavaTM. Однако, в текущих версиях языка не используется. |
| **группа (group)** | Совокупность пользователей в пределах данного домена политики безопасности. |
| **графический интерфейс пользователя (Graphical User Interface - GUI)** | Графический интерфейс, предназначенный для удобства использования некоторых программ. |
| **манипулятор (handle)** | Объект, используемый для уникальной идентификации корпоративного компонента (enterprise bean). Клиент может сериализовать манипулятор, а затем десериализовать его для получения ссылки на корпоративный компонент (enterprise bean). |
| **шестнадцатеричный (hexadecimal)** | Система счисления с основанием 16. Знаки 0-9 и a-f представляют цифры от 0 до 15. В программе, написанной на языке JavaTM, шестнадцатеричные числа должны начинаться с символов 0x. См. также восьмеричный. |
| **иерархия (hierarchy)** | Классификация соотношений, в которой каждый элемент, кроме верхнего (называемого корнем), является специализированным видом элемента, расположенного над ним. Каждый элемент может иметь один или несколько элементов, находящихся ниже него в иерархии. В иерархии классов JavaTM, образуемой при наследовании, корнем является класс Object. |
| **"домашний" интерфейс (home interface)** | Один из двух интерфейсов для корпоративного компонента (enterprise bean). "Домашний" интерфейс может определять несколько методов для создания и удаления корпоративного компонента (enterprise bean). Для сессионных компонентов (session beans) "домашний" интерфейс определяет методы создания и удаления, тогда как для сущностей (entity beans) - методы создания, нахождения и удаления. |
| **"домашний" манипулятор (home handle)** | Объект, используемый для получения ссылки на "домашний" интерфейс. "Домашний" манипулятор может быть сериализован и записан в ПЗУ, а также десериализован для получения ссылки. |
| **Браузер HotJavaTM (HotJava(TM) Browser)** | Легко настраиваемый Web-браузер, разработанный компанией Sun Microsystems и написанный на языке программирования JavaTM. |
| **язык разметки гипертекста (HyperText Markup Language - HTML)** | Файловый формат для гипертекстовых страниц в Internet, основанный на SGML. Очень прост и разрешает внедрение изображений, звуков, видео, полей форм, а также простое форматирование текста. Ссылки на другие объекты внедряются с помощью URL. |
| **протокол передачи гипертекстовых файлов (HyperText Transfer Protocol - HTTP)** | Internet-протокол, основанный на TCP/IP и предназначенный для доступа к гипертекстовым объектам с удаленного компьютера. См. также TCP/IP. |
| **протокол защищенной передачи гипертекстов (HyperText Transmission Protocol, Secure - HTTPS)** | HTTP в совокупности с SSL (Secure Sockets Layer) - протоколом защищенных сокетов. |
| **язык описания интерфейсов (Interface Definition Language - IDL)** | Интерфейсы прикладного программирования (APIs), написанные на языке JavaTM, которые обеспечивают функциональную совместимость на основе стандартов и способность взаимодействия с CORBA (Common Object Request Broker Architecture). |
| **идентификатор (identifier)** | Имя элемента (переменной, класса, метода и проч.) программы, написанной на языке JavaTM. |
| **Internet InterORB Protocol - IIOP** | Протокол, определяющий передачу сообщений между сетевыми объектами по протоколам TCP/IP и используемый для коммуникаций между брокерами объектных запросов CORBA. |
| **if** | Ключевое слово языка программирования JavaTM, предназначенное для проверки условия и выполнения блока инструкций, если проверка дала положительный результат (true). |
| **заимствование прав (impersonation)** | Процесс, при котором одна сущность принимает идентичность и привилегии другой сущности без ее уведомления о том, что произошло делегирование. Заимствование прав - это случай простого делегирования. |
| **implements** | Ключевое слово языка программирования JavaTM, включаемое при необходимости в объявление класса, для определения интерфейсов, которые реализует данный класс. |
| **import** | Ключевое слово языка программирования JavaTM, определяющее классы или целые пакеты, на которые может ссылаться программа, без включения названий пакетов в ссылку. |
| **наследование (inheritance)** | Концепция классов, автоматически включающих все переменные и методы, определенные в супертипе. См. также суперкласс, подкласс. |
| **параметр инициализации (initialization parameter)** | Параметр, который инициализирует контекст, связанный с сервлетом. |
| **экземпляр (instance)** | Объект класса. В программах, написанных на языке JavaTM, экземпляр класса создается при помощи оператора new, за которым следует имя класса-типа. |
| **метод экземпляра (instance method)** | Любой метод, который вызван от имени экземпляра класса. Также называется просто метод. См. метод класса. |
| **переменная экземпляра (instance variable)** | Любой элемент данных, связанный с объектом. Каждый экземпляр класса имеет собственную копию переменной, определенной в классе. Также носит название поля. См. переменная класса. |
| **instanceof** | Ключевое слово языка программирования JavaTM, имеющее два аргумента и проверяющее, является ли тип первого аргумента преобразуемым к типу второго аргумента. |
| **int** | Ключевое слово языка программирования JavaTM, используемое для определения переменной целого типа (integer). |
| **interface** | Ключевое слово языка программирования JavaTM, используемое для определения набора методов и постоянных значений (класса специального вида). Интерфейс в дальнейшем может реализовываться классами, которые определяют этот интерфейс с ключевым словом implements. |
| **Интернет, международная компьютерная сеть (Internet)** | Огромная сеть, состоящая из миллионов компьютеров многих организаций и стран всего мира. Физически сеть Internet составлена из большого количества сетей, причем данные передаются при помощи единого набора протоколов. |
| **протокол сети Интернет (Internet Protocol - IP)** | Основной протокол Internet. Осуществляет ненадежную доставку индивидуальных пакетов от одного компьютера другому. Не дает никаких гарантий о том, будет ли доставлен пакет, сколько времени займет доставка, придут ли групповые пакеты в том порядке, в котором они были посланы. Протоколы, надстроенные над IP, добавляют понятия связности и надежности. См. также TCP/IP. |
| **интерпретатор (interpreter)** | Программа, которая поочередно декодирует и исполняет каждую инструкцию кода. Интерпретатор JavaTM декодирует и выполняет код для виртуальной машины Java\*. См. также компилятор, система поддержки и выполнения программ. |
| **независимый поставщик ПО (Independent Software Vendor - ISV)** | Фирма, разрабатывающая программное обеспечение для платформы, созданной другой организацией. |
| **приложение J2EE (J2EE application)** | Любой составной модуль, поддерживающий функциональность J2EE. Это может быть один модуль или группа модулей, упакованных в .war-файл, содержащий дескриптор размещения J2EE-приложения. Приложения J2EE обычно проектируются так, что они распределены по нескольким вычислительным уровням. |
| **продукт J2EE (J2EE product)** | Продукт разработки, соответствующий спецификации платформы J2EE. |
| **поставщик продуктов J2EE (J2EE Product Provider)** | Производитель, поставляющий продукты J2EE. |
| **сервер J2EE (J2EE server)** | Исполняемая часть продукта J2EE. Сервер J2EE обеспечивает Web и/или EJB-контейнеры. |
| **JAIN** | См. Java APIs for Integrated NetworksTM. |
| **JAR-файлы (Java ARchive files (.jar))** | Файловый формат, используемый для группирования нескольких файлов в один. |
| **файловый формат JAR (JAR file format)** | Платформно-независимый файловый формат, соединяющий несколько файлов в один. Составные апплеты, написанные на языке JavaTM, и необходимые для них компоненты (.class-файлы, изображения, звуковые и другие файлы) могут быть упакованы в JAR-файл и затем загружены в браузер за одну HTTP-транзакцию. Данный формат также поддерживает сжатие файлов и цифровые подписи. |
| **JavaTM** | Торговая марка компании Sun для ряда технологий по созданию и безопасной работе программного обеспечения как в автономных, так и в сетевых средах. |
| **JavaTM Application Environment - JAE** | Версия исходного кода программного обеспечения Java Development Kit (JDKTM). |
| **Java APIs for Integrated NetworksTM - JAIN** | Дает возможность быстрой разработки продуктов и сервисов, использующих сетевые взаимодействия, на платформе Java. |
| **платформа J2SE (JavaTM 2 Platform, Standard Edition (J2SE platform))** | Ядро платформы технологии Java. |
| **платформа J2EE (JavaTM 2 Platform, Enterprise Edition (J2EE platform))** | Среда для разработки корпоративных приложений. Платформа J2EE состоит из набора услуг, интерфейсов прикладного программирования (APIs) и протоколов, обеспечивающих функциональные возможности для разработки многоуровневых Web-приложений. |
| **JavaTM 2 SDK, Enterprise Edition** | Реализация платформы J2EE компании Sun. Данная реализация включает описание работы платформы J2EE. |
| **JavaBeansTM** | Модель переносимых платформонезависимых компонентов многократного использования. |
| **Java BlendTM** | Продукт, позволяющий разработчику упростить разработку приложений баз данных при помощи отображения записей баз данных в объекты языка JavaTM и Java-объектов - в базы данных. |
| **Java CardTM API** | API для систем с минимальным набором ресурсов, в частности для смарт-карт. Среда прикладных программных средств, соответствующая стандарту ISO 7816-4 для интеллектуальных карт. |
| **JavaCheckTM** | Инструмент для проверки соответствия приложений и апплетов языка Java. |
| **JavaChipTM** | Процессор фирмы Sun, выполняющий байт-код виртуальной машины Java\*. На компьютере с процессором JavaChipTM байт-код минует эмуляцию Java-машины на какой-либо платформе, исполняясь непосредственно аппаратным процессором. |
| **JavaTM Compatibility Kit (JCK)** | Набор испытательных программ и инструментальных средств, используемых для проверки совместимости реализации платформы Java соответствующим спецификациям Java-платформ, а также эталонным реализациям программ Java. TCK для Java 2 Standard Edition. См. TCK. |
| **интерфейс JDBC (Java Database Connectivity - JDBCTM)** | Промышленный стандарт для независимого от базы данных взаимодействия JavaTM-платформы и широкого диапазона баз данных. JDBCTM определяет API для доступа к базам данных из Java-приложений. |
| **Java Developer Connection** | Служба, предназначенная для индивидуальных разработчиков, предоставляющая интерактивное обучение, скидки на продукты, специальные статьи, информацию об ошибках, а также возможности раннего доступа к новым продуктам. |
| **комплект разработчика для Java (Java Development Kit - JDKTM)** | Среда программирования для написания апплетов и приложений в языке программирования Java. |
| **JavaTM Electronic Commerce Framework** | Структурированная архитектура для разработки приложений электронной коммерции в языке программирования JavaTM. |
| **JavaTM Enterprise API** | Данный API упрощает создание крупномасштабных приложений и приложений баз данных, которые совместно используют мультимедийные данные с другими приложениями в пределах организации или через Internet. В семействе JavaTM Enterprise API разработано четыре интерфейса прикладного программирования. |
| **библиотека базовых классов Java (JavaTM Foundation Classes - JFC)** | Дополнительные библиотеки классов графического пользовательского интерфейса (Graphical User Interface - GUI), добавляемые к абстрактному оконному интерфейсу (Abstract Window Toolkit - AWT). |
| **JavaTM IDL** | Технология, обеспечивающая функциональную совместимость и способность к взаимодействию CORBA и J2EE-платформы. Эти возможности позволяют приложениям J2EE вызывать операции удаленных сетевых служб, используя OMG IDL и IIOP. |
| **JavaTM Interface Definition Language (IDL)** | API, написанные на языке программирования Java, которые обеспечивают функциональную совместимость и способность к взаимодействию с CORBA (Common Object Request Broker Architecture). |
| **JavaMailTM** | API для отправки и получения сообщений. |
| **JavaTM Media APIs** | Набор интерфейсов прикладного программирования (APIs) для интеграции аудио и видео файлов, двумерных шрифтов, графики и изображений, а также трехмерных моделей и телефонии. |
| **JavaTM Media Framework** | Ядро структуры поддерживает часы для синхронизации различных потоков (например, аудио и видео сигналов). Standard extension framework позволяет пользователям организовать потоки аудио и видео информации. |
| **JavaTM Message Service - JMS** | Интерфейс прикладного программирования (API) для использования корпоративных систем передачи сообщений, таких как IBM MQ Series, TIBCO Rendezvous и т.д. |
| **Java Naming and Directory InterfaceTM (JNDI)** | Стандартный API к корпоративной службе каталогов. |
| **операционная система Java (JavaOSTM)** | Операционная система, основанная на технологии JavaTM и оптимизированная для работы на различных платформах. Операционная среда JavaOSTM обеспечивает работу Java-приложений непосредственно на аппаратных платформах, минуя ведущую операционную систему. |
| **JavaPlanTM** | Инструмент для объектно-ориентированного проектирования и схематического изображения, написанный на языке программирования JavaTM. |
| **платформа JavaTM (JavaTM Platform)** | Состоит из языка Java, предназначенного для написания программ, набора интерфейсов прикладного программирования (APIs), библиотек классов, других программ, используемых при разработке и компиляции, программ проверки ошибок, а также виртуальной машины Java, которая загружает и выполняет файлы классов. |
| **издания платформы Java (JavaTM Platform Editions)** | "Издание" платформы Java - это окончательная и согласованная версия платформы Java, которая обеспечивает функциональные возможности, необходимые в широком секторе рынка. Издание составлено из двух видов API: (i) "основные пакеты", которые являются необходимыми для всех реализаций данного издания платформы; (ii) "дополнительные пакеты", которые доступны данному изданию платформы и могут поддерживаться совместимыми реализациями. Существует три различных издания платформы Java:    - Java 2 Platform, Enterprise Edition: Издание платформы Java, используемое на предприятиях и предназначенное для разработки и развития многоуровневых приложений, управляемых центральным сервером; -    Java 2 Platform, Standard Edition: Издание платформы Java, предназначенное для разработки, развития и управления межплатформенными универсальными приложениями.; -    Java 2 Platform, Micro Edition: Издание платформы Java, предназначенное для малых автономных потребителей, с целью разработки, развития и управления приложений, варьирующихся от смарт-карт до обычных вычислительных устройств. |
| **JavaTM Remote Method Invocation (RMI)** | Распределенная объектная модель для Java-программы, в которой методы и удаленные объекты, написанные на языке программирования Java, могут быть вызваны другой виртуальной машиной Java, возможно, расположенной на другом компьютере. |
| **JavaTM Runtime Environment (JRE)** | Подмножество комплекта разработчика Java (Java Developer Kit - JDKTM) для конечных пользователей и разработчиков, которые хотят самостоятельно настроить оперативные средства управления работой программы (run-time environment). Оперативные средства включают в себя виртуальную машину Java\*, классы ядра Java и вспомогательные файлы. |
| **JavaSafeTM** | Инструмент для отслеживания и управления изменениями исходного файла, написанного на языке программирования Java. |
| **язык сценариев JavaScript (JavaScriptTM)** | Язык сценариев, который используется как браузерами, так и Web-серверами. Подобно всем языкам сценариев, предназначен главным образом для интеграции компонент и пользовательского ввода. |
| **Java Server PagesTM (JSP)** | Расширяемая Web-технология, использующая данные шаблона, заказные элементы, языки сценариев и серверные объекты Java для предоставления клиенту динамического содержания. Обычно, данные шаблона - это HTML или XML элементы, а клиент - это Web-браузер. |
| **JAX** | JAX относится к набору интерфейсов прикладного программирования (APIs), который предназначен для управления различными операциями, включающими данные, определенные посредством XML. Сюда входят такие операции как синтаксический анализ XML, регистрация в репозиториях ebXML или UDDI, обмен сообщениями между приложениями, привязка данных и удаленный вызов процедур. |
| **действие Java Server PagesTM (JSP) (JSP action)** | JSP элемент, который может действовать на неявные объекты и другие серверные объекты или определять новые переменные создания сценария. Действия придерживаются синтаксиса XML для элементов с начальным тэгом, телом и конечным тэгом; если тело пусто, может использоваться синтаксис пустого тэга. Тэг должен использовать префикс. |
| **стандартное действие Java Server PagesTM (JSP action, standard)** | Действие, которое определено спецификацией JSP и всегда доступно JSP-файлу без импортирования. |
| **специальное действие Java Server PagesTM (JSP action, custom)** | Действие, описанное с помощью тэгов и набора классов Java, включенных в страницу JSP с помощью тэговых дескрипторов. |
| **приложение JSP (JSP application)** | Автономное Web- приложение, написанное с использованием технологии Java Server Pages и включающее JSP файлы, сервлеты, HTML файлы, изображения, апплеты и компоненты JavaBeans. |
| **контейнер JSP (JSP container)** | Контейнер, предоставляющий те же услуги, что и сервлет-контейнер, а также механизм интерпретации и переработки JSP страниц в сервлеты. |
| **распределенный контейнер JSP (JSP container, distributed)** | JSP контейнер, который может запускать Web-приложения, помеченные как распределенные и выполняемые одновременно на нескольких виртуальных машинах Java. При этом виртуальные машины могут быть запущены, как на одном, так и на разных компьютерах. |
| **объявление JSP (JSP declaration)** | Элемент сценариев JSP, который объявляет методы и переменные в JSP файле. |
| **директива JSP (JSP directive)** | Элемент JSP, который дает команды JSP контейнеру и интерпретируется во время трансляции. |
| **элемент JSP (JSP element)** | Часть JSP страницы, которая распознается JSP транслятором. Элемент JSP может быть директивой, действием или элементом сценария. |
| **выражение JSP (JSP expression)** | Элемент сценария, который содержит допустимое выражение языка сценариев, вычисленное, преобразованное в строку и помещенное в неявный выходной объект. |
| **файл JSP (JSP file)** | Файл с расширением .jsp, который создается разработчиком при помощи стандартных тэгов HTML, основных JSP тэгов и инструкций языка сценариев, для отображения динамических страниц в Web-браузере. |
| **страница JSP (JSP page)** | Текстовый документ, использующий фиксированные шаблонные данные и элементы JSP. Описывает, как обработать запрос. |
| **элемент сценария JSP** | Объявление JSP, скриптлет или выражение, синтаксис которого определен спецификацией JSP и содержимое которого написано согласно языку сценариев, используемому в странице JSP. Спецификация JSP описывает синтаксис и семантику для того случая, когда атрибут языка страницы - java. |
| **скриптлет JSP (JSP scriptlet)** | Элемент сценария JSP, содержащий любой фрагмент кода, допустимый в том языке сценариев, который используется на JSP странице. Спецификация JSP определяет, что является допустимым скриптлетом для того случая, когда атрибут языка страницы - java. |
| **тэг JSP (JSP tag)** | Текст между левой и правой угловыми скобками, который используется в файлах JSP, как часть элемента JSP. В отличие от данных, тэг является элементом разметки документа, т.к. он выделен угловыми скобками. |
| **библиотека тэгов JSP (JSP tag library)** | Совокупность тэгов, описанных посредством библиотечных дескрипторов и классов Java. Библиотека тэгов JSP может импортироваться в любой JSP файл и использоваться с различными языками сценариев. |
| **Java StudioTM** | Инструментальный комплекс для построения Java-программ из готовых компонент, основанный на визуальном стиле проектирования и предназначенный для непрограммистов. |
| **технологии Java (JavaTM Technologies)** | Ряд технологий по созданию и безопасной работе программного обеспечения, как в автономных, так и в сетевых средах. |
| **JavaTM Transaction API (JTA)** | API, позволяющий приложениям и сервлетам J2EE иметь доступ к транзакциям. |
| **JavaTM Transaction Service (JTS)** | Определяет реализацию менеджера транзакций, который поддерживает JTA и осуществляет Java отображение спецификации OMG Object Transaction Service (OTS) 1.1 на уровень ниже API. |
| **виртуальная машина Java\* (JavaTM virtual machine - JVM\*)** | Программный "механизм выполнения", который безопасно выполняет байт-коды файлов классов Java на микропроцессоре (компьютера или другого электронного устройства). Программный "механизм выполнения", который безопасно выполняет байт-коды файлов классов Java на микропроцессоре (компьютера или другого электронного устройства). Механизм выполнения Java HotSpot - это высокоэффективный механизм для среды выполнения Java программ, который представляет собой адаптивный компилятор, динамически оптимизирующий работу приложений. Виртуальная машина KJava - малогабаритная, высоко оптимизированная основа среды выполнения программ (runtime environment) в Java 2 Platform, Micro Edition. KJava разработана на основе виртуальной машины Java и предназначена для малых устройств связи. Ее размер составляет от 30 до 128 Кб, в зависимости от функциональных возможностей устройства. Виртуальная машина Java Card - малогабаритная, высоко оптимизированная основа среды выполнения программ (runtime environment) в Java 2 Platform, Micro Edition. KJava разработана на основе виртуальной машины Java и предназначена для смарт-карт и других устройств с ограниченной памятью (порядка 24Кб ПЗУ, 16Кб EEPROM и 512б ОЗУ). |
| **Web-сервер Java (Java Web ServerTM)** | Удобное, открытое, легко администрируемое, безопасное, платформонезависимое решение для ускорения и простоты разработки Internet/Intranet Web-страниц. Обеспечивает немедленное повышение производительности для трудоемких, полнофункциональных серверных приложений Java. |
| **Java WorkshopTM** | Законченный набор инструментов, интегрированный в единую среду программирования с использованием Java технологий. Программное обеспечение Java Workshop использует модульную структуру, которая позволяет легко подключать новые инструментальные средства. |
| **JavaTM wallet** | Пользовательский интерфейс, основанный на JavaTM Electronic Commerce Framework, предназначенный для осуществления интерактивных покупок, передачи данных и выполнения административных функций. |
| **JavaSpacesTM** | Технология, содержащая механизмы распределенной работы и обмена данными при программировании на языке JavaTM. |
| **JavaSoftTM** | Ранее филиал компании Sun Microsystems, в настоящее время известный как подразделение Sun Microsystems Java Software. |
| **JDBCTM** | См. интерфейс JDBC (Java Database Connectivity - JDBCTM). |
| **JDKTM** | См. комплект разработчика для Java (Java Development Kit - JDKTM). |
| **JFC** | См. библиотека базовых классов Java (JavaTM Foundation Classes - JFC). |
| **технология Jini (JiniTM Technology)** | Набор интерфейсов прикладного программирования (APIs) Java, которые могут быть включены в дополнительный пакет к любому изданию платформы Java 2. Jini позволяет обеспечивать совместную работу в сети различных устройств и служб, а также устраняет необходимость системного или сетевого административного вмешательства пользователя. В настоящее время технология Jini является дополнительным пакетом, доступным любой Java платформе. |
| **JavaTM Management API - JMAPI** | Совокупность Java классов и интерфейсов, позволяющих разработчику создавать системные, сетевые и служебные приложения. |
| **JMS** | См. JavaTM Message Service. |
| **интерфейс JNDI** | См. Java Naming and Directory InterfaceTM. |
| **стандарт JPEG (Joint Photographic Experts Group - JPEG)** | Стандарт сжатия файлов изображений, установленный группой экспертов по машинной обработке фотографических изображений. Сильное сжатие достигается ценой внесения искажений в изображение, которые почти всегда являются незаметными. |
| **JRE** | См. JavaTM Runtime Environment. |
| **компилятор JIT (Just-in-time (JIT) Compiler)** | Компилятор, динамически ("на лету") преобразующий байт-код в объектный код целевой платформы. Его применение приводит к значительному ускорению выполнения Java-программы. |
| **JVM** | См. виртуальная машина Java\*. |
| **ключевое слово (keyword)** | Слово, которое зарезервировано языком программирования JavaTM, и поэтому не может быть именем переменной или метода. |
| **лексический (lexical)** | Соответствующий уровню лексем (идентификаторов, чисел, изображений строк и т.д.) языка программирования. Лексический анализ - фаза компилятора, на которой последовательность символов файла исходного текста преобразуется в последовательность лексем. |
| **компоновщик (linker)** | Модуль, который формирует запускаемую, законченную программу из составных модулей машинного кода. Компоновщик JavaTM создает работоспособную программу из откомпилированных классов. См. также компилятор, интерпретатор, система поддержки исполнения программ. |
| **литерал (literal)** | Основное представление любого целого, символьного значения или значения с плавающей запятой. Например, 3.0 - литерал с плавающей запятой двойной точности, "а" - символьный литерал. |
| **локальная переменная (local variable)** | Элемент данных, известный в пределах блока, но недоступный вне блока. Например, любая переменная, определенная внутри метода, является локальной и не может использоваться вне этого метода. |
| **long** | Ключевое слово языка программирования JavaTM, используемое для определения переменных типа long. |
| **элемент (member)** | Поле или метод класса. Пока не оговорено обратное, член не является статическим. |
| **метод (method)** | Функция, определенная в классе. См. также метод экземпляра, метод класса. Пока не оговорено обратное, метод не является статическим. |
| **разрешение на вызов метода (method permission)** | Разрешение на вызов указанной группы методов "домашнего" или удаленного интерфейса корпоративного компонента (enterprise bean). |
| **модуль (module)** | Программный модуль, который состоит из одного или более компонентов J2EE, принадлежащих контейнеру одного типа, и из дескриптора размещения этого типа. Существует три типа модулей: EJB, Web и клиентские приложения. Модули можно использовать автономно или собирать в приложения. |
| **Mosaic** | Программа c несложным графическим интерфейсом (GUI), предназначенная для простого доступа к данным, хранящимся в Internet. Данные могут быть обычными файлами или гипертекстовыми документами. Mosaic была разработана в NCSA. |
| **многопоточный (multithreaded)** | Описывает программу, спроектированную и реализованную в виде совокупности параллельных потоков управления (threads), синхронизированных между собой по общим ресурсам и событиям. См. также поток. |
| **взаимная аутентификация (mutual authentication)** | Процесс, при котором клиент использует сертификат с открытым ключом для установления своей идентичности и поддерживает защиту своего контекста. |
| **native** | Ключевое слово языка программирования JavaTM, используемое в объявлении метода для указания на то, что метод реализован не в файле Java, а на другом языке. |
| **Национальный центр по приложениям для суперкомпьютеров (National Center for Supercomputer Applications - NCSA)** | Исследовательский центр, в котором была разработана программа Mosaic. |
| **new** | Ключевое слово языка программирования JavaTM, используемое для создания нового экземпляра класса. |
| **null** | Тип null имеет единственное значение - "пустую" ссылку, представленную литералом null, который сформирован из символов ASCII. Литерал null всегда имеет тип null. |
| **объект (object)** | Основной компоновочный блок объектно-ориентированных программ. Каждый объект программного модуля состоит из данных (переменные экземпляра) и функциональных возможностей (методы экземпляра). См. также класс. |
| **объектно-ориентированное проектирование (object-oriented design)** | Метод проектирования программного обеспечения, позволяющий моделировать абстрактные или реальные объекты при помощи классов и объектов. |
| **восьмеричный (octal)** | Система счисления с основанием 8. Знаки 0-7 представляют цифры. В программе, написанной на языке JavaTM, восьмеричные числа должны начинаться с символа 0. См. также шестнадцатеричный. |
| **открытое сетевое окружение (Open Net Environment - ONE)** | Сетевая среда, которая была разработана компанией Sun Microsystems и поддержана многими ведущими производителями. Описывает всестороннюю архитектуру для создания, сборки и использования сетевых услуг. ONE платформонезависима и полностью основана на открытых стандартах. Sun ONE - конкретная реализация данной архитектуры, разработанная компаниями Sun и iPlanet. |
| **дополнительный пакет (Optional Package)** | Набор API в издании платформы Java, который может быть доступным или поддерживаться в совместимой реализации. По мере необходимости, дополнительные пакеты могут становиться необходимыми в издании. |
| **посредник запросов к объектам (Object Request Broker - ORB)** | Библиотека, позволяющая объектам CORBA определять местонахождение и устанавливать связь друг с другом. |
| **принцип (OS principal)** | Свойство (principal), присущее той операционной системе, на которой запущена платформа J2EE. |
| **Object Transaction Service (OTS)** | Интерфейсы, позволяющие объектам CORBA участвовать в транзакциях. |
| **перегрузка (операций) (overloading)** | Использование одного идентификатора для ссылки на разные элементы в одной области действия. В языке программирования JavaTM можно перегружать методы, однако, нельзя перегружать переменные или операторы. |
| **замещение (overriding)** | Означает другую реализацию метода в подклассе класса, первоначально определившего метод. |
| **пакет (package)** | Группа типов. Пакеты объявляются при помощи ключевого слова package. |
| **пассивация (passivation)** | Процесс передачи корпоративного компонента (enterprise bean) из памяти во вторичное устройство хранения данных. См. также активация. |
| **одноранговые (peer)** | В организации сетей любые функциональные единицы, находящиеся на одном уровне. |
| **persistence** | Протокол передачи состояния сущности (entity bean) между переменными ее экземпляра и базой данных. |
| **PersonalJavaTM** | Среда выполнения Java для сетевых приложений на персональных устройствах потребителя (домашнего, мобильного или офисного использования). |
| **пиксель (pixel)** | Элемент площади изображения, например, экрана монитора или напечатанной страницы. Каждый пиксель является индивидуально доступным. |
| **Portable Object Adapter - POA** | Стандарт CORBA для создания серверных приложений, которые переносимы между различными посредниками запросов к объектам (ORBs). |
| **интерфейс переносимой операционной системы (Portable Operating System Interface - POSIX)** | Стандарт, который определяет языковой интерфейс между операционной системой UNIX и прикладными программами посредством минимального набора поддерживаемых функций. |
| **первичный ключ (primary key)** | Объект, уникально идентифицирующий сущность (entity bean). |
| **принцип (principal)** | Отличительная черта, присвоенная сущности в результате аутентификации. |
| **private** | Ключевое слово языка программирования JavaTM, используемое при объявлении метода или переменной. Указывает на то, что к методу или переменной класса не могут обращаться элементы других классов. |
| **привилегия (privilege)** | Атрибут защиты, который не имеет свойства уникальности и который может быть использован несколькими principals. Пример привилегии - группа. |
| **процесс (process)** | Виртуальное адресное пространство, содержащее один или более потоков. |
| **свойство (property)** | Характеристика объекта, которую может устанавливать пользователь. Например, цвет окна. |
| **профили (Profiles)** | Наборы Java API, которые служат дополнением к изданиям платформ Java и обеспечивают дополнительные возможности. Профили также могут включать другие определенные профили. Реализация профиля нуждается в издании Java Platform для создания законченной разработки и среды ее использования на целевом вертикальном рынке. Каждый профиль подчиняется связанному набору требований совместимости. Профили могут использоваться одним или несколькими изданиями. Примеры профилей в Java 2 Platform, Micro Edition: PersonalJavaTM, Java CardTM. |
| **protected** | Ключевое слово языка программирования JavaTM, используемое при объявлении метода или переменной. Указывает на то, что к методу или переменной класса могут обращаться другие элементы данного класса, его подклассов или классов из того же пакета. |
| **private** | Ключевое слово языка программирования JavaTM, используемое при объявлении метода или переменной. Указывает на то, что к методу или переменной класса могут обращаться элементы других классов. |
| **растр (raster)** | Двумерная прямоугольная сетка пикселей. |
| **область (realm)** | См. область политики безопасности. Строка, проходящая базовую аутентификацию, как часть HTTP-запроса, которая определяет пространство защиты. Защищенные ресурсы на сервере могут находиться в разных пространствах защиты, каждое со своей схемой аутентификации и/или базой данных авторизации. |
| **реентерабельный корпоративный компонент (re-entrant enterprise bean)** | Корпоративный компонент, который может обрабатывать несколько одновременных или вложенных вызовов, не пересекающихся друг с другом. |
| **ссылка (reference)** | Элемент данных, значение которого является адресом памяти. |
| **ссылочная реализация (Reference Implementation - RI)** | Прототип реализации спецификации Java-технологии. Является неотъемлемой частью любой Java-технологии и служит доказательством того, что данная Java-технология может быть реализована на практике. |
| **удаленный интерфейс (remote interface)** | Один из двух интерфейсов корпоративной компоненты (enterprise bean). Удаленный интерфейс определяет бизнес-методы, вызываемые клиентом. |
| **метод удаления (remove method)** | Метод, определенный в "домашнем" интерфейсе и вызываемый клиентом для уничтожения корпоративной компоненты (enterprise bean). |
| **адаптер ресурсов (resource adapter)** | Программный драйвер системного уровня, используемый EJB-контейнером или клиентским приложением для связи с EIS. Адаптер ресурсов специфичен для каждой EIS. Он представляет собой библиотеку, и используется в пределах адресного пространства сервера или клиента, которые используют адаптер. Адаптер ресурсов подключается к контейнеру. Прикладные компоненты, содержащиеся в контейнере, используют API (представленный адаптером) или инструментально созданные высокоуровневые абстракции для доступа к низкоуровневой EIS. Адаптер ресурса и EJB-контейнер взаимодействуют для того, чтобы обеспечить низкоуровневые механизмы - транзакции, безопасность, организация связного пула - для связи с EIS. |
| **менеджер ресурсов (resource manager)** | Обеспечивает клиенту доступ к набору общедоступных ресурсов. Менеджер ресурсов участвует в транзакциях, которые управляются и координируются менеджером транзакций. Менеджер ресурсов обычно находится в другом адресном пространстве или на другой машине. Замечание: к EIS обращаются, как к менеджеру ресурсов, когда она упомянута в контексте управления ресурсами и транзакциями. |
| **соединение с менеджером ресурсов (resource manager connection)** | Объект, представляющий собой сеанс связи с менеджером ресурсов. |
| **мастер соединения с менеджером ресурсов (resource manager connection factory)** | Объект, используемый для создания сеанса связи с менеджером ресурсов. |
| **return** | Ключевое слово языка программирования JavaTM, используемое для окончания выполнения метода. За ним может следовать значение, возвращаемое методом. |
| **запрос на улучшение (Request for Enhancement - RFE)** | RMI |
| **RMI** | См. JavaTM Remote Method Invocation. |
| **роль разработки (role (development))** | Функция, выполняемая человеком в стадии развития приложения, разрабатываемого с помощью технологии J2EE. Примеры ролей: поставщик программных компонент (Application Component Provider), компоновщик приложения (application assembler), "разместитель" (deployer), поставщик платформы J2EE (J2EE Platform Provider), поставщик EJB-контейнера (EJB Container Provider), поставщик EJB-сервера (EJB Server Provider), поставщик Web-контейнера (Web Container Provider), поставщик Web-сервера (Web Server Provider), поставщик инструментов (Tool Provider), системный администратор (System Administrator). |
| **роль безопасности (role (security))** | Абстрактная логическая группировка пользователей, осуществляемая компоновщиком приложения. Когда приложение установлено, роли ставятся в соответствие идентичностям безопасности, таким как principals или группы. |
| **распределение ролей (role mapping)** | Процесс объединения групп, распознанных контейнером, в роли безопасности, которые определены в дескрипторе размещения. Роли безопасности должны быть составлены "разместителем" перед установкой контейнера на сервер. |
| **откат (rollback)** | Момент транзакции, когда все изменения, вносимые в базу данных, отменяются. |
| **корень (root)** | Элемент иерархии, из которого происходят все элементы. Ни один элемент не может находиться в иерархии выше "корня". См. также иерархия, класс, пакет. |
| **вызов удаленной процедуры (Remote Procedure Call - RPC)** | Протокол, позволяющий приложениям вызывать процедуры, физически расположенные в другой части сети. |
| **система поддержки исполнения программ (runtime system)** | Программная среда, в которой могут работать программы, откомпилированные для виртуальной машины JavaTM\*. Система поддержки исполнения включает весь код, необходимый для того, чтобы загружать программы, написанные на Java, динамически связывать "родные" методы, управлять памятью, обрабатывать исключительные ситуации, а также реализацию виртуальной машины Java. |
| **простой API для XML (Simple API for XML - SAX)** | Управляемый событиями, последовательный механизм доступа к XML документам. |
| **Sandbox** | Содержит множество взаимодействующих системных компонент, начиная от менеджеров безопасности, которые выполняются как часть приложения, и заканчивая мерами безопасности, встроенными непосредственно в виртуальную машину Java\* и язык Java. Sandbox гарантирует, что ненадежные и, возможно, злонамеренные приложения не смогут получить доступ к системным ресурсам. |
| **область действия (scope)** | Характеристика идентификатора, определяющая, где может использоваться данный идентификатор. Большинство идентификаторов в языке программирования Java имеют либо локальную область действия, либо область действия - класс. Для переменных экземпляров и классов, а также методов областью действия является класс; они могут использоваться вне класса и его подклассов, только если перед ними ставится имя экземпляра класса или (в случае переменных и методов класса) имя самого класса. Все другие переменные объявляются внутри методов и имеют локальную область действия; они могут использоваться только внутри блока. |
| **протокол безопасных соединений (Secure Socket Layer - SSL)** | Протокол, который позволяет шифровать сообщения, передаваемые между Web-браузером и сервером в целях безопасности. |
| **атрибуты безопасности (security attributes)** | Набор свойств, связанных с principal. Атрибуты безопасности могут быть связаны с principal посредством протокола аутентификации и/или поставщиком продуктов J2EE (J2EE Product Provider). |
| **ограничения безопасности (security constraint)** | Декларативный способ аннотирования необходимой защиты содержимого сети. Ограничение безопасности состоит из совокупности Web-ресурсов, ограничения авторизации, ограничения пользовательских данных. |
| **контекст безопасности (security context)** | Объект, включающий разделяемые свойства для описания безопасности сущностей. |
| **право безопасности (security permission)** | Механизм, определенный J2SE и используемый платформой J2EE для определения программных ограничений, наложенных на поставщиков программных компонент (Application Component Providers). |
| **набор прав безопасности (security permission set)** | Минимальный набор прав безопасности, обеспечиваемый поставщиком программных компонент (Application Component Provider) для выполнения каждого типа компонент. |
| **область политики безопасности (security policy domain)** | Область, в переделах которой определена и приведена в исполнение администратором политика безопасности. Область политики безопасности имеет следующие характеристики:    обладает набором пользователей (или principals); использует хорошо определенный протокол аутентификации пользователей (или principals); может иметь группы для упрощения настройки политики безопасности. |
| **security role** | См. роль безопасности (role(security)). |
| **область технологии безопасности (security technology domain)** | Область, в пределах которой для осуществления политики безопасности используется один и тот же механизм. В пределах единой области технологии безопасности может существовать несколько областей политики безопасности. |
| **server principal** | Принцип ОС, в соответствии с которым работает сервер. |
| **сервлет (servlet)** | Java программа, которая расширяет функциональные возможности Web-сервера, динамически генерируя содержание и взаимодействуя с Web-клиентами при помощи принципа запрос-ответ. |
| **контейнер сервлета (servlet container)** | Контейнер, обеспечивающий сетевые службы, при помощи которых посылаются запросы и ответы, декодируются запросы и форматируются ответы. Все контейнеры сервлетов должны поддерживать HTTP-протокол, но могут также поддерживать дополнительные протоколы, например, HTTPS. |
| **распределенный контейнер сервлета (servlet container, distributed)** | Контейнер сервлета, запускающий Web-приложения, которые помечены как распределенные и выполняются на нескольких виртуальных машинах Java. При этом виртуальные машины могут быть запущены, как на одном, так и на разных компьютерах. |
| **контекст сервлета (servlet context)** | Объект, содержащий представление (вид) Web-приложения, в котором запущен сервлет. Используя контекст, сервлет может вести журнал событий, получать URL-ссылки на ресурсы, а также устанавливать и хранить атрибуты, которые могут использоваться другими сервлетами в приложении. |
| **отображение сервлета (servlet mapping)** | Определяет связь между структурой URL и сервлетом. Используется для отображения запросов в сервлеты. Если контейнер, обрабатывающий запрос, является JSP-контейнером, то неявно отображается URL, содержащий расширение .jsp. |
| **сессия (session)** | Объект, используемый сервлетом, для прослеживания взаимодействий пользователя с Web-приложением при помощи множества HTTP-запросов. |
| **сессионный компонент (session bean)** | Корпоративный компонент (enterprise bean), который создается клиентом и обычно существует только в течение одной клиент-серверной сессии. Сессионный компонент выполняет для клиента вычислительные операции и организует доступ к базе данных. В случае сбоя системы сессионный компонент не восстанавливается. Объекты сессионных компонент могут не менять своего состояния или поддерживать диалоговое состояние в процессе выполнения методов и транзакций. Если объект поддерживает состояние, то EJB-контейнер управляет этим состоянием, если объект должен быть удален из памяти. Однако объекты сессионного компонента должны управлять собственными хранимыми данными. |
| **short** | Ключевое слово языка программирования JavaTM, используемое для определения переменных типа short. |
| **одинарная точность (single precision)** | В спецификации языка Java описывает число с плавающей запятой, занимающее 32 бита данных. См. также двойная точность. |
| **стандартный язык обобщенной разметки (Standardized Generalized Markup Language - SGML)** | Стандарт ISO/ANSI/ECMA для определения структуры и управления содержимым любого электронного документа. |
| **Smart Web Services** | Расширяют основную концепцию сетевых услуг, добавляя пользовательский контекст, и способны модифицировать свои действия, чтобы следить за изменениями текущего состояния клиента. Сюда входят классические дескрипторы "кто, что, когда, где, почему", которые объединяются для определения пользовательского контекста в данный момент. |
| **The Simple Object Access Protocol - SOAP** | Использует основанное на XML структурирование данных и HTTP для того, чтобы определить стандартизованные методы для вызова методов объектов, распределенных в различных средах по всей Internet. |
| **язык структурированных запросов (Structured Query Language - SQL)** | Стандартизованный язык реляционных баз данных, предназначенный для определения объектов баз данных и манипулирования данными. |
| **сессионный компонент с диалоговым состоянием (stateful session bean)** | Сессионный компонент с диалоговым состоянием. |
| **сессионный компонент, не имеющий состояния (stateless session bean)** | Сессионный компонент, не имеющий состояния. Все экземпляры такого компонента идентичны. |
| **static** | Ключевое слово языка программирования JavaTM, используемое для определения переменной класса (типа). Классы обеспечивают только одну копию таких переменных, не зависимо от того, сколько экземпляров класса было создано. Слово static также может использоваться при определении метода класса. Такие статические методы вызываются от имени не экземпляра объекта, а его типа (класса), и могут, в свою очередь, оперировать только статическими переменными. |
| **статическое поле (static field)** | Другое название переменной класса. |
| **статический метод (static method)** | Другое название метода класса. |
| **поток (stream)** | Последовательность байтов данных, пересылаемых от отправителя к получателю. Существует две основные категории потоков, поэтому пакет java.io включает два абстрактных класса (InputStream и OutputStream). |
| **подмассив (subarray)** | Массив, содержащийся в другом массиве. |
| **подкласс (subclass)** | Класс, который "произведен" из другого класса. См. также суперкласс, супертип. |
| **подтип (subtype)** | Если тип X "расширяет" или реализует тип Y, то X - подтип типа Y. См. также супертип. |
| **суперкласс (superclass)** | Класс, из которого "произведены" другие классы. См. также подкласс, подтип. |
| **super** | Ключевое слово языка программирования JavaTM, используемое для доступа к членам класса, наследуемого классом, из которого производится вызов. |
| **супертип (supertype)** | Все интерфейсы или классы, расширяемые или реализуемые данным типом. См. также подтип, суперкласс. |
| **switch** | Ключевое слово языка программирования JavaTM, используемое для определения переменной, которая в дальнейшем может быть использована ключевым словом case для выполнения блока инструкций. |
| **Swing** | Кодовое название совокупности графических компонентов, которые выполняются на любой платформе, поддерживающей виртуальную машину JavaTM. Данные компоненты могут обеспечивать большие функциональные возможности, вследствие того, что они целиком написаны на языке JavaTM. См. также AWT. |
| **synchronized** | Ключевое слово языка программирования JavaTM, которое при применении к методу или блоку кода, гарантирует, что данный код будет выполняться не более чем одним потоком одновременно. |
| **системный администратор (System Administrator)** | Человек, ответственный за конфигурирование и администрирование компьютеров компании, сетей и систем программного обеспечения. |
| **TCP/IP (TCP/IP)** | Протокол управления передачей данных, основанный на IP. Internet-протокол, который обеспечивает надежную транспортировку потоков данных между компьютерами. См. также IP. |
| **Испытатель Совместимости Технологии (TCK) (Technology Compatibility Kit - TCK)** | Набор тестов, инструментов, сопутствующих утилит и документации, которые позволяют реализатору Спецификации той или иной технологии определить, соответствует ли его реализация заданной Спецификации. Любая Java-технология состоит из Спецификации, ТСК и ссылочной реализации (reference implementation). |
| **"тонкий" клиент (Thin Client)** | Система, использующая очень упрощенную операционную систему, не требующая локального системного администрирования и выполняющая приложения, загруженные из сети. |
| **this** | Ключевое слово языка JavaTM, которое может использоваться для ссылки на экземпляр класса, в котором используется эта ссылка. Слово this может использоваться для доступа, как к полям, так и к методам класса. |
| **процесс (поток) (thread)** | Основная единица выполнения программы. Процесс может иметь несколько потоков, работающих одновременно и выполняющих различные задачи, такие как, ожидание события или исполнение трудоемкой по времени работы, окончание которой не требуется для дальнейшего выполнения программы. После выполнения потоком своей работы он приостанавливается или уничтожается. См. также процесс. |
| **throw** | Ключевое слово языка Java TM, которое позволяет пользователю сгенерировать исключительную ситуацию или любой класс, реализующий "throwable" интерфейс (т.е. интерфейс, позволяющий классу генерировать прерывания). |
| **throws** | Ключевое слово языка JavaTM, используемое в описании метода и определяющее, какие исключительные ситуации не обрабатываются внутри метода, а передаются на следующий, более высокий уровень программы. |
| **поставщик инструментов (Tool Provider)** | Организация или поставщик программного обеспечения, предоставляющий инструменты, используемые для разработки, компоновки и внедрения J2EE приложений. |
| **транзакция (transaction)** | Минимальная единица работы по изменению данных. Транзакция включает в себя одну или несколько программных инструкций, которые могут выполниться только все вместе. Если хотя бы одна из инструкций не выполняется, происходит возврат в исходное состояние (откат). Транзакции разрешают одновременный доступ нескольким пользователям к одним и тем же данным. |
| **атрибут транзакции (transaction attribute)** | Значение, определяемое в дескрипторе внедряемой корпоративной компоненты (enterprise bean), которое используется в контейнере EJB для управления областью действия транзакции при вызове методов компоненты. Атрибут транзакции может принимать следующие значения: Required, RequiresNew, Supports, NotSupported, Mandatory, Never ("требуется", "требуется для новых", "поддерживается", "не поддерживается", "принудительно", "никогда"). |
| **уровень изоляции транзакции (transaction isolation level)** | Степень видимости промежуточного состояния модифицируемых транзакцией данных для других параллельных транзакций и данных, модифицируемых другими транзакциями, для данной транзакции. |
| **менеджер транзакции (transaction manager)** | Обеспечивает сервисы и управляющие функции, требуемые для поддержки разграничения транзакции, управления ресурсами транзакции, синхронизации и прохождения содержания транзакции. |
| **transient** | Ключевое слово языка JavaTM, которое определяет, что поле не является частью сериализуемой формы объекта. Когда объект сериализуется, значения его transient полей не включаются в представление сериализации, в то время как значения не transient полей включаются. |
| **try** | Ключевое слово языка Java(TM), определяющее блок операторов, которые могут генерировать исключительные ситуации Java. При возникновении исключительной ситуации необязательный catch блок может обработать определенные исключительные ситуации, сгенерированные внутри блока try. Также необязательный блок finally будет выполнен независимо от того, генерировалась исключительная ситуация или нет. |
| **type** | Класс или интерфейс. |
| **UDDI** | Проект "Универсальное обнаружение и интеграция описаний" (The Universal Description Discovery and Integration - UDDI) обеспечивает глобальный, публичный, основанный на XML, онлайновый бизнес-регистр, в котором пользователи регистрируют и представляют свои web службы. UDDI представляет собой Internet-версию желтых страниц в справочнике телефонов. |
| **уникод (Unicode)** | 16-битная кодовая таблица, определенная ISO 10646. См. также ASCII. Все исходные коды программной среды Java(TM) написаны в Unicode. |
| **унифицированный идентификатор ресурсов (Uniform Resource Identifier - URI)** | Компактная строка символов для идентификации абстрактного или физического ресурса. URI может быть или URL или URN. URL и URN представляют собой конкретные сущности, которые действительно существуют; URI является абстрактным суперклассом. |
| **унифицированный указатель информационного ресурса (Uniform Resource Locator - URL)** | Стандарт для записи текстовой ссылки на произвольные данные в WWW. URL выглядит следующим образом: "протокол://хост/локальная\_информация", где протокол определяет конкретный протокол, используемый для доступа к объекту (например HTTP или FTP), хост определяет Internet-имя хоста, на котором объект находится, и локальная\_информация - строка (часто имя файла), передаваемая обработчику протокола на удаленном хосте. |
| **URL path** | URL, передаваемый запросом HTTP для вызова сервлета. URL состоит из Context Path + Servlet Path + PathInfo, где Context Path является префиксом пути, ассоциируемым с контекстом сервлета. Этот сервлет является частью контекста. Если этот контекст является контекстом по умолчанию, находящимся в основном пространстве имен URL Web-сервера, префикс пути является пустой строкой. В противном случае префикс пути начинается с символа /, но не заканчивается символом /. Servlet Path - это участок пути, который прямо соответствует отображению, активизировавшему данный запрос. Этот путь начинается с символа /. PathInfo - это часть пути запроса, не являющаяся частьюContext Path или Servlet Path. |
| **унифицированное имя ресурса (URN)** | Уникальный идентификатор, который идентифицирует сущность, но не показывает, где она расположена. Система может использовать URN для локального поиска сущности перед попыткой найти ее в Web. Этот идентификатор также позволяет изменение Web-ссылки при сохранении возможности нахождения сущности. |
| **ограничитель данных пользователя (user data constraint)** | Определяет, как должны быть защищены данные между клиентом и контейнером. Защита может быть предотвращением несанкционированного изменения данных либо предотвращением перехвата данных. |
| **переменная (variable)** | Элемент данных, имеющий идентифицирующее его имя. Каждая переменная имеет тип (например, int или Object) и область видимости. См. также переменная класса, переменная экземпляра, локальная переменная. |
| **виртуальная машина (virtual machine)** | Абстрактная спецификация для вычислительного устройства, которое может быть реализовано различным образом - программно или аппаратно. Вы компилируете последовательность команд виртуальной машины точно так же, как будто вы компилируете последовательность команд микропроцессора. Виртуальная машина JavaTMсостоит из байтовых команд, набора регистров, стека, "кучи" со сборкой "мусора" и областью для хранения методов. |
| **словарь (vocabulary)** | Обычно компьютерные программы пишутся и компилируются в машинные коды, которые прямо зависят от операционной системы, управляющей микропроцессором в компьютере. Java-платформа смягчает эту зависимость, обеспечивая модель, по которой программы пишутся, компилируются и могут быть переданы по сети и выполнены в любом месте, где присутствует полностью совместимая виртуальная машина. Эта модель обеспечивает дополнительное преимущество в повышении безопасности. Во-первых, потому что программы могут быть проверены виртуальной машиной после того, как они были переданы по сети. И, во-вторых, потому что виртуальная машина может запустить программу в защищенном пространстве, которое предотвратит определенные разрушающие действия. Разработчики программного обеспечения выбирают платформу Java потому, что это уменьшает стоимость и время написания и поддержки программного кода. Им больше не требуется переписывать программы для функционирования на различных компьютерах с различными операционными системами и микропроцессорами. Внедрение приложений Java технологии в компаниях и организациях выгодно, потому что это минимизирует стоимость покупки и модификации различных версий приложений для различных типов компьютеров и серверов внутри их сетей. |
| **void** | Ключевое слово языка JavaTM, используемое в описании метода для указания на то, что метод не возвращает никакого значения. void может также использоваться как нефункциональный оператор. |
| **volatile** | Ключевое слово языка JavaTM, используемое в описания переменной для указания на то, что переменная модифицируется асинхронно несколькими совместно выполняющимися потоками. |
| **wait** | Команда UNIX, которая будет ожидать завершения всех фоновых процессов и отчета об их статусе завершения. |
| **Web-приложение, распределенное(Web application, distributable)** | Web-приложение, использующее технологию J2EE и написанное таким образом, что оно может быть внедрено в Web контейнер, распределенный по нескольким виртуальным машинам Java, работающих на одном хосте или на различных хостах. Дескриптор "размещения" для такого приложения использует распределенный элемент. |
| **Web-компонент (Web component)** | Компонент, обеспечивающий сервисы в ответ на запросы, сервлеты или JSP страницы. |
| **Web-контейнер (Web container)** | Контейнер, обеспечивающий исполнение Web компонентов, удовлетворяющих правилам J2EE архитектуры. Эти правила определяют среду времени выполнения (runtime environment) для Web-компонентов, включая безопасность, совместную работу, управление жизненным циклом, транзакции, внедрение и другие службы. Контейнер, обеспечивающий такие же службы как JSP контейнер и интегрированный взгляд на API платформы J2EE. Web контейнер обеспечивается Web или J2EE сервером. |
| **Web-контейнер, распределенный (Web container, distributed)** | Web-контейнер, который может запускать Web-приложение, помеченное как распределенное, которое выполняется на нескольких виртуальных машинах Java, работающих на одном или на различных хостах. |
| **Web-сервер (Web server)** | Программа, обеспечивающая службы для доступа в Internet, Intranet, или Extranet. Web-сервер содержит Web-сайты, обеспечивая поддержку HTTP и других протоколов и выполняет серверные программы (такие как CGI-скрипты или сервлеты), которые выполняют определенные функции. Например, Web-контейнер обычно основан на Web-сервере для обеспечения обработки сообщений HTTP. Архитектура J2EE предполагает, что Web-контейнер предоставляется Web-сервером от одного и того же поставщика, т.е. не определяя правила между этими двумя сущностями. Web-сервер может предоставлять один или много Web-контейнеров. |
| **Web-службы (Web Services)** | Свободно собранные программные компоненты, способные взаимодействовать между собой по многочисленным сетям для предоставления определенного результата конечному пользователю. Во время работы они используют развивающуюся группу стандартов, которые определяют их (служб) описание и взаимодействие, таких как SOAP (простой протокол доступа к объектам), UDDI (универсальное обнаружение и интеграция описаний), XML (открытый язык меток), WSDL (язык описания Web-служб). |
| **while** | Ключевое слово языка JavaTM, используемое для определения цикла, который повторяет блок инструкций. Условие продолжения цикла указывается как часть оператора while. |
| **общедоступные для чтения файлы (world readable files)** | Файлы в файловой системе, которые могут быть просмотрены (прочтены) любым пользователем. Например: файлы, расположенные на Web-серверах могут быть просмотрены пользователями Internet, если права файлов были установлены как "доступные для чтения". |
| **wrapper** | Объект, который инкапсулирует и уполномочивает другой объект на изменение своего интерфейса или поведения определенным образом. |
| **язык описания Web-сервисов (Web Services Description Language - WSDL)** | XML язык, который используется для описания Web-сервиса и для определения способа общения с Web-сервисом. |
| **World Wide Web - WWW** | "Всемирная паутина". Сеть систем и данных в них, т.е. часть Internet. См. также Internet. |
| **расширяемый язык меток (Extensible Markup Language - XML)** | Расширяемый язык разметки текстов (или текстовых документов). Дескрипторы размещения J2EE выражены при помощи XML. |