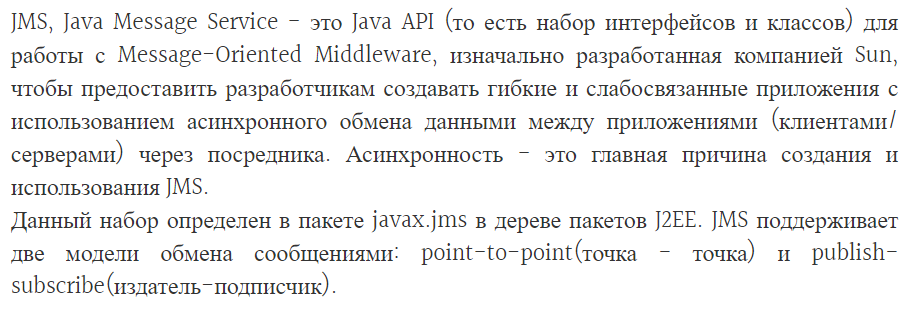
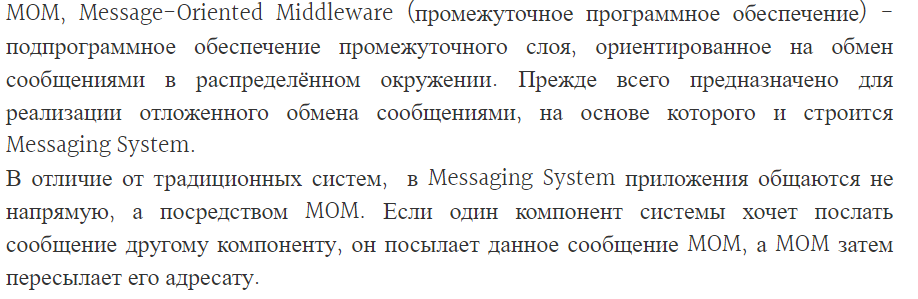
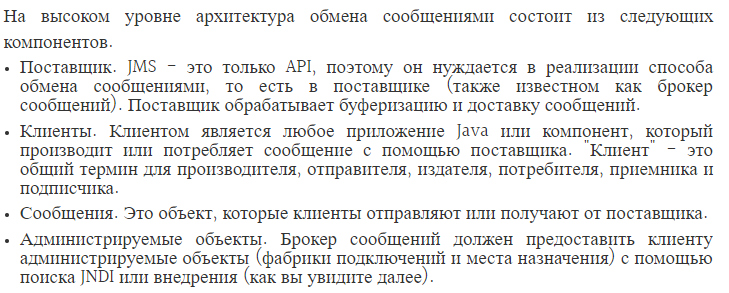
1. **Что такое MOM, JMS?**

**J**ava **M**essaging **S**ystem(**JMS**) представляет собой универсальный интерфейс обобщенного Java API для корпоративных приложений (КИС), ори­енти­рованных на работу с помощью *сообщений*(*message-oriented application programming*) с целью обеспечения их независимости от конкретных реализаций служб обработки сообщений.





1. **Поясните принцип работы MOM. Варианты архитектуры.**



1. **Объясните принцип работы режима point-to-point (P2P).**

Модель передачи сообщений «точка-точка» позволяет клиентам JMS посылать и принимать сообщения как синхронно, так и асинхронно, через виртуальные каналы, известные как очереди (queues). Модель передачи сообщений p2p традиционно основывается на модели вытягивания (pull), или опроса (poll), в которой сообщения запрашиваются из очереди вместо автоматического выталкивания клиенту.

У очереди может быть несколько получателей, но только один получатель может принимать каждое отдельное сообщение. Провайдер JMS заботится о раздаче сообщений клиентам JMS, гарантируя, что каждое сообщение будет использовано только одним клиентом.

Не гарантирует доставку сообщений в определенном порядке (нет временной зависимости между отправителем и получателем)

1. **Объясните принцип работы publish/subscribe (pub/sub).**

При передаче сообщений «издание-подписка» один поставщик может посылать сообщение многим потребителям через виртуальный канал, называемый темой (topic). Потребители могут выбрать подписку (subscribe) на любую тему. Все сообщения, направляемые в тему, передаются всем потребителям данной темы. Каждый потребитель принимает копию каждого сообщения.

Модель передачи сообщений pub/sub, по существу, представляет собой модель, основанную на проталкивании (pushbased), где сообщения автоматически распространяются по потребителям без необходимости запрашивать, или вытаскивать, новые сообщения из темы. В модели передачи сообщений pub/sub производитель, посылающий сообщение, не зависит от потребителей, принимающих сообщение. Дополнительно клиенты JMS, использующие pub/sub, могут устанавливать долговременные подписки, позволяющие потребителям отсоединиться и позже снова подключиться, и забрать сообщения, поступившие во время отключения связи.

5. Поясните запуск и настройку брокеров в Open MQ.

1. **Что такое администрируемые объекты в MOM?**

Администрируемые объекты - объекты, которые конфигурируются административно, а не программно.

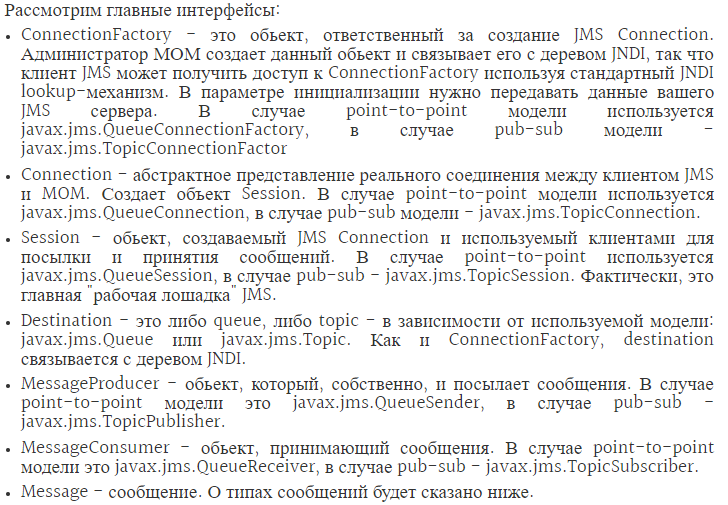
Нужно настроить эти объекты и сделать их доступными в пространстве имен JNDI

Типы объектов:

фабрики соединений — используются клиентами для создания подключения к пункту назначения;

места назначения —точки распространения сообщений, которые получают, хранят и распространяют сообщения; (P2P) (pub-sub).

**7. Перечислите классы и интерфейсы JMS API.**



8. Охарактеризуйте администрируемые объекты JMS.

**9. Структура сообщения JMS.**

Сообщения соостоят из трех частей: заголовка сообщения, раздела свойств и собственно тела сообщения.

Заголовок сообщения содержит дополнительную информацию, которую разработчик может использовать в своем приложении. JMS предоставляет get и set методы для каждого поля заголовка. Некоторые из них устанавливаютя  автоматически, другие могут быть использованы разработчиком приложения.

**JMSDestination** – содержит имя destination, в который посылается сообщение.

**JMSDeliveryMode** – определяет, является ли сообщение сохраняемым или нет. Если оно сохраняемо, то будет сохранено MOM в базе данных (или в файле), а потому оно переживет "гибель" системы, в отличие от несохраняемого сообщения.

**JMSExpiration** – определяет, когда сообщение устареет и будет удалено из системы. По умолчанию сообщение не устаревает никогда.

**JMSPriority** – как и следует из названия, определяет приоритет сообщения (от 0 до 9). По умолчанию равно 4.

**JMSMessageID** – уникальный идентификатор сообщения

**JMSTimestamp** – содержит информацию, когда именно MOM приняла сообщение от producer.

**JMSCorrelationID** – может быть использовано разработчиком для согласования сообщений: например, если вы хотите переслать ряд сообщений, обьединенных в одну логическую группу (такую как набор товаров в заказе, при этом в каждое сообщение о товаре вы можете добавить в данное поле заголовка номер заказа).

**JMSReplyTo** – может быть использовано разработчиком для того, чтобы consumer знал, кому (то есть в какой destination) при желании отсылать ответ.

**JMSType** – поле может быть использовано разработчиком для того, чтобы дать приложению информацию, как обращаться с данным сообщением. Тип здесь понимается как application-specific type, а не тот, что использован выше в разделе "типы сообщений".

**JMSRedelivered** – устанавливается, если сообщение не было доставлено с первой попытки – например, в случае, когда consumer не подтвердил получение сообщения (если он должен был подтвердить, конечно)

Раздел свойств содержит пары "ключ-значение", которые могут быть использованы для пересылки определенных данных между producer и consumer. В качестве значений могут быть использованы примитивные типы (boolean, byte, float, double, short, int, long), а так же строки (java.lang.String).

**10. Типы сообщений JMS.**

*Сообщение* в **JMS** - это Java-объект, состоящий из **заголовка** (*header*) и **тела** (*body*). В *заголовке* находится служебная информация, а *тело сообщения* содержит в себе пользователь­ские данные.

**JMS** API определяет несколько типов сообщений:

**StreamMessage** – сообщение, содержащее сериализованный поток обьектов

**MapMessage** – сообщение, содержащее пары "ключ-значение"

**TextMessage** – сообщение, содержащее строку

**ObjectMessage**– сообщение, содержащее сериализованный обьект

**ByteMessage** – сообщение, содержащее поток байтов

Кроме того, некоторые имплементации (например, OpenFusion и WebLogic) предоставляют еще один "почти стандартный" тип:

**XMLMessage** – расширение TextMessage, используется для доставки XMLсообщений

Все типы сообщений являются подклассами javax.jms.Message.

11. В чем разница при получении сообщений синхронно и асинхронно?

12. Приведите алгоритм обмена сообщениями

**13. В чем разница между долговременной и – не долговременной подпиской при создании topic?**

Долговременная – гарантирует доставку, даже если подписчики не активны

Не долговременная – получают только активные подписчики (сообщение удаляется после подтверждения)

**14. Перечислите и охарактеризуйте модели подтверждения доставки сообщений.**

**Уведомления о доставке сообщений** - это механизм, при котором провайдер оповеща­ется о том, было ли сообщение доставлено адресату. Тип уведомления обычно задается при создании объекта сессии. Всего типов уведомления три:

- **Session.DUPS\_OK\_ACKNOWLEDGE** - «ленивое» уведомление о доставке. При нем сводится к минимуму работа по нахождению дубликатов сообщения. Должно использоваться только в том случае, если ожидается появление дублирующих сообщений.

- **Session.AUTO\_ACKNOWLEDGE** - автоматическое уведомление об отправке при получении сообщения адресатом.

- **Session.CLIENT\_ACKNOWLEDGE**- уведомление об отправке устанавливается вручную вызовом метода **acknowledge( )** объектов **Message**.

**15. Что такое и как используются селекторы сообщений (фильтры)?**

JMS предоставляет механизм под названием селектор сообщений (message selector), позволяющий приложению фильтровать и классифицировать принимаемые сообщения. Селектор сообщений — это строка, содержащая выражение, основанное на спецификации SQL92. Селектор вызывается каждый раз при получении очередного сообщения. Сообщения, прошедшие отбор по критерию, становятся доступными приложению, остальные отбрасыва­ются. Выборка основана на сравнении полей заголовка и свойств сообщения.

**16. Что такое MDB?**

Message-Driven Bean - является корпоративным компонентом, который позволяет приложению обрабатывать сообщения асинхронно. Асинхронная обработка реализуется через пул объектов, из которых на обработку сообщения сервер выделят объекты при необходимости.  
Для реализации MBD достаточно унаследовать бин от интерфейса javax.jms.MessageListener, реализуя метод onMessage(), и аннотировать соответствующим образом класс.