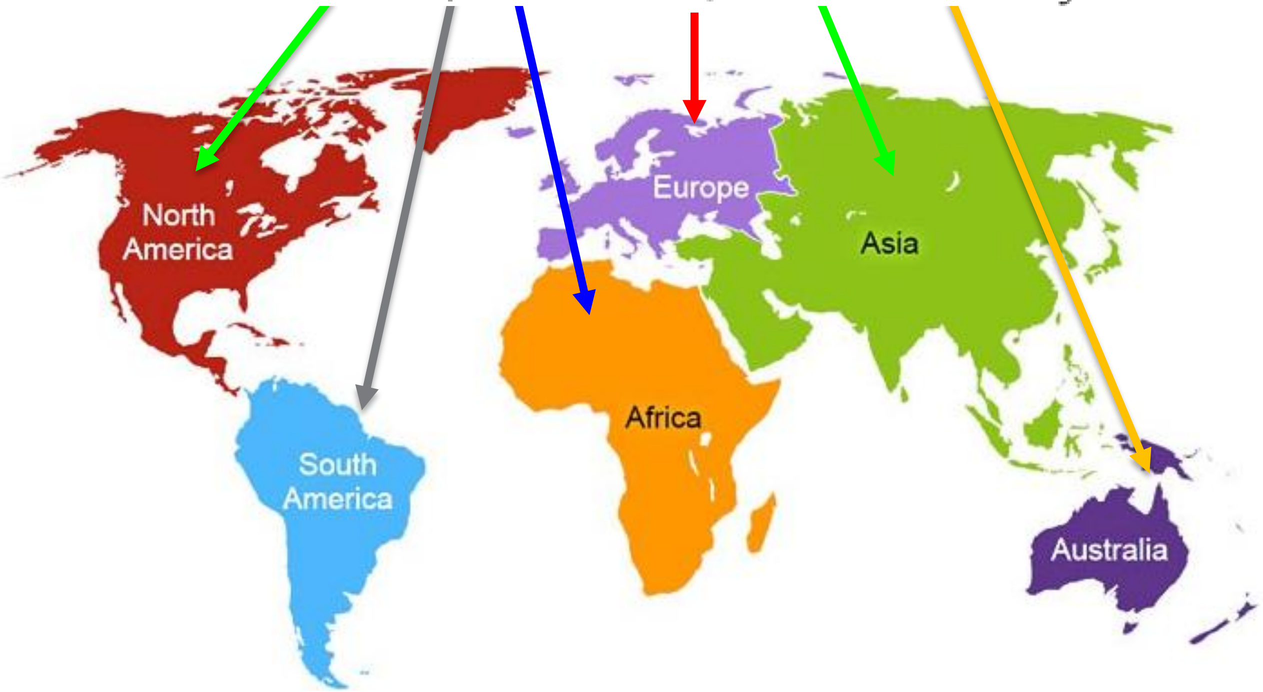
H2K Infosys, LLC

Dream, Strive & Achieve Victory



H2K

Live - Green - Everywhere

H2K Infosys is E-Verify business based in Atlanta, Georgia – United States
Providing Online IT training services world wide.

www.H2KINFOSYS.com

USA - +1-(770)-777-1269, UK - (020) 3371 7615
Training@H2KInfosys.com / H2KInfosys@Gmail.com

H2K INFOSYS PROVIDES WORLD CLASS SERVICES IN

IT Trainings with Real time Project Work for Corporates & Individuals

Special IT Training for MS Students in US

Software Design, Development, QA – Manual, Automation & Performance Testing, Maintenance.

IT Staff Augmentation

Job Placement Assistance

Tech Support

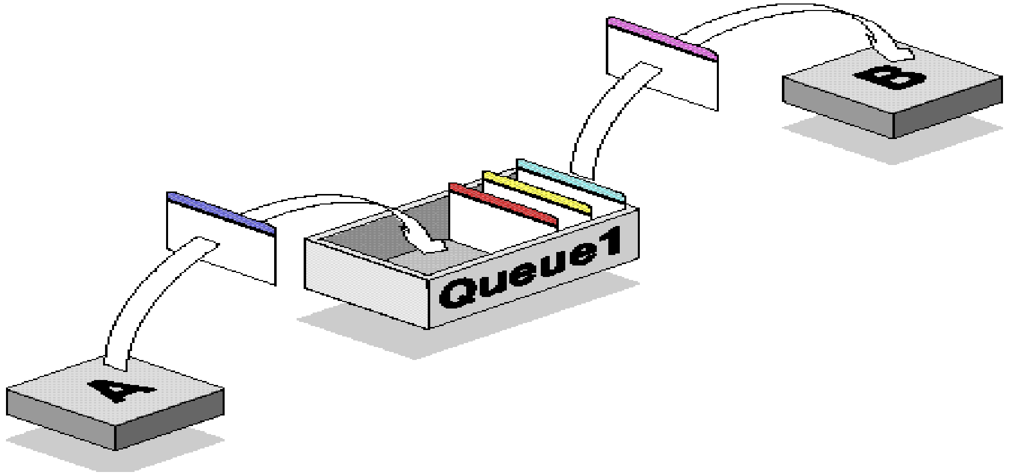
Communication Patterns……

Synchronous: App sends request, then blocks until request is processed

- Requires service available at EXACTLY same time as client needs service
- Similar to a phone conversation
- Example: a web service that talks to a DB and returns XML or JSON to the invoking application, all in one call

Asynchronous: App sends request and proceeds with its normal operations

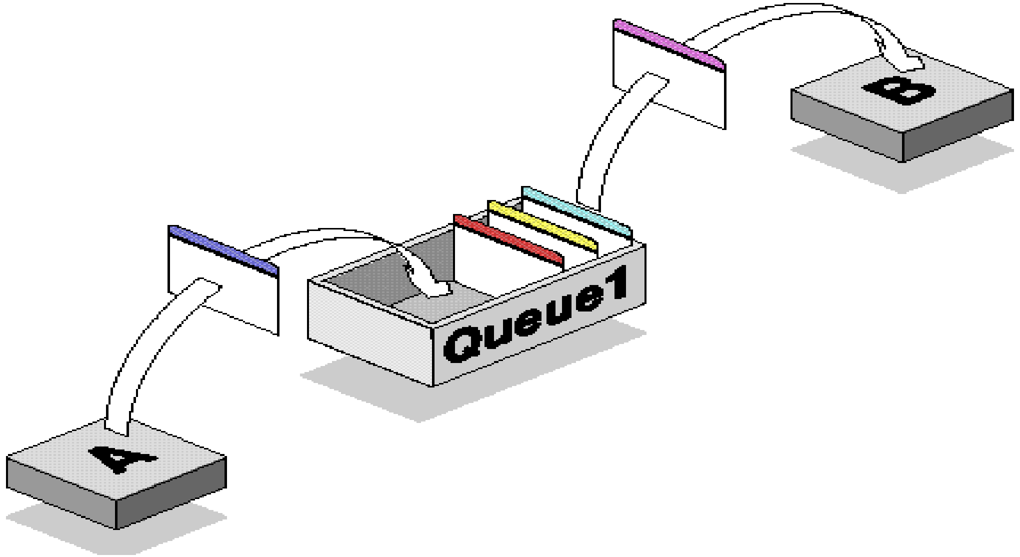
- Consuming service need not be available at the same time as client sends request
- Similar to email or texting - “fire and forget”
- The sending and receiving applications do not have to be on the same platform or even necessarily know about each other



Why Messaging……

Importance of Asynchronous Messaging.

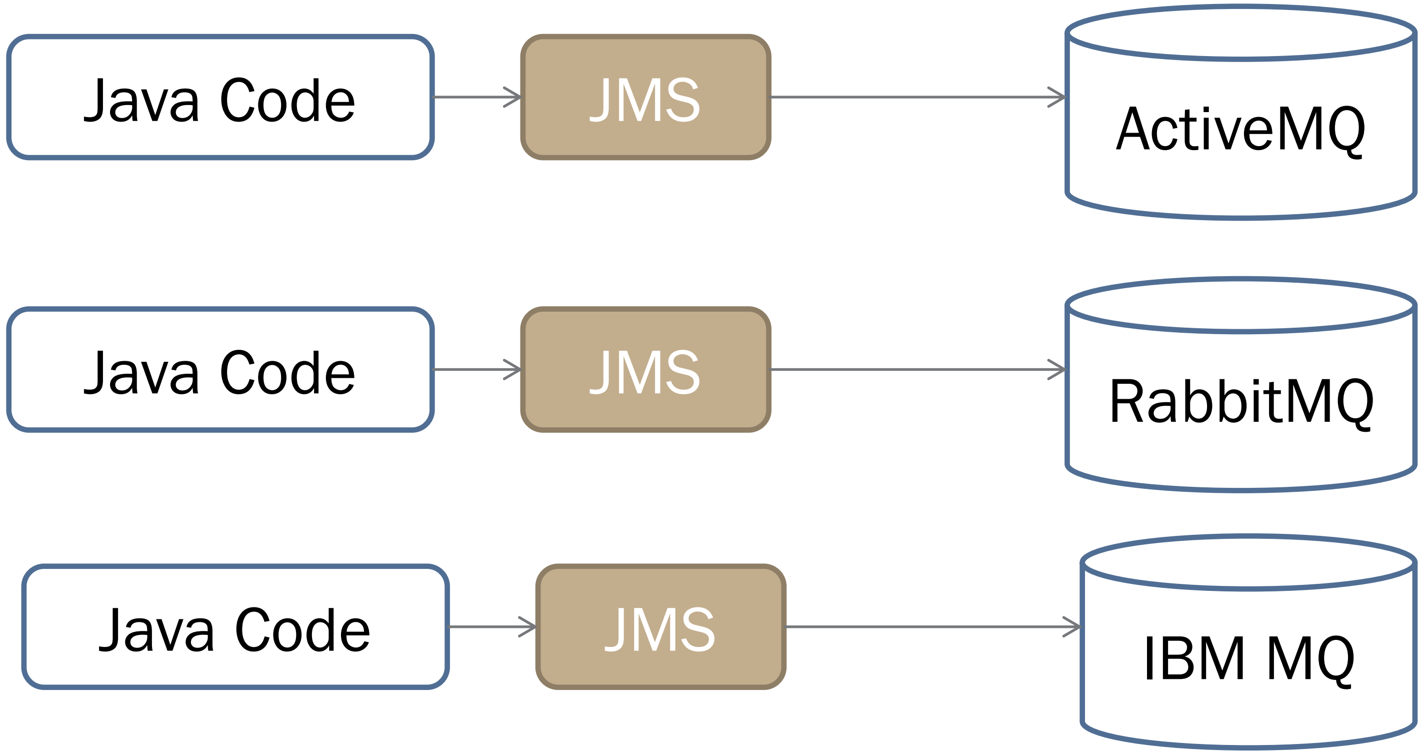
- Messaging is a method of communication between software components or applications.
- Messaging enables distributed communication that is loosely coupled. A component sends a message to a destination, and the recipient can retrieve the message from the destination. However, the sender and the receiver do not have to be available at the same time in order to communicate.
- In fact, the sender does not need to know anything about the receiver; nor does the receiver need to know anything about the sender.



JMS API…….

What is JMS API?

- The Java Message Service is a Java API that allows applications to create, send, receive, and read messages.
- It also strives to maximize the portability of JMS applications across JMS providers in the same messaging domain.



JMS API

The JMS API enables communication that is not only loosely coupled but also

- **Asynchronous**. A JMS provider can deliver messages to a client as they arrive; a client does not have to request messages in order to receive them.
- **Reliable**. The JMS API can ensure that a message is delivered once and only once.

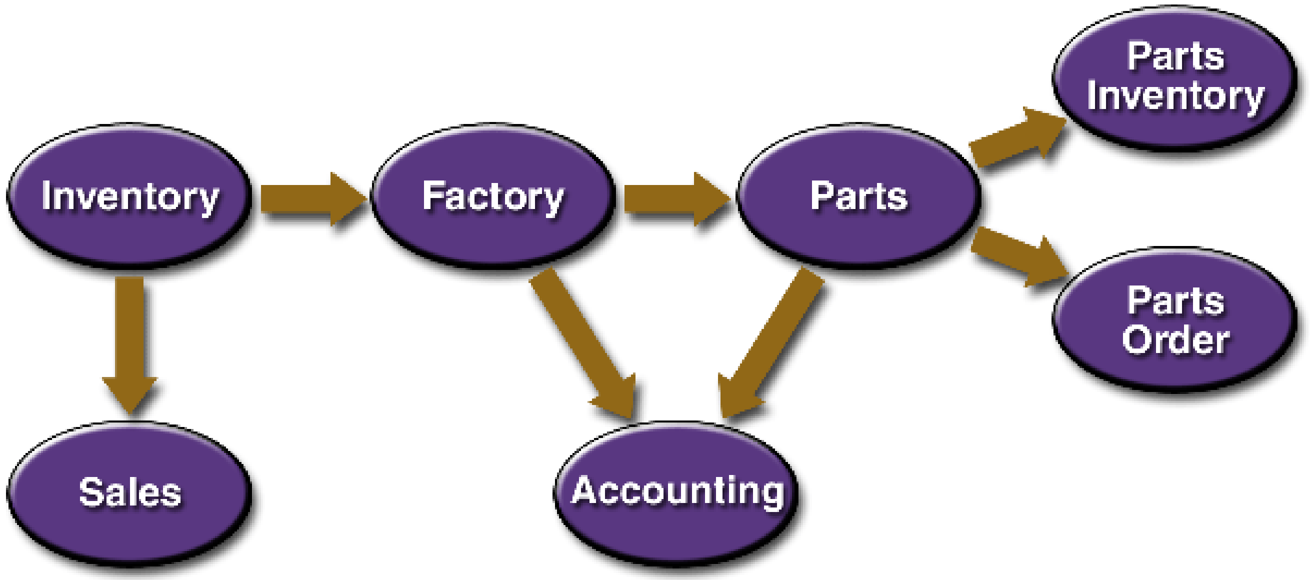


Figure 1.1. Messaging in an Enterprise Application

Messaging Domains

Point to Point Messaging Domain:

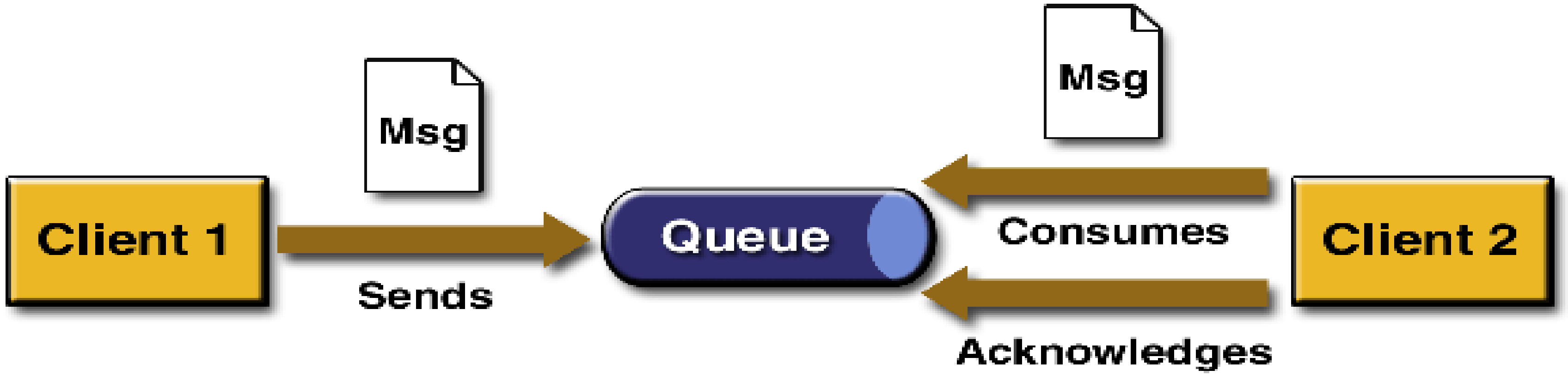


Figure 2.2 Point-to-Point Messaging

- Each message is addressed to a specific queue, and receiving clients extract messages from the queue(s) established to hold their messages.
- Queues retain all messages sent to them until the messages are consumed or until the messages expire.
- This is same as one person texting to another person.
- Queue : Bucket which holds the in-transit messages between two client.

- Each message has only one consumer.
- A sender and a receiver of a message have no timing dependencies. The receiver can fetch the message whether or not it was running when the client sent the message.
- The receiver acknowledges the successful processing of a message

Messaging Domains

Publish/Subscribe Messaging Domain:

- This Messaging domain is used when same message is intended for multiple recipient.
- Message is sent to Topic by sender application and same message will be delivered to all interested parties.
- “Interested Parties” are referred as “Subscribers” and Sender application is referred as “Publisher”
- Publishers and subscribers have a timing dependency. A client that subscribes to a topic can consume only messages published after the client has created a subscription, and the subscriber must continue to be active in order for it to consume messages.
- The JMS API relaxes this timing dependency to some extent by allowing clients to create *durable subscriptions*. Durable subscriptions can receive messages sent while the subscribers are not active. Durable subscriptions provide the flexibility and reliability of queues but still allow clients to send messages to many recipients.

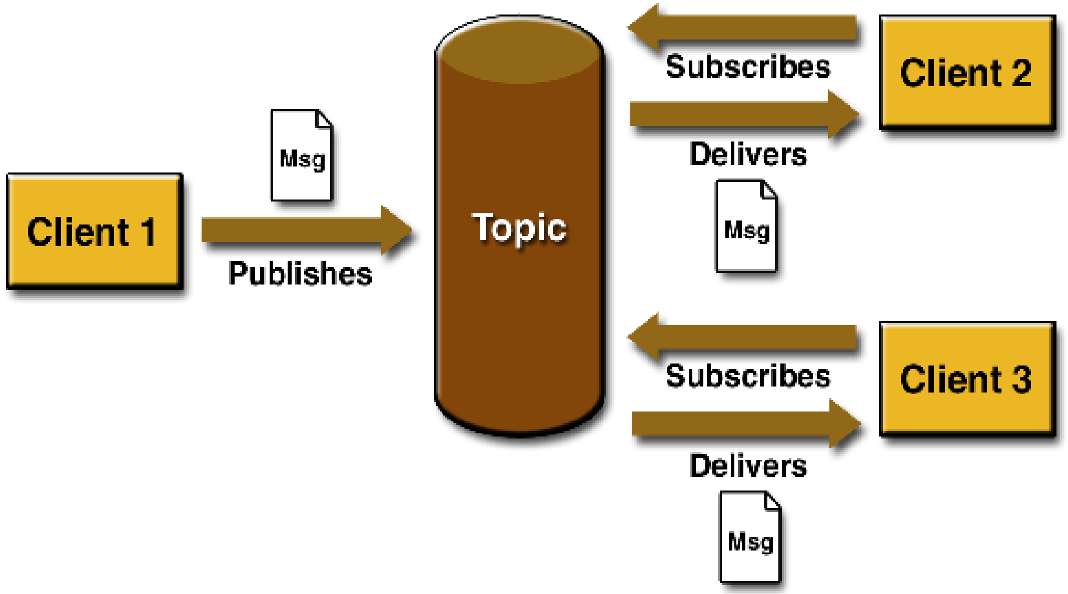


Figure 2.3 Publish/Subscribe Messaging

The JMS API Programming Model

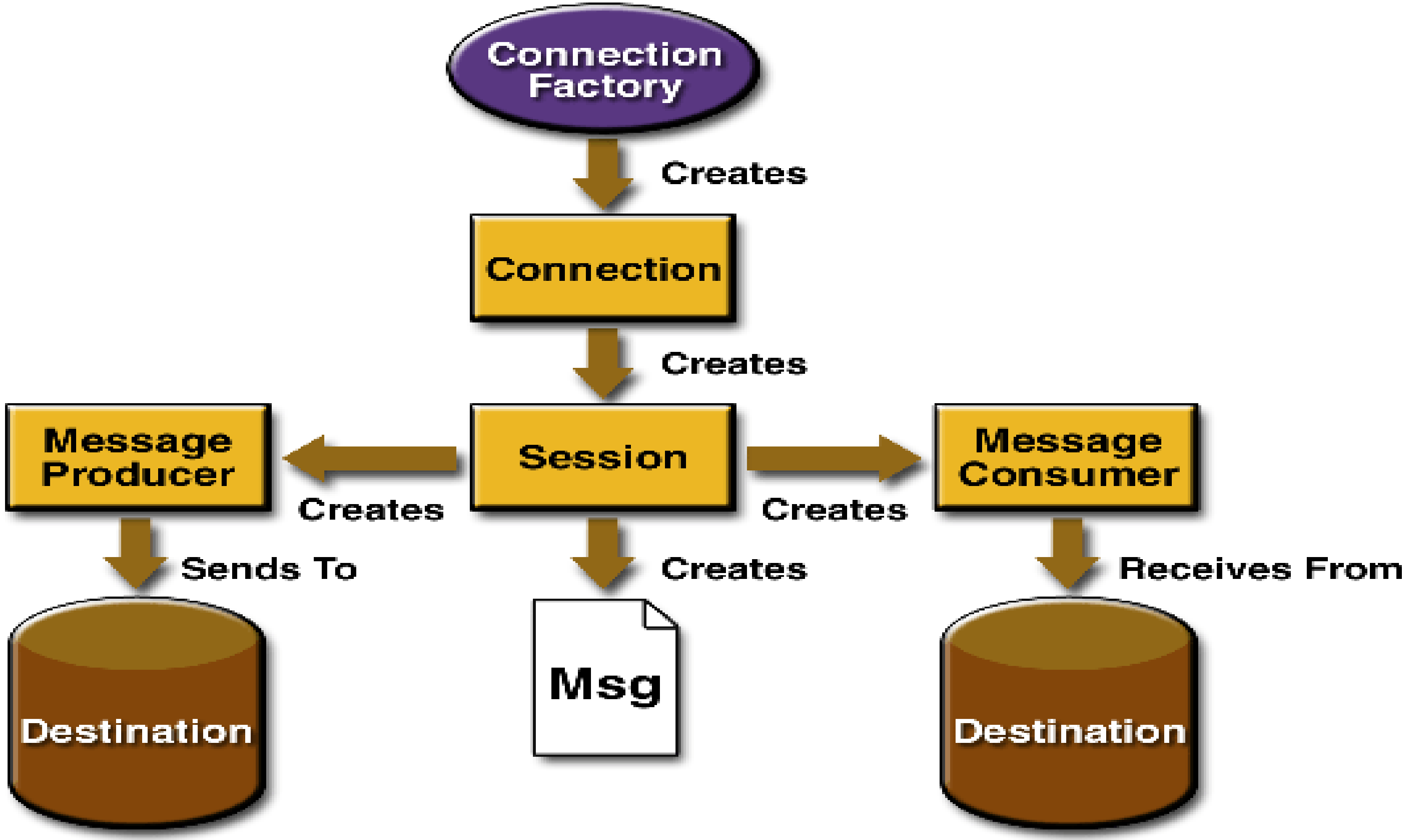
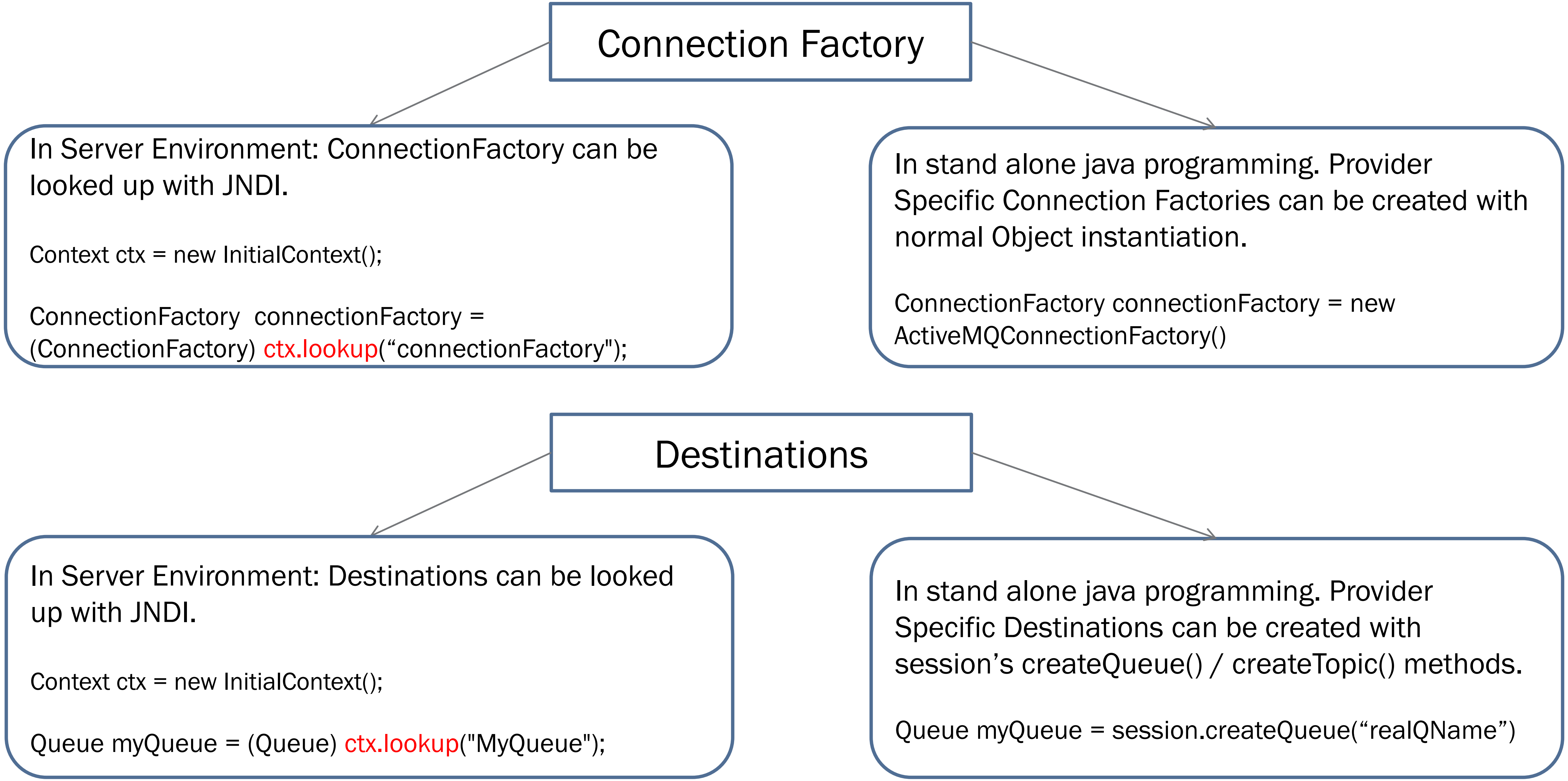


Figure 3.1 The JMS API Programming Model

The JMS API Programming Model



Connection Factory

In Server Environment: ConnectionFactory can be looked up with JNDI.

```
Context ctx = new InitialContext();

ConnectionFactory  connectionFactory =
(ConnectionFactory) ctx.lookup(“connectionFactory");
```

In stand alone java programming. Provider Specific Connection Factories can be created with normal Object instantiation.

```
ConnectionFactory connectionFactory = new
ActiveMQConnectionFactory()
```

Destinations

In Server Environment: Destinations can be looked up with JNDI.

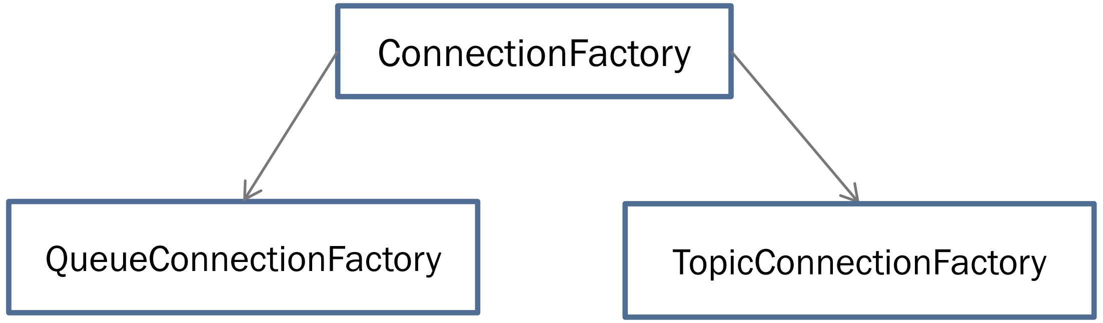
```
Context ctx = new InitialContext();

Queue myQueue = (Queue) ctx.lookup("MyQueue");
```

In stand alone java programming. Provider Specific Destinations can be created with session’s createQueue() / createTopic() methods.

```
Queue myQueue = session.createQueue(“realQName”)
```

The JMS API Programming Model

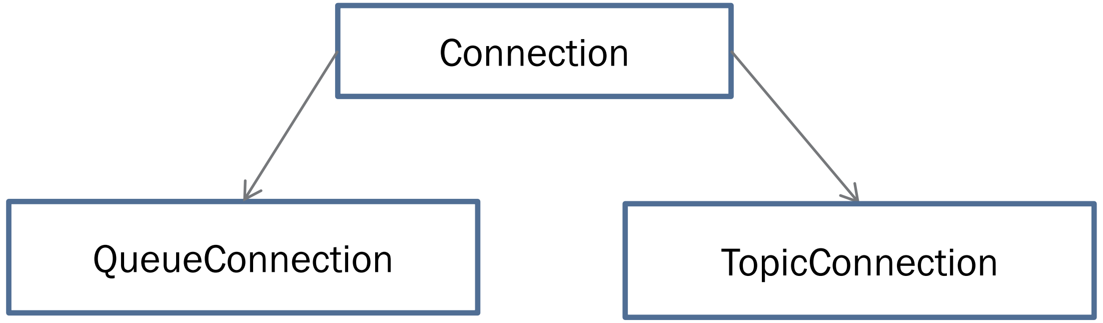
Separate Interfaces based on Destination type.



ConnectionFactory

QueueConnectionFactory

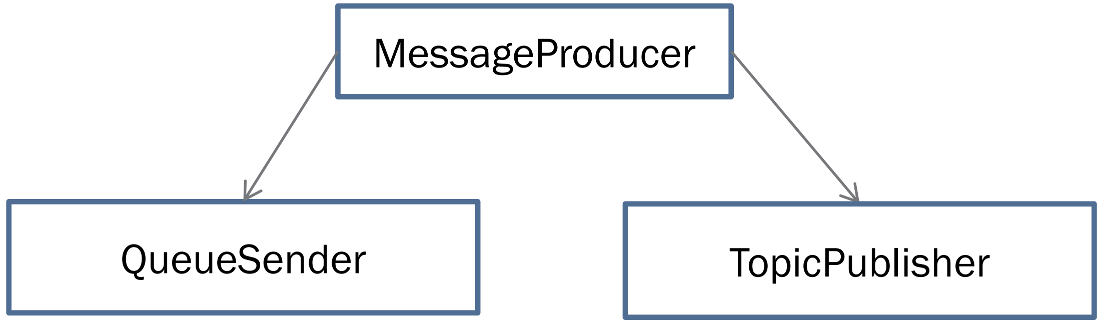
TopicConnectionFactory



Connection

QueueConnection

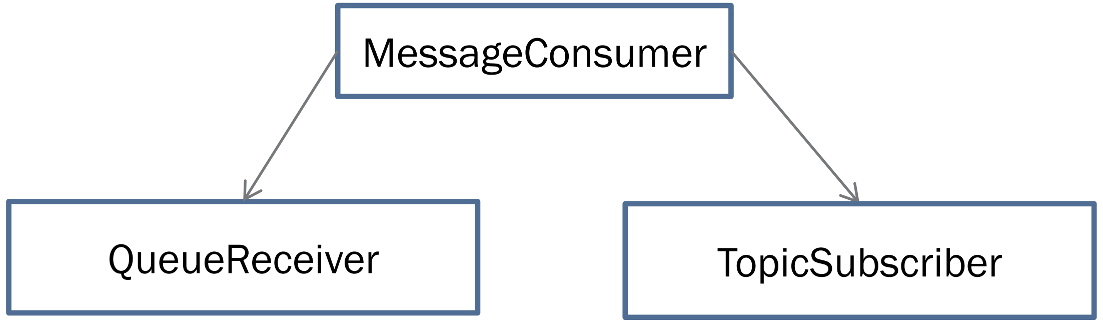
TopicConnection



MessageProducer

QueueSender

TopicPublisher



MessageConsumer

QueueReceiver

TopicSubscriber

The JMS API Programming

Write a Program to Send a Message to Queue

Step 1. Create ConnectionFactory instance

Step 2. Create Connection from connectionFactory

Step 3. Start the Connection

Step 4. Create a JMS Session from Connection

Step 5. Create a Queue object from Session

Step 6 Create a Producer object from Session

Step 7. Create a TextMessage from Session

Step 8. Send the Message to Queue.

The JMS API Programming

Receive a message using MessageListener:

Step 1. Create ConnectionFactory instance

Step 2. Create Connection from connectionFactory

Step 3. Start the Connection

Step 4. Create a JMS Session from Connection

Step 5. Create a Queue object from Session

Step 6. Class should implement MessageListener (should have onMessage()) method

Step 7 Create a Consumer from session

Step 8 add current class as MessageListener with addMessageListener() method

Session.AUTO…..

Session Option	Meaning
Session.AUTO_ACKNOWLEDGE	With this acknowledgment mode, the session automatically acknowledges a client's receipt of a message either when the session has successfully returned from a call to receive or when the message listener the session has called to process the message successfully returns.
Session.CLIENT_ACKNOWLEDGE	With this acknowledgment mode, the client acknowledges a consumed message by calling the message’s acknowledge method.
Session.DUPS_OK_ACKNOWLEDGE	This acknowledgment mode instructs the session to lazily acknowledge the delivery of messages.
Session.SESSION_TRANSACTED	If the session is transacted, developer has to take care of commit and rollback logic over session.

Summary:

- What is Messaging, why we use it?
- What is JMS API?
- What are the messaging domains?
- How JMS API Programmable Model look?
- Creating Connection Factories and Other JMS Objects
- Lastly, how to send and receive messages from Destinations

Questions:

Remember: No Question is Dumb Question.

Thank you

Your feedback is highly important to improve our course material and teaching methodologies. Please email your suggestions.

USA +1-(770)-777-1269
Training@H2KInfosys.com

UK (020) 3371 7615
H2KInfosys@Gmail.com

H2K Infosys is e-Verified business based in Atlanta, Georgia – United States

H2K Infosys acknowledges the proprietary rights of the trademarks and product names of other companies mentioned in this document.

DISCLAIMER

H2K Infosys, LLC (hereinafter "H2K") acknowledges the proprietary rights of the trademarks and product names of other companies mentioned in any of the training material including but not limited to the handouts, written material, videos, power point presentations, etc. All such training materials are provided to H2K students for learning purposes only. H2K students shall not use such materials for their private gain nor can they sell any such materials to a third party. Some of the examples provided in any such training materials may not be owned by H2K and as such H2K does not claim any proprietary rights for the same. H2K does not guarantee nor is it responsible for such products and projects. H2K acknowledges that any such information or product that has been lawfully received from any third party source is free from restriction and without any breach or violation of law whatsoever.