# Java Message Service - What and Why?

Bill Kelly, Silvano Maffeis

SoftWired AG, Zürich

info@softwired-inc.com

# Agenda

- Make or Buy?
- Middleware Taxonomy, Messaging
- Java Message Service
  - Overview
  - Features of note
  - Its place in J2EE, EJB
  - Products
- For Further Information

# Middleware - Make or Buy?

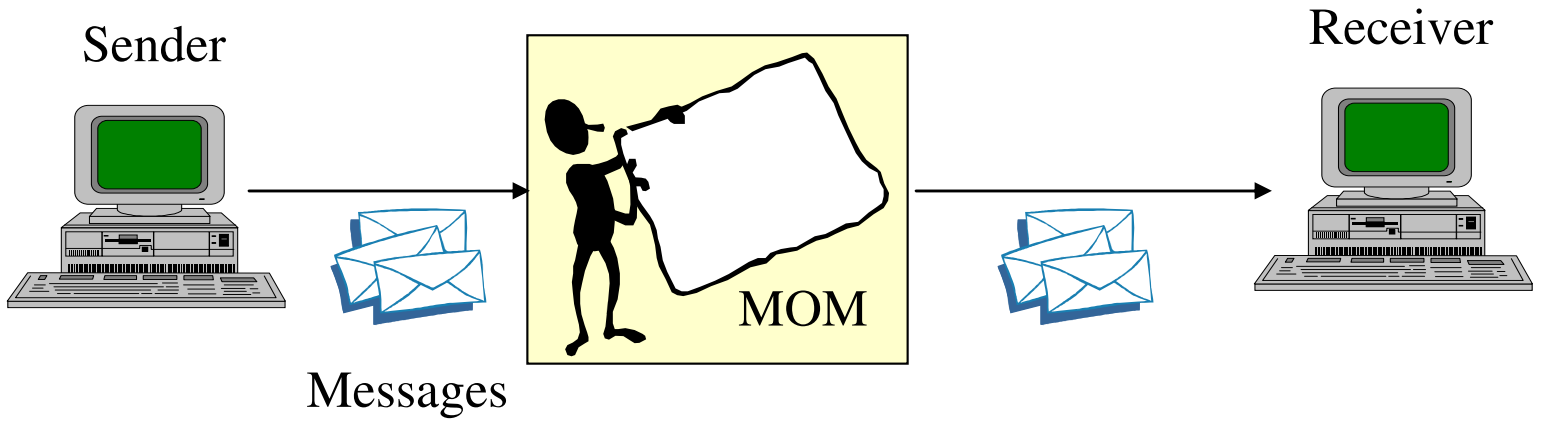
- Information systems are increasingly based on distributed architectures.
- Mobile and other new devices must be integrated: Server, PC, Laptop, PDA, Cell Phone, ...
- New transport protocols (e.g. wireless), different qualities of service (best-effort, guaranteed, ...).
- Systems become more complex, deadlines shorter. “Write-it-yourself” less an option.

# Middleware Taxonomy

- Client/Server
  - a.k.a. RPC; procedure-oriented
- Distributed Objects
  - Object-oriented.  CORBA, DCOM, RMI
- Message Oriented Middleware (MOM, Messaging)
  - “Connectionless”, “asynchronous”
  - Best known through message queuing

# Messaging

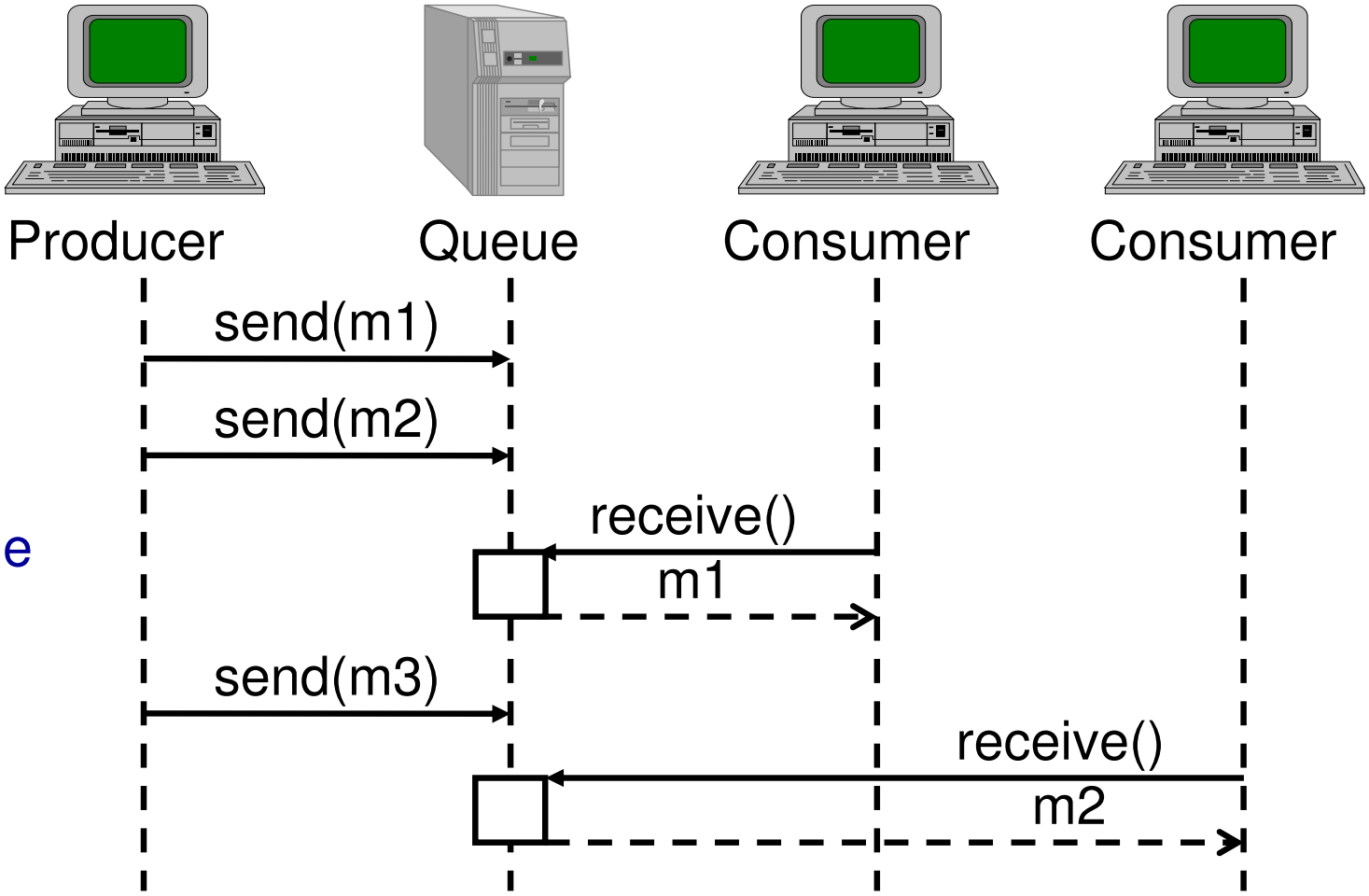
- Messaging is a model in which applications are *loosely coupled* through the exchange of *self-describing* messages.
- Message Oriented Middleware (MOM) encompasses *publish/subscribe* and *message queuing* communications.



# Message Queuing

Put message into queue

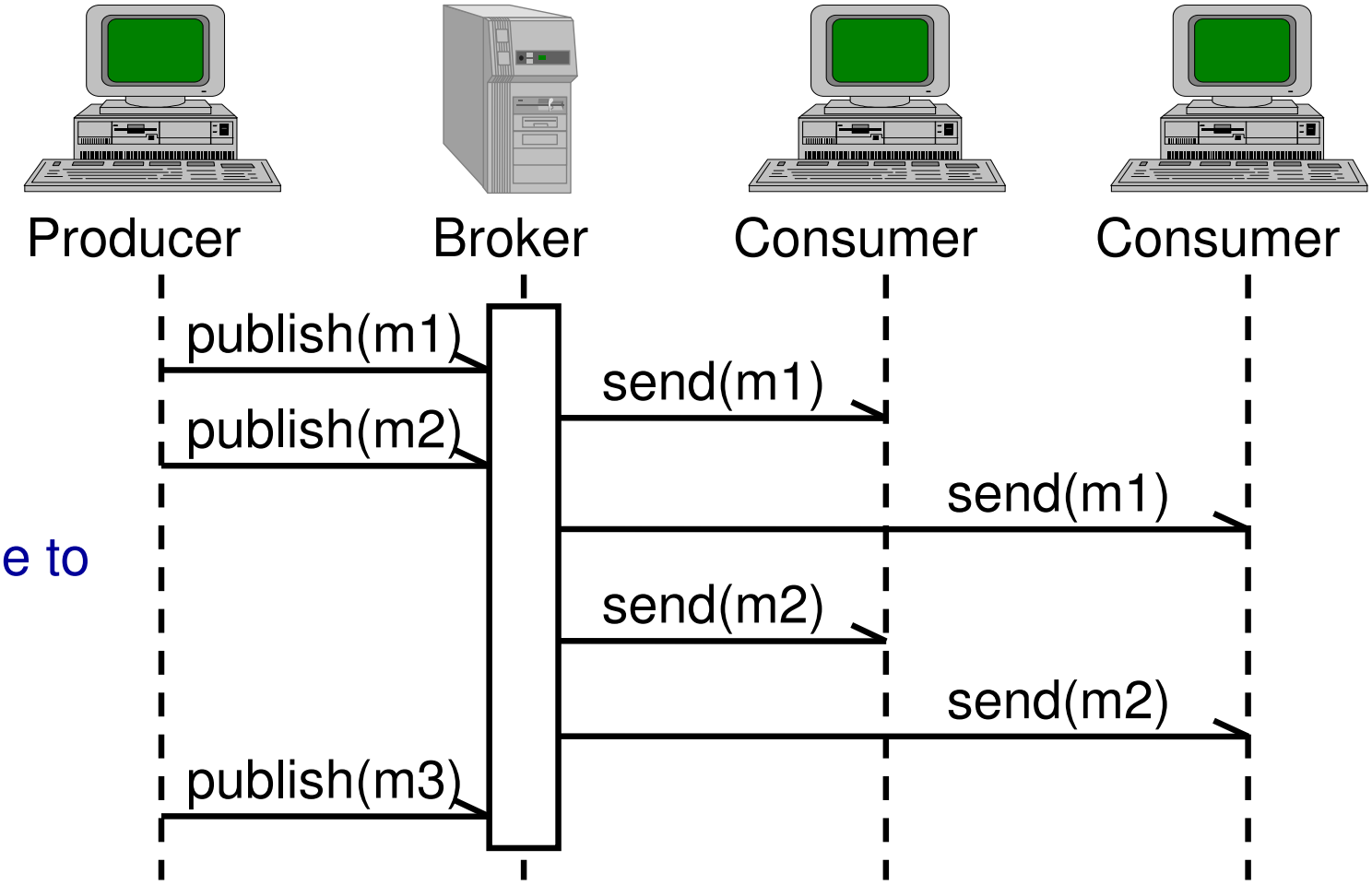
Consume message



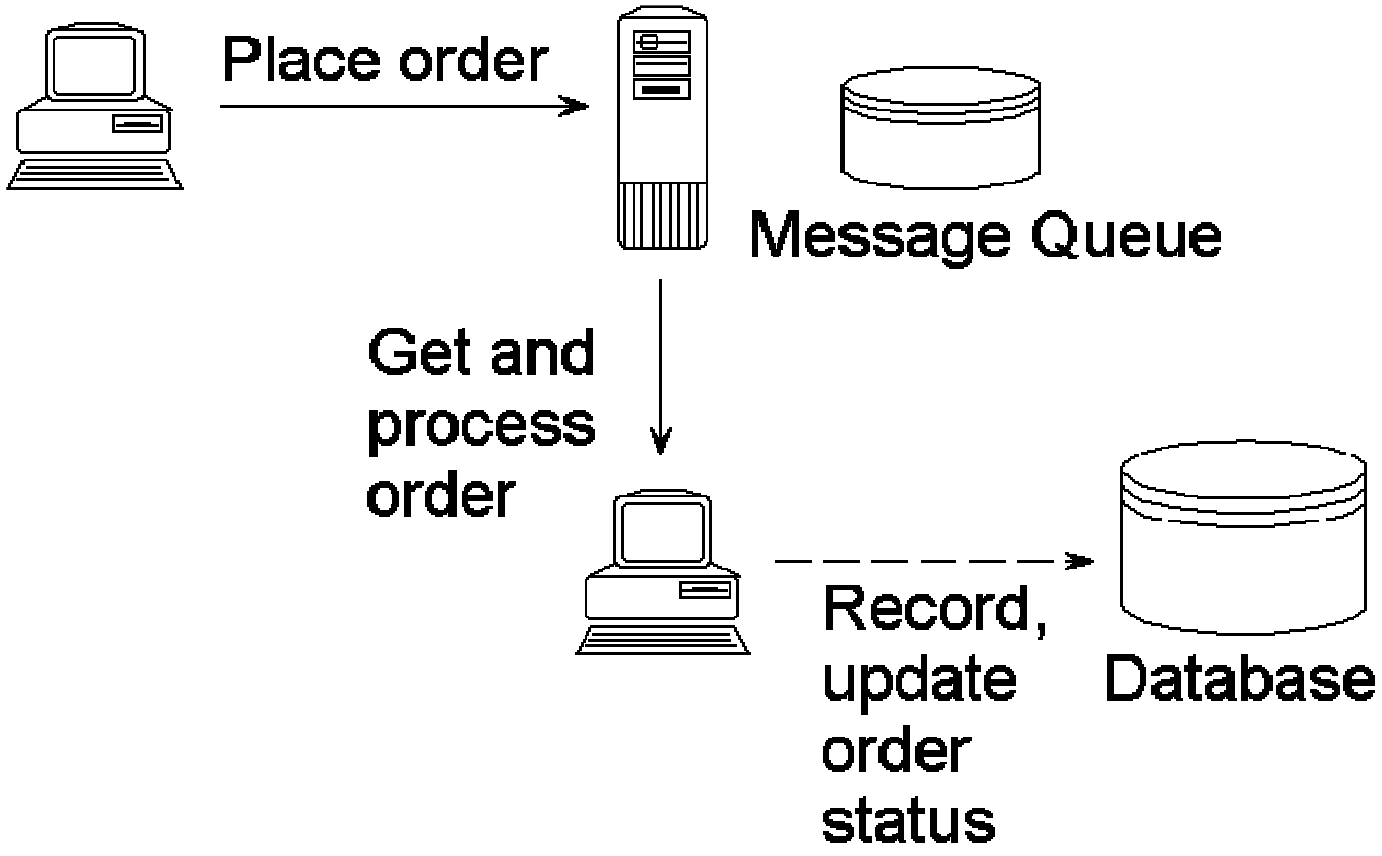
# Publish/Subscribe

Pass message to broker

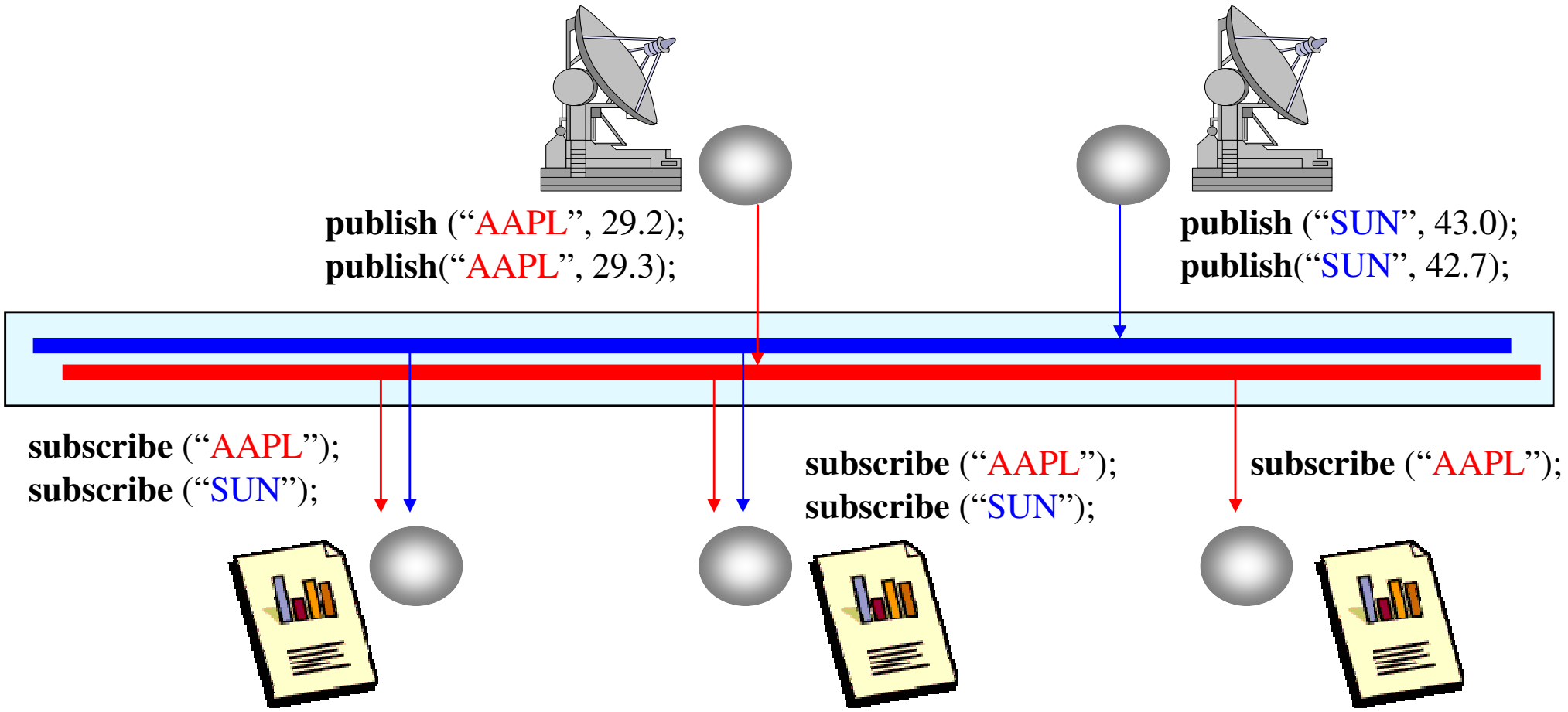
Dispatch message to all consumers



# Message Queuing Application



# Publish/Subscribe Application



# JMS Overview

Goals of Java Message Service (JMS):

- Standardized API for Messaging in Java
- System-independent API for development of heterogeneous, distributed applications
- Use of arbitrary Java objects as messages
- Natural fit with XML messages (Extensible Markup Language) through data-centric model.
- Dual API for the two models:
  - Point to point (Message Queuing)
  - Publish-Subscribe

# JMS Functionality

- Message Formats
  - TextMessage, BytesMessage, MapMessage (Hashtable), StreamMessage, ObjectMessage
- Quality of Service
  - Persistent/non-persistent delivery
  - Priorities, time to live, transactions
- Threaded programming model
- Outside the spec:
  - Security services
  - Management services

# JMS Publisher Example

***Initialize JMS:***

```
session    = connection.createTopicSession(
   transacted, ackMode);
topic      = session.createTopic(”quotes");
publisher  = session.createPublisher(topic);
```

***Create a message:***

```
message  = session.createTextMessage(…);
```

***Send a message:***

```
publisher.publish(message);
```

# JMS Subscriber Example

***Initialize JMS:***

```
session    = connection.createTopicSession(...);
topic      = session.createTopic(”quotes");
subscriber= session.createSubscriber(topic);
```

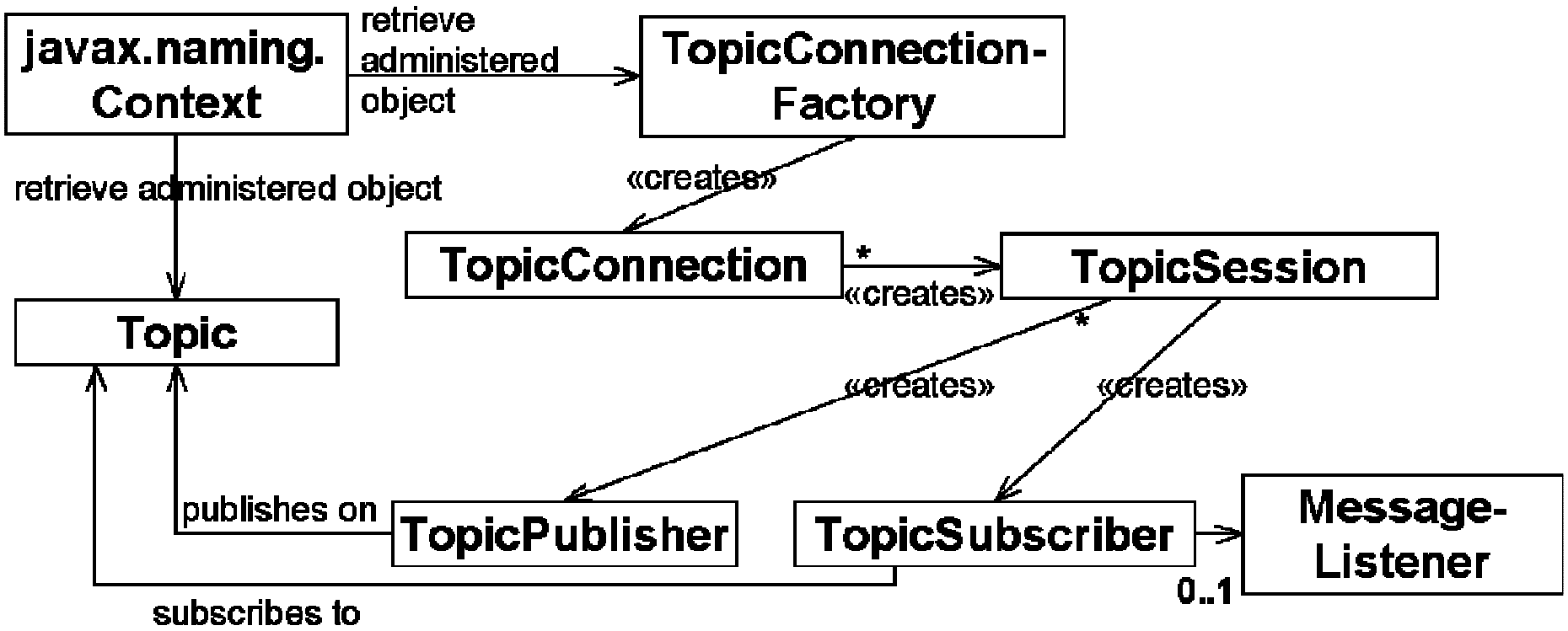
***Create a consumer:***

```
consumer = new MyConsumer();
subscriber.setMessageListener(consumer);
```

***Consumer receives messages via listener:***

```
void onMessage(Message message);
```

# JMS Pub-Sub Classes



# More JMS Features

- Message selectors:
  - SQL-like syntax for accessing *header*:
    subscriber = session.createSubscriber(
      topic, “priority > 6 AND type = ‘alert’ ”);
  - Point to point: selector determines single recipient
  - Pub-sub: acts as filter
- Transactions
  - void onMessage(Message m) {
      try { Message m2=processOrder(m);
        publisher.publish(m2); session.commit();
    } catch(Exception e) { session.rollback(); }

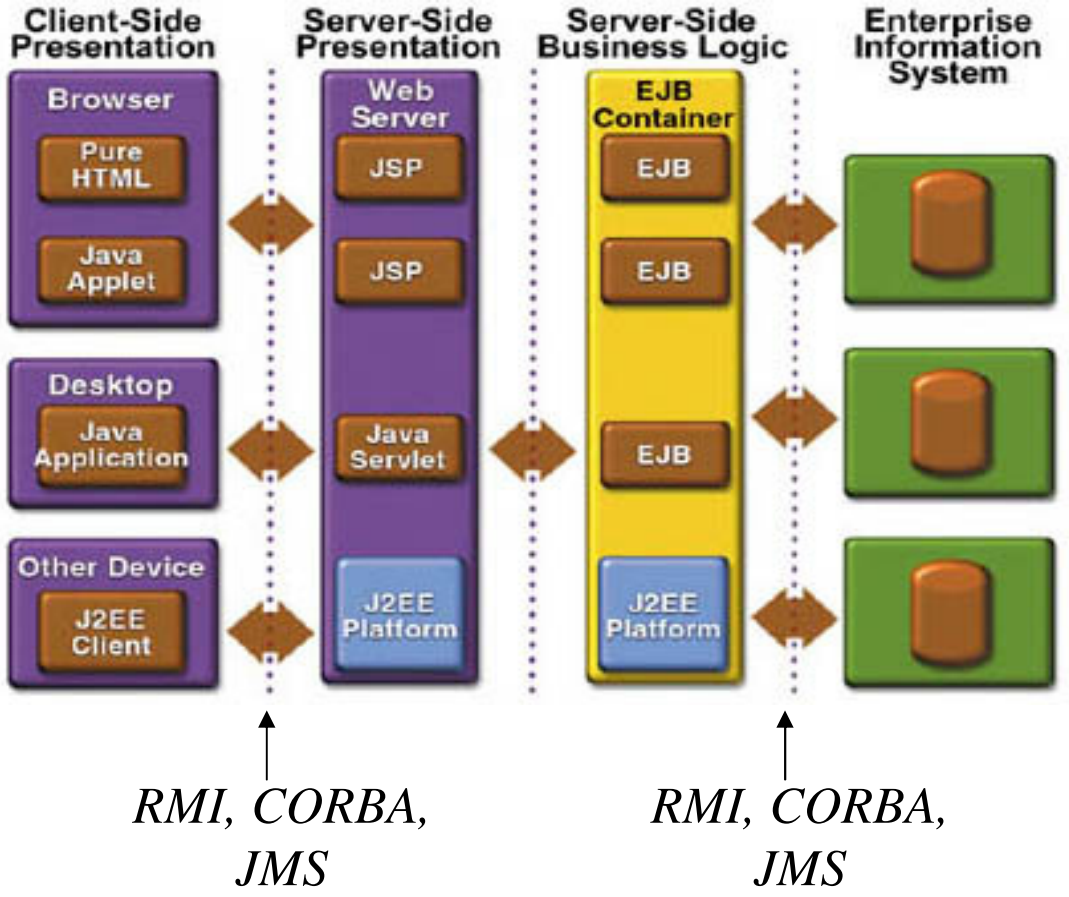
# Request/Reply with Messages

- “80% of inter-application communication is asynchronous, 20% is synchronous (RPC)”
- JMS also provides request/reply
- Request message includes Topic/Queue to reply to
- TopicRequestor/QueueRequestor helper classes
- Idea can easily be extended, e.g. iBus has:
  - Request with timeout
  - Request with multiple replies
- Uses:
  - Fault tolerance (N equivalent replyers).

# JMS and J2EE, EJB

The J2EE Family:

- Enterprise JavaBeans
- JavaServer Pages
- Servlets
- Java Naming and Directory Interface (JNDI)
- Java Transaction API (JTA)
- CORBA
- JDBC data access
- ... and JMS!



# JMS and Enterprise Java Beans

- Application server provides EJB, freeing applications from details of threading, transactions, scalability, fault-tolerance.
- JMS plays similar role to CORBA and RMI: connection from the outside wanting service. Full integration into EJB spec expected June 2000.
- App server transactions replace/augment JMS transactions.
- Messaging implementations from app-server vendors may not be as scalable, flexible as from “pure messaging vendors”.

# JMS Products

- Pure Java
  - SoftWired iBus (http://www.JavaMessaging.com/ibus)
  - Progress SonicMQ (http://www.progress.com/sonicmq/)
  - FioranoMQ (http://www.fiorano.com)
- Java API, C/C++ Implementation
  - Sun's JMQ Product
- JMS API to existing MOM Products
  - IBM MQSeries (http://www. ibm.com/mqseries)
- Application Server Add-On
  - BEA Systems WebLogic, Borland Application Server.

# Distinguishing Features of Products

- Pure Java?
- Pub-sub *and* point to point domains?
- Performance?
- Integration with other products, other languages?
- Quality of service and transport protocols (HTTP)?
- Security?
- XML? (usually simplistic)
- Management tools?
- Pricing, professional services, and support?

# For Further Information

- http://www.java.sun.com/products/jms
- *Developing Java Enterprise Applications*
  by Stephen Asbury and Scott R. Weiner. Wiley & Sons.
  has 80 pages on JMS, also addresses JNDI, EJB
  (OK overview book).
- SoftWired JMS articles:
  http://www.JavaMessaging.com