Experiment No. 4:

Publish–Subscribe Messaging System using JMS (Java Messaging Service) and Apache ActiveMQ.

🧠 Concept Overview

In the Publish–Subscribe model:

- The Publisher sends messages to a Topic.
- One or more Subscribers listen to that topic.
- Every active subscriber receives a copy of each message.

⚙️ Required Tools

- Java 8 or above
- Eclipse IDE (or command line)
- Apache ActiveMQ (download from https://activemq.apache.org/)

After installation, start the broker using:

./bin/activemq start

(Default broker URL → tcp://localhost:61616)

🧩 1. Publisher Code (Publisher.java)

```
// Publisher.java
import javax.jms.*;
import org.apache.activemq.ActiveMQConnectionFactory;

public class Publisher {
   public static void main(String[] args) {
      // ActiveMQ Broker URL
      String brokerURL = "tcp://localhost:61616";

      try {
         // 1. Create a Connection Factory
         ActiveMQConnectionFactory factory = new
ActiveMQConnectionFactory(brokerURL);

         // 2. Create Connection and start it
         Connection connection = factory.createConnection();
         connection.start();

         // 3. Create Session (no transaction, auto acknowledge)
         Session session = connection.createSession(false,
Session.AUTO_ACKNOWLEDGE);

         // 4. Create a Topic named "MyTopic"
         Topic topic = session.createTopic("MyTopic");

         // 5. Create a MessageProducer to send messages to the topic
         MessageProducer producer = session.createProducer(topic);
         producer.setDeliveryMode(DeliveryMode.NON_PERSISTENT);

         // 6. Create a message
```

```
            String text = "Hello Subscribers! Welcome to JMS Publish-Subscribe Example.";
            TextMessage message = session.createTextMessage(text);

            // 7. Send message to Topic
            producer.send(message);
            System.out.println(" 📤 Publisher Sent Message: " + text);

            // 8. Clean up
            session.close();
            connection.close();
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

🧩 2. Subscriber Code (Subscriber.java)

```
// Subscriber.java
import javax.jms.*;
import org.apache.activemq.ActiveMQConnectionFactory;

public class Subscriber {
    public static void main(String[] args) {
        String brokerURL = "tcp://localhost:61616";

        try {
            // 1. Create Connection Factory
            ActiveMQConnectionFactory factory = new
ActiveMQConnectionFactory(brokerURL);

            // 2. Create Connection and start
            Connection connection = factory.createConnection();
            connection.start();

            // 3. Create Session
            Session session = connection.createSession(false,
Session.AUTO_ACKNOWLEDGE);

            // 4. Create Topic (same as publisher)
            Topic topic = session.createTopic("MyTopic");

            // 5. Create a Consumer to receive messages
            MessageConsumer consumer = session.createConsumer(topic);

            // 6. Asynchronous Listener
            consumer.setMessageListener(new MessageListener() {
                public void onMessage(Message msg) {
                    try {
                        if (msg instanceof TextMessage) {
                            TextMessage textMsg = (TextMessage) msg;
```

```
                    System.out.println("📩 Subscriber Received Message: " +
textMsg.getText());
                }
            } catch (Exception e) {
                e.printStackTrace();
            }
        }
    });

    System.out.println("✅ Subscriber is waiting for messages on 'MyTopic'...");
    System.out.println("Press Ctrl+C to exit.");

  } catch (Exception e) {
    e.printStackTrace();
  }
 }
}
```

🧪 Execution Steps

Step 1: Start Apache ActiveMQ

Open terminal → navigate to ActiveMQ folder → run:

./bin/activemq start

Access the console: http://localhost:8161/admin

(Default credentials: admin/admin)

Step 2: Compile Both Files

Make sure activemq-all-x.x.x.jar (found in apache-activemq/lib/) is added to your project or classpath.

Example (command line):

javac -cp "activemq-all-5.16.3.jar;." Publisher.java Subscriber.java

Step 3: Run Subscriber First

java -cp "activemq-all-5.16.3.jar;." Subscriber

You'll see:

✅ Subscriber is waiting for messages on 'MyTopic'...

Step 4: Run Publisher

java -cp "activemq-all-5.16.3.jar;." Publisher

Publisher Output:

📤 Publisher Sent Message: Hello Subscribers! Welcome to JMS Publish-Subscribe Example.

Subscriber Output:

📩 Subscriber Received Message: Hello Subscribers! Welcome to JMS Publish-Subscribe Example.

🏁 Conclusion

✓ You have successfully implemented a Distributed Messaging Application using Java JMS (ActiveMQ) under the Publish–Subscribe Paradigm.

✓ The Publisher broadcasts a message to a Topic, and all active subscribers receive the same message asynchronously.