C0303 - Use messaging resources for asynchronous communications

i) Objective

Java Messaging Service (JMS) architecture and handling message through asynchronous communication.

ii) Introduction

Messaging is a method of communication between software components or applications.
A messaging client can send messages to, and receive messages from, any other client.
Each client connects to a messaging agent that provides facilities for creating, sending, receiving, and reading messages

Asynchronous means that the sender and receiver of the message do not need to interact at the same time.

Messages placed onto the queue are stored until the recipient retrieves them.

Ex : Message queues and mailboxes

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.
The sender and the receiver need to know only what message format and what destination to use.

The Java Message Service is a Java API that allows applications to create, send, receive, and read messages designed by Sun and several partner companies
The JMS API defines a common set of interfaces that allow programs written in the Java programming language to communicate with other messaging implementations.

Modes in JMS :

- Point to Point

In the point-to-point mode, the message producer sends a message to a queue. The message is read just once by a consumer. After the message is read it is deleted
from the queue. There may be more than one potential consumer, however once a message has been read by one consumer it cannot be read by any of the remaining
consumers. A consumer does not need to be registered with the queue at the time the message is sent.

- Publish/subscriber

the message producer sends a message to a topic. One or more consumers register with, or subscribe to, a topic. The message
remains in the topic until all consumers have read the message. If a consumer has not registered with a topic at the time the message is sent then it will be unable to
subsequently read the message.

iii) Presentation

Attachment :

For J2EE configuration, refer the below link :

http://download.oracle.com/javaee/1.3/jms/tutorial/1_3_1-fcs/doc/j2eeapp1.html#1036262

iv) Sample Exercises

Producer

```
import java.util.Hashtable;
import javax.jms.Connection;
import javax.jms.ConnectionFactory;
import javax.jms.Destination;
import javax.jms.JMSException;
import javax.jms.MessageProducer;
import javax.jms.Session;
import javax.jms.TextMessage;
import javax.naming.Context;
import javax.naming.InitialContext;
import javax.naming.NamingException;



public class Producer {


    public static void main(String[] args) {
        final int NUM_MSGS;


        String destName = "jndi/Queue-1";
        System.out.println("Producer.Destination name is " + destName);
```

```
         NUM_MSGS = 2;


        /*
         * Create a JNDI API InitialContext object if none exists
         * yet.
         */
        Context jndiContext = null;

        try {
            jndiContext = new InitialContext();
        } catch (NamingException e) {
            System.out.println("Producer.Could not create JNDI API context: " +
                e.toString());
            System.exit(1);
        }

        /*
         * Look up connection factory and destination.  If either
         * does not exist, exit.  If you look up a
         * TopicConnectionFactory or a QueueConnectionFactory,
         * program behavior is the same.
         */
        ConnectionFactory connectionFactory = null;
        Destination dest = null;

        try {

                Hashtable env = new Hashtable();
            env.put(Context.INITIAL_CONTEXT_FACTORY, "weblogic.jndi.WLInitialContextFactory");
            env.put(Context.PROVIDER_URL, "t3://172.22.45.136:20000");
            InitialContext ctx = new InitialContext(env) ;
            connectionFactory = (ConnectionFactory) ctx.lookup("jndi/ConnectionFactory-1");
            System.out.println(connectionFactory.getClass());
            System. out. println("Producer. connection is loooked up") ;
            dest = (Destination) ctx.lookup(destName);
            System. out. println("Producer. queue is loooked up") ;

        } catch (Exception e) {
            System.out.println("Producer.JNDI API lookup failed: " + e.toString());
            e.printStackTrace();
            System.exit(1);
        }

        /*
         * Create connection.
         * Create session from connection; false means session is
         * not transacted.
         * Create producer and text message.
         * Send messages, varying text slightly.
         * Send end-of-messages message.
         * Finally, close connection.
         */
        Connection connection = null;
        MessageProducer producer = null;

        try {
            connection =  connectionFactory.createConnection();

            Session session = connection.createSession(false, Session.AUTO_ACKNOWLEDGE);
            producer = session.createProducer(dest);

            TextMessage message = session.createTextMessage();

            for (int i = 0; i < NUM_MSGS; i++) {
```

```
                message.setText("This is message  : " + (i + 1));
                System.out.println("Producer.Sending message: " + message.getText());
                producer.send(message);
            }

            /*
             * Send a non-text control message indicating end of
             * messages.
             */
            producer.send(session.createMessage());
            System.out.println("Producer.Sending completed");
        } catch (JMSException e) {
            System.out.println("Producer.Exception occurred: " + e.toString());
        } finally {
            if (connection != null) {
                try                   connection.close();
                } catch (JMSException e) {
                }
            }
        }
    }


}
```

Consumer

```
import java.io.IOException;
import java.io.InputStreamReader;
import java.util.Hashtable;

import javax.jms.Connection;
import javax.jms.ConnectionFactory;
import javax.jms.Destination;
import javax.jms.JMSException;
import javax.jms.MessageConsumer;
import javax.jms.Session;
import javax.jms.TextMessage;
import javax.naming.Context;
import javax.naming.InitialContext;
import javax.naming.NamingException;


public class Consumer {

        /**
     * Main method.
     *
     * @param args     the destination name and type used by the
     *                 example
     */
    public static void main(String[] args) {
        String destName = null;
        Context jndiContext = null;
        ConnectionFactory connectionFactory = null;
        Connection connection = null;
        Session session = null;
        Destination dest = null;
        MessageConsumer consumer = null;
        TextListener listener = null;
        TextMessage message = null;
        InputStreamReader inputStreamReader = null;
        char answer = '\0';
```

```
            destName = "jndi/Queue-1";
            System.out.println("Consumer.Destination name is " + destName);

            /*
             * Create a JNDI API InitialContext object if none exists
             * yet.
             */
            try {
                jndiContext = new InitialContext();
            } catch (NamingException e) {
                System.out.println("Consumer.Could not create JNDI API context: " +
                    e.toString());
                System.exit(1);
            }

            /*
             * Look up connection factory and destination.  If either
             * does not exist, exit.  If you look up a
             * TopicConnectionFactory or a QueueConnectionFactory,
             * program behavior is the same.
             */
            try {
                    Hashtable env = new Hashtable();
                env.put(Context.INITIAL_CONTEXT_FACTORY, "weblogic.jndi.WLInitialContextFactory");
                env.put(Context.PROVIDER_URL, "t3://172.22.45.136:20000");
                InitialContext ctx = new InitialContext(env) ;
                connectionFactory = (ConnectionFactory) ctx.lookup("jndi/ConnectionFactory-1");
                System. out. println("Consumer. connection is loooked up") ;
                dest = (Destination) ctx.lookup(destName);
                System. out. println("Consumer. queue is loooked up") ;
            } catch (Exception e) {
                System.out.println("Consumer.JNDI API lookup failed: " + e.toString());
                System.exit(1);
            }

            /*
             * Create connection.
             * Create session from connection; false means session is
             * not transacted.
             * Create consumer.
             * Register message listener (TextListener).
             * Receive text messages from destination.
             * When all messages have been received, type Q to quit.
             * Close connection.
             */
            try {
                connection = connectionFactory.createConnection();
                session = connection.createSession(false, Session.AUTO_ACKNOWLEDGE);
                consumer = session.createConsumer(dest);
                listener = new TextListener();
                consumer.setMessageListener(listener);
                connection.start();
                System.out.println("Consumer.To end program, type Q or q, " +
                    "then <return>");
                inputStreamReader = new InputStreamReader(System.in);

                while (!((answer == 'q') || (answer == 'Q'))) {
                    try {
                        answer = (char) inputStreamReader.read();
                        System.out.println("Consumer.answer: "+answer);
                    } catch (IOException e) {
                        System.out.println("Consumer.I/Oq exception: " + e.toString());
                    }
                }
            } catch (JMSException e) {
                System.out.println("Consumer.Exception occurred: " + e.toString());
```

```
        } finally {
            if (connection != null) {
                try {                connection.close();
                } catch (JMSException e) {
                }
            }
        }
    }

}
```

TextListener

```
import javax.jms.JMSException;
import javax.jms.Message;
import javax.jms.MessageListener;
import javax.jms.TextMessage;


public class TextListener implements MessageListener {
        /**
     * Casts the message to a TextMessage and displays its text.
     *
     * @param message     the incoming message
     */
    public void onMessage(Message message) {
        TextMessage msg = null;

        try {
            if (message instanceof TextMessage) {
                msg = (TextMessage) message;
                System.out.println("Reading message: " + msg.getText());
            } else {
                System.out.println("Message is not a TextMessage");
            }
        } catch (JMSException e) {
            System.out.println("JMSException in onMessage(): " + e.toString());
        } catch (Throwable t) {
            System.out.println("Exception in onMessage():" + t.getMessage());
        }
    }

}
```

Name	Size	Creator	Creation Date	Comment	
JMS.ppt	190 kB	LALITHA R.	Sep 02, 2011 15:31		View
J2EE.zip	2 kB	LALITHA R.	Sep 02, 2011 15:26		
ConfigurationInWeblogic.ppt	42 kB	LALITHA R.	Sep 02, 2011 15:16		View
source.zip	3 kB	LALITHA R.	Sep 02, 2011 15:17		

V) Assignment

Do the exercise for Sending and Receiving message with newly created queue