# Part 2

## Representative Questions – Please write your answers with an example for each questions.

* How do you design an application with JMS messaging?

Need to follow below steps to design an application.

Step 1) Create a JMS Queue in weblgic JMS. webspear MQ, ActiveMQ etc.

Step 2) Have a JNDI/Broker Name for JMQ Queue.

Step 3) Write a sender application to send message to the queue.

QueueConnectionFactory

QueueConnection

Queue

QueueSession

QueueSender/QueueReceiver

Step 4) Share the JNDI/Broker information to queue receiver to receive messages from the question.

* How do you handle exception in JMS consumers and how to you recover?

In order to handle any exception in consumer,We have to set Session.AUTO\_ACKNOWLEDGE false. This means consumer must acknowledge (msg.acknowledge()) after receiving the message if any issue in receiver side and not calling this method, the message will be remaining in the queue.

* How do you implement LRU or MRU cache?

LRU – least recently used. – I will use double linked list and HashMap for LRU implementation. Most LRU Caching algorithms consist of two parts: a dictionary and a list. The dictionary guarantees quick access to your data, and the list, ordered by age of the objects, controls the lifespan of objects and determines which objects are to be removed first. A simple LRU Cache implementation uses a doubly linked list; adding new items to the head, removing items from the tail, and moving any existing items to the head when referenced.

MRU – Most recently used.

Basically, is a HashMap (extends HashMap<Object, Object> if you will)

Each value in the Map points to an object in a sorted list, based on which is most used.

Objects recently used are added to the head of the list - O(1)

Purging least-recently used means truncating the end of the list - O(1)

Still gives you Map lookups, but keeps recently-used items first

* How would you implement Executor Service?

* Describe singleton design pattern – how would you implement?
* Describe properties of Java String.