**1) How do you design an application with JMS messaging?**

JMS (Java Message Service) is mainly used to send and receive message from one application to another application. It’s an API used in designing an application which provides loosely coupled, reliable and asynchronous communication. To design an application with JMS we will use following steps:

1)Point-to-point Messaging domain.

1. We should create connection factory and destination source,
2. We should create sender and receiver application

Here one message is deliver to only one client and we use queue as a middleware which is responsible to hold and deliver messages.

2)Publisher/subscriber Messaging domain

1. We should create connection factory and destination source,
2. We should create sender and receiver application

Here one message is delivered to all subscribers and we use topic as a middleware.

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

This can be achieved my using session acknowledgement. For this we should modify our producer code to use Session.AUTO\_ACKNOWLEDGE. While creating Queue session, we should AUTO\_ACKNOWLEDGE as false, which means consumer should acknowledge. When the consumer sends acknowledgement of message, then the message will be deleted from the queue, otherwise it will remain in the queue. On the consumer side, we should do the same thing, we should create a queue session with AUTO\_ACKNOWLEDGE as false.

After working on your message, we can send acknowledge to delete the message from the queue or the message will remain in the queue.

**3) How do you implement LRU or MRU cache?**

LRU can be implemented in following 2 ways

**Queue:** It can be implemented using a doubly linked list. The maximum size of the queue will be equal to the total number of frames available (cache size). The most recently used pages will be near front end and least recently pages will be near rear end.

**Hashing:** A hash with page number as key and address of corresponding queue node as a value, when a page is referenced, the required page may be in the memory, if it is in the memory, we need to detach the node of the list and bring it to the front of the queue. If the required page is not in the memory, we bring that in memory. In simple words, we add a new node to the front of the queue and update the corresponding node address in the hash. If the queue is full, i.e. all the frames are full, we remove a node from the rear of queue, and add the new node to the front of queue.

**MRU:** To implement a cache, we should derive a subclass from this template class. As it is a implementer, we should implement two methods:

HandleNonExistingKeyFetch to handle cache misses. In this method, you access the real source of data behind the cache and return the value.

HandleItemRelease- called when an item is removed from the cache.

The cache class is a template of two types, a key and value. The value type is the type of the resource, and key type is the type of the resource address.

**4) How would you implement Executor Service?**

Since executor service is an interface, you need to its implement it to make any use of it. The executiveservice has the following implementation in the java.uril.concurent package.

ThreadPoolExecutor

ScheduledThreadPool Executor.

Following are the steps to be followed

Creating an ExecutiveService.

ExecutorService Usage: there are few different ways to delegate tasks for execution to an ExecutorService: Execute(runnable), Submit(runnable). Submit(callable), InvokeAny(…), InvokeAll(..)

ExecutorService Shutdown

**5) Describe singleton design pattern – how would you implement?**

Singleton example is one of the most straightforward outline designs in Java. This sort of configuration example goes under creational design as this example gives one of the most ideal approaches to make a question.

This example includes a solitary class which is capable to make a question while ensuring that lone single protest gets made. This class gives an approach to get to its lone question which can be gotten to straightforwardly without need to instantiate the protest of the class.

Sample Implementation code in java:

class Singleton {

public:

static Singleton\* Instance();

protected:

Singleton();

private:

static Singleton\* \_instance;

}

// Implementation

Singleton\* Singleton::\_instance = 0;

Singleton\* Singleton::Instance() {

if (\_instance == 0) {

\_instance = new Singleton;

}

return \_instance;

}

**6) Describe properties of Java String.**

Java String is a final class and is a immutable.java Properties is a subclass of Hashtable. It is used to maintain lists of values in which the key is a String and the value is also a String. The Properties class is used by many other Java classes. For example, it is the type of object returned by System.getProperties() when obtaining environmental values.

**References:**

1.<https://www.ibm.com/support/knowledgecenter/en/SSAW57_8.5.5/com.ibm.websphere.nd.doc/ae/tmj_desap.html>

2.<https://docs.oracle.com/en/>