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

If we take a small Banking application, where user wants to get “Detailed statement” with in an amount of some period. Then we can get the current month followed by number of previous months and trigger the data separately from bank account and put it in Queue. When we receive the data we can merge them and make it align and send back to user.

*Usage:* User gets quick updates since we make asynchronous calls to Bank.

*Reference*: We used RabbitMQ for one of the applications , I tried to map how JMS can fit in banking application

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

🡪 Using normal Java Exception Handling and whenever we get “JMSException” ,

🡪 catch normal Exception and try if it’s instanceOf JMSException and handle

and enclose this part of code in onResponse() or onMessage().

Ex : Response onResponse(…) {

Try {

* Do something

} catch (JMSException je) {

logger or retry if timeout occurred

}

}

void onError(Exception ex) {

logger. Warn or logger. Error – to intimate the client

}

1. **How do you implement LRU or MRU cache?**

I would implement using Doubly Linked List and Hashtables. Cause I am convinced with whatever he implemented here and I never used these much directly.

*Reference:* <https://www.codeproject.com/Articles/7047/Most-Recently-Used-MRU-hashtable-LRU-> expiration

1. **How would you implement Executor Service?**

Initialize the service with a Factory Object and we can create 3 types of Executor Services.

1. ThreadPoolExecutor
2. ScheduledThreadPoolExecuter()

Since these 2 can dynamically increase and decrease the executor creations based on the tasks.

However, we can use State callback mechanism, using callbacks or Promises and get the asynchronous framework rather going to thread concepts and getting dead lock situation.

1. **Describe singleton design pattern – how would you implement?**

A class can Initialize only once (or) Only one instance a class can hold with.

In general, Create instance of a class once and use it several times.

*Examples*: I have used singleton DP to

1. create a connection object, whenever we need connection, just call connectionfactory.getConnection()
2. Get logger object Loggermgr.getLogger()
3. Or any pojo classes where we can say StorageResponse.getInstance()

*Implementation:*

1. Make contructor private
2. *Private static Class <instanceOfCls>;*
3. *Synchronize method for getInstance() or getConnection() where you will create object if there is not one, otherwise return the existing one*.
4. **Describe properties of Java String.**
5. 1. String is immutable, cannot change contents once it’s created.
6. 2. Its final class, we cannot extend String
7. 3. Can create with literal and new operator.
8. 4. Any operation on String returns us new Object by not affecting the actual String.
9. 5. Can compare the address and contents of strings.
10. 5. Length() , toUppercase() , toLowercase() , concate(), charAt(), toCharArray(), equals()….etc