**1).How to design an application using JMS messaging?**

JMS (Java Message Service) is an API that provides the facility to create, send and read messages. It provides loosely coupled, reliable and asynchronous communication.

JMS is also known as a messaging service.

Messaging is a technique to communicate applications or software components.

JMS is mainly used to send and receive message from one application to another.

Generally, user sends message to application. But, if we want to send message from one application to another, we need to use JMS API.

Consider a scenario, one application A is running in INDIA and another application B is running in USA. To send message from A application to B, we need to use JMS.

1) Asynchronous: To receive the message, client is not required to send request. Message will arrive automatically to the client.

2) Reliable: It provides assurance that message is delivered.

There are two types of messaging domains in JMS.

Point-to-Point Messaging Domain

Publisher/Subscriber Messaging Domain

1) Point-to-Point (PTP) Messaging Domain

In PTP model, one message is delivered to one receiver only. Here, Queue is used as a message oriented middleware (MOM).

The Queue is responsible to hold the message until receiver is ready.

In PTP model, there is no timing dependency between sender and receiver.

2) Publisher/Subscriber (Pub/Sub) Messaging Domain

In Pub/Sub model, one message is delivered to all the subscribers. It is like broadcasting. Here, Topic is used as a message oriented middleware that is responsible to hold and deliver messages.

In PTP model, there is timing dependency between publisher and subscriber.

Create Sender Receiver Application:

-----------------------------------

import java.io.BufferedReader;

import java.io.InputStreamReader;

import javax.naming.\*;

import javax.jms.\*;

public class MySender {

public static void main(String[] args) {

try

{ //Create and start connection

InitialContext ctx=new InitialContext();

QueueConnectionFactory f=(QueueConnectionFactory)ctx.lookup("myQueueConnectionFactory");

QueueConnection con=f.createQueueConnection();

con.start();

//2) create queue session

QueueSession ses=con.createQueueSession(false, Session.AUTO\_ACKNOWLEDGE);

//3) get the Queue object

Queue t=(Queue)ctx.lookup("myQueue");

//4)create QueueSender object

QueueSender sender=ses.createSender(t);

//5) create TextMessage object

TextMessage msg=ses.createTextMessage();

//6) write message

BufferedReader b=new BufferedReader(new InputStreamReader(System.in));

while(true)

{

System.out.println("Enter Msg, end to terminate:");

String s=b.readLine();

if (s.equals("end"))

break;

msg.setText(s);

//7) send message

sender.send(msg);

System.out.println("Message successfully sent.");

}

//8) connection close

con.close();

}catch(Exception e){System.out.println(e);}

}

}

import javax.jms.\*;

import javax.naming.InitialContext;

public class MyReceiver {

public static void main(String[] args) {

try{

//1) Create and start connection

InitialContext ctx=new InitialContext();

QueueConnectionFactory f=(QueueConnectionFactory)ctx.lookup("myQueueConnectionFactory");

QueueConnection con=f.createQueueConnection();

con.start();

//2) create Queue session

QueueSession ses=con.createQueueSession(false, Session.AUTO\_ACKNOWLEDGE);

//3) get the Queue object

Queue t=(Queue)ctx.lookup("myQueue");

//4)create QueueReceiver

QueueReceiver receiver=ses.createReceiver(t);

//5) create listener object

MyListener listener=new MyListener();

//6) register the listener object with receiver

receiver.setMessageListener(listener);

System.out.println("Receiver1 is ready, waiting for messages...");

System.out.println("press Ctrl+c to shutdown...");

while(true){

Thread.sleep(1000);

}

}catch(Exception e){System.out.println(e);}

}

mport javax.jms.\*;

public class MyListener implements MessageListener {

public void onMessage(Message m) {

try{

TextMessage msg=(TextMessage)m;

System.out.println("following message is received:"+msg.getText());

}catch(JMSException e){System.out.println(e);}

}

}

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

-------------------------------------------------------------------

Create the session like this:

Session session = connection.createSession(false,

Session.CLIENT\_ACKNOWLEDGE);

when you try to deliver the message to your third party app:

If it's working you should acknoledge the message.

If it is down you should'nt acknwoledge it, this way the JMS provider will be able to rediliver it,and the message will not be lost. message.acknowledge();

public class SampleJMSConsumer implements MessageListener {

@Override

public void onMessage(Message message) {

ObjectMessage objectMessage = (ObjectMessage) message;

Object object;

try {

object = objectMessage.getObject();

if (object instanceof String) {

System.out.println("Message received - " + object.toString());

throw new JMSException("JMS exception");

}

message.acknowledge();

} catch (JMSException e) {

session.recover();

}

}

}

**3).How do you Implement LRU or MRU cache?**

------------------------------------------

LRU Cache:

-----------

import java.util.HashMap;

public class LRUCache<K, V> {

private transient Entry<K, V> header = new Entry<K, V>(null, null, null, null);

public HashMap<K,IndexNode<Entry<K,V>>> indexMap = new HashMap<K,IndexNode<Entry<K,V>>>();

private final int CACHE\_LIMIT = 3;

private int size;

public LRUCache() {

header.next = header.previous = header;

this.size = 0;

}

public void put(K key,V value){

Entry<K,V> newEntry = new Entry<K,V>(key,value,null,null);

addBefore(newEntry, header);

}

private void addBefore(Entry<K,V> newEntry,Entry<K,V> entry){

if((size+1)<(CACHE\_LIMIT+1)){

newEntry.next=entry;

newEntry.previous=entry.previous;

IndexNode<Entry<K,V>> indexNode = new IndexNode<Entry<K,V>>(newEntry);

indexMap.put(newEntry.key, indexNode);

newEntry.previous.next=newEntry;

newEntry.next.previous=newEntry;

size++;

}else{

Entry<K,V> entryRemoved = remove(header.next);

indexMap.remove(entryRemoved.key);

addBefore(newEntry, entry);

}

}

public void get(K key){

if(indexMap.containsKey(key)){

Entry<K,V> newEntry = remove(indexMap.get(key).pointer);

addBefore(newEntry,header);

}else{

System.out.println("No such element was cached. Go and get it from Disk");

}

}

private Entry<K,V> remove(Entry<K,V> entry){

entry.previous.next=entry.next;

entry.next.previous = entry.previous;

size--;

return entry;

}

public void display(){

for(Entry<K,V> curr=header.next;curr!=header;curr=curr.next){

System.out.println("key : "+curr.key+" value : " + curr.value);

}

}

private static class IndexNode<Entry>{

private Entry pointer;

public IndexNode(Entry pointer){

this.pointer = pointer;

}

}

private static class Entry<K, V> {

K key;

V value;

Entry<K, V> previous;

Entry<K, V> next;

Entry(K key, V value, Entry<K, V> next, Entry<K, V> previous) {

this.key = key;

this.value = value;

this.next = next;

this.previous = previous;

}

}

public static void main(String[] args) {

LRUCache<String, Integer> cache = new LRUCache<String, Integer>();

cache.put("abc", 1);

//cache.display();

cache.put("def", 2);

cache.put("ghi", 3);

cache.put("xyz", 4);

cache.put("xab", 5);

cache.put("xbc", 6);

cache.get("xyz");

cache.display();

System.out.println(cache.indexMap);

}

}

MRU Cache:

----------

public interface ICache {

public void put(Object key, Object val);

public Object get(Object key);

public void invalidate(Object key);

public void printKeyOrder();

}

mport java.util.HashMap;

import java.util.LinkedList;

public abstract class AbstractCache implements ICache {

int cacheSize;

HashMap map;

LinkedList list;

public AbstractCache(int cacheSize) {

this.cacheSize = cacheSize;

map = new HashMap(cacheSize);

list = new LinkedList();

}

public abstract void put(Object key, Object val);

public abstract Object get(Object key);

public void invalidate(Object key) {

list.remove(key);

map.remove(key);

}

public void printKeyOrder() {

System.out.println("KeyOrder() " + list + " Cache content: " + map);

}

public void prune()// removes the tail

{

Object key = list.removeLast();

map.remove(key);

}

}

public class CacheFactory {

public static ICache newInstance(int cacheSize, EvictionStrategy strategy) {

if (EvictionStrategy.LRU == strategy)

{

return new LRUCache(cacheSize);

}

else if (EvictionStrategy.MRU == strategy) {

return new MRUCache(cacheSize);

}

else {

throw new RuntimeException("Invalid EvictionStrategy");

}

}

}

public enum EvictionStrategy {

LRU, MRU

}

public class LRUCache extends AbstractCache {

public LRUCache(int cacheSize) {

super(cacheSize);

}

public void put(Object key, Object val) {

// check if pruning is needed

if (list.size() == this.cacheSize) {

this.prune();

}

list.addFirst(key);

map.put(key, val);

}

public Object get(Object key) {

boolean res = list.remove(key);

if (res) {

list.addFirst(key);

return map.get(key);

}

return null;

}

}

public class MRUCache extends AbstractCache {

public MRUCache(int size) {

super(size);

}

@Override

public void put(Object key, Object val) {

// check if pruning is needed

if (list.size() == this.cacheSize) {

this.prune();

}

list.addLast(key);

map.put(key, val);

}

@Override

public Object get(Object key) {

boolean res = list.remove(key);

if (res) {

list.addLast(key);

return map.get(key);

}

return null;

}

}

**4).How do you implement Executor service in java?**

-------------------------------------------------

ExecutorService threadPoolService = Executors.newFixedThreadPool(10);

for ( Runnable task : myTasks )

{

threadPoolService.submit(task);

}

public class TaskRepeatingThreadPoolExecutor extends ThreadPoolExecutor {

public TaskRepeatingThreadPoolExecutor(int corePoolSize, int maximumPoolSize, long keepAliveTime, TimeUnit unit, BlockingQueue<Runnable> workQueue) {

super(corePoolSize, maximumPoolSize, keepAliveTime, unit, workQueue);

}

@Override

protected void afterExecute(Runnable r, Throwable t) {

super.afterExecute(r, t);

this.submit(r);

}

}

**5).Describe singleton design pattern?How do you Implement?**

----------------------------------------------------------

In Java the Singleton pattern will ensure that there is only one instance of a class is created in the Java Virtual Machine. It is used to provide global point of access to the object. In terms of practical use Singleton patterns are used in logging, caches, thread pools, configuration settings, device driver objects.

To implement Singleton pattern, we have different approaches but all of them have following common concepts.

Private constructor to restrict instantiation of the class from other classes.

Private static variable of the same class that is the only instance of the class.

Public static method that returns the instance of the class, this is the global access point for outer world to get the instance of the singleton class.

public class ThreadSafeSingleton {

private static ThreadSafeSingleton instance;

private ThreadSafeSingleton(){}

public static synchronized ThreadSafeSingleton getInstance(){

if(instance == null){

instance = new ThreadSafeSingleton();

}

return instance;

}

}

public static ThreadSafeSingleton getInstanceUsingDoubleLocking(){

if(instance == null){

synchronized (ThreadSafeSingleton.class) {

if(instance == null){

instance = new ThreadSafeSingleton();

}

}

}

return instance;

}

public enum EnumSingleton {

INSTANCE;

public static void doSomething(){

//do something

}

}

**Describe properties of Java String?**

------------------------------------

String Class

1. The String class is immutable.

2. The contents of the String object can not be changed.

3. String class is final class. That implies it can not have sub classes.

4. String objects can be concatenated by using + and += operators

Following are the examples to create String class objects

String city1 = “Bangalore”;

String city2 = “Mumbai”;

String city3 = “New Delhi is the “ + “Capital of India”;

String name = “Fedrick”;

String str4 = “My name is” + name;

String str5 = new String(“My name is Jones”);

Methods of String class

Accessor Methods

1. length() – To find the number of characters of a string

2. charAt(index) – Returns the character a the given index

3. split(string,delimiter) – Splits the at every delimiter and returns a String array

4. substring(start[,endindex]) – Returns all the characters from start upto but not including endindex

Modifier Methods

1. concat(string) : Returns a new string after concatenating str to the original string. The original string remain unchanged.

2. replace(charwith, charreplacement) : Replaces charwith with charreplacement and returns a new string.

Boolean test methods

1. endsWith(strend) : Returns true, if the string ends with strend

2. equals(str) : Returns true, if the string and str are equal

3. equalsIgnoreCase(str) : Returns true, if the string and the str are equal irrespective of case sensitivity

4. startsWith(strbeg) : Returns true, if the string starts with strbeg

Inter test methods

1. compareTo(str) : Returns 0 if str equals with the object, -1 if object is before parameter in sort order and +1 if otherwise

2. indexOf(substr) : Returns the position of the first occurrence of a substring in the given string

3. length() : Returns the number of characters in the string