Traditional collections :

===========================

1)like list set and map are not thread safe

2)even we have vector,hashtable but performance wise not good..(because total object locking..)

Collections.synchroonizedList()

Collections.synchroonizedSet()

Collections.synchroonizedMap()

3)while one thread is iterating the object other thread not allowed to modifying the object other wise we are going to get CMException.

Concurrent collections:

======================

1)more thread safe

2)high performance beacuase of different locking mechanisam

3)concurrent modification we can do we are not going to get CMException....

three important councrrent collections:

=======================================

1)ConcurrentHashMap

2)CopyOnWriteArrayList

3)CopyOnWriteArraySet

ConurrentMap:

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

\*\* All concurrent collections classes present in java.util.concurrent

Map is the parent Interface of ConcurrentMap(I)--->ConcurrentHashMap(class) child to ConcurrentMap

in ConcurrentMap has 3 special methods they are like

\* putIfAbsent()

\* remove()

\* replace()

1)Object putIfAbsent(Object key,Object value),same like put method but if key is already there it wont add new value.

2)boolean remove(Object key,Object value),we need pass key and value to remove...

3)boolean replace(object key,object oldvalue,object newvalue)

ConurrentHashMap:

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

hashmap -->not thread safe

hashtable-->thread safe

concurrenthashmap-->threadsafe

default intial size of hasmap,concurrenthashmap is 16..

0 to 15 buckets we are going to have ...

when we are working with hashtable the lock for complete object no other thread can write or update even read also

bucket level locking,segment level locking:

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

\*In concurrenthashmap multiple thread can read at a time and for write and update bucket lock is required

so in concurrenthashmap at a time 16 threads can execute....(16 concurrency level)

CopyOnWriteArrayList:

======================

\*it is realted to arraylist...

\*it is concurrent version arraylist(thread safe version of arraylist)

\* it is a child class List(I)...like arraylist

\*\* for every write or update operation on cloned object,then there is no effect on original object.

@certain point two objects will be sync...

Methods of CopyOnWriteArrayList:

================================

1)boolean addIfAbsent(Object o);

2)int addAllIfAbsent(Object o);

CopyOnWriteArraySet:

===================

\*It is a thread safe version of set.

\* child class Set interface.

\*Insertion order is Preserved

\*Duplicate are not allowed.

\*Multiple thread can able read operation simultaneously but for every update operation a separate

cloned copy will created.