Chapter- 11

Collection Framework

অ্যারে এর ক্ষেত্রে যদি আমারা ১০ length এর একটি অ্যারে তৈরি করি তবে যদি আরেতে একটি ডাটা রাখি তবেও অ্যারে তার সকল length এর ডাটা প্রিন্ট করবে, তবে null ।

Collection

Set List Queue

HashSet LinkedHashSet SortedSet ArrayList Vector LinkedList PriorityQueue

TreeSet

Methods of Collection:-

* Int size()
* Boolean isEmpty()
* Boolean add(E e) return false is add duplicate
* Boolean remove(Object o) return false is object is no exists
* Boolean contains(Object o)
* Object[] toArray()
* Boolean containAll(Collection<?> c)
* Boolean addAll(Collection<? Extends E> c)
* Boolean removeAll(Collection<?> c)

List:-

Methods of List:-

* Boolean add(E e)
* Void add(int index, E e)
* E remove(int index)
* E get(int index)
* E set(int index,E elements)
* Int indexOf(Object o)
* Int lastIndexOf(Object o)

Stack:-

* E push(E item)
* E pop()
* E pcek()
* Boolean empty()

Iterable & Iterator

Java.lang.Iterable এটি Collection Framework এর মূল Interface গুলোর অন্যতম একটি ।

এটি Collection interface কে externds করে।

কোন Class এই interface কে implements করলে সেই class এ for each loop ব্যবহার করা যায়।

List<Integer> nums=new ArrayList<> ();

For(Integer num:nums)

{

}

এর একটি Method আছে যা iterator interface এর reference return করে।

Public Iterator<T> iterator();

Iterator :-

* Boolean hasNext()

আরও উপাদান কনটেইনার ক্লাস এ রয়েসে কিনা তা নিদেশ করে।

* E next()

পরের উপাদান নিদেশ করে।

Set

It don’t store Duplicate value.

HashSet:-

এটি নিদিষ্ট অডার এ ডাটা ধারন করে না।

LinkedHashSet:-

এতে যে অর্ডার এ ভালু ধারণ করা হয় সেই অডার এই থাকে।

TreeSet:-

এটি SortedSet কে ইমপ্লেমেন্ট করে, তাই এতে Sorted থাকে।

কিন্ত এতে যে অবজেক্ট রাখা হবে তাকে অবশ্যই java.lang.Comparable কে implements করতে হবে।

E get(int index) নেই।

Queue

* E elements()

return first elements but not delete.

if failed thow NoSuchElementException

* E peek()

return first value with out delete.

* boolean add(E e)

if failed thow IllegalStateException.

* boolean offer(E e)

store value.

* E poll()

return & delete.

implements of it,

1. LinkedList
2. PriorityQueue

when we store a object in PriorityQueue, then the object class have to implements Comparable interface.

Stategy:-

* Array to ArrayList:-

List<Integer> list=Arrays.asList(0,1,2,3,4);

* List to Array:-

List<Integer> list=new ArrayList<> ();

Integer[] arr=list.toArray(new Integer[list.size()]);

* List to Set:-

List<Integer> list=new ArrayList<>();

Set<Integer> set=new HashSet<>(list);

* Set to List:-

List<Integer> list=new HashSet<>(set);

Map

it called Key-Value pair / Associative Array.

Map

HashSet HashMap LinkedHashMap SortedMap

TreeMap

methods:-

* V put(K key,V value)
* V get(K key)
* V remove(K key)
* boolean containsKey(K key)

check key have or not

* boolean containsValue(V value)
* int size()
* Set<K> keySet()

return set of key.

* Collection<V> values()

return collection of values.

Implementation of Map:-

* HashSet<K, V>
* TreeMap<K, V>
* LinkedHashMap<K,V>
* Hashtable<K,V>

HashMap:-

class Person implements comparable<Person >

{

private String name;

private Integer age;

public Person(String name,Integer age)

{

this.name=name;

this.age=age;

}

public String getName()

{

return name;

}

public Integer getAge()

{

return age;

}

@Override

public String toString()

{

return name+” “+age;

}

@Override

public int compareTo(Person o)

{

return o.age.compareTo(this.age);

}

}

public class HashMapExample

{

public static void main(String ar[])

{

Map<Person,String> phoneBook=new HashMap<>();

Person pollob=new Person(“Pollob”,19);

Person ray=new Person(”Roy”,20);

phoneBook.put(pollob,”123456”);

phoneBook.put(roy,”12345”);

for(Map.Entry<Person,String> entry:phoneBook.entrySet())

{

System.out.println(entry.getKey().toString()+” “+entry.getValue());

}

}

}

Map এ Key দইটি Value pair রয়েছে।

যদি কোন ক্ষেত্রে দুইটি কী একই হয় তবে, তাদের দইটিকে সমান বলতে হলে, এই ক্লাস এ আরও দটি মেথড equals() & hashCode() override করতে হবে।

@Override

Public Boolean equals(Object o)

{

If(this==0)

Return true;

If(o==null || getClass()!=o.getClass())

Return false;

Person person=(Person) o;

Return Objects.equalse(name,person,name);

}

@Override

Public int hashcode()

{

Return Objects.hash(name);

}

Treemap:-

Treemap এর key সবসময় sorted থাকে। TreeMap SortedMap interface কে implements করে।

TreeMap এ Key objects গুলোর class কে অবশ্যয় Comparable interface টি implements টি implement করতে হবে।

HashTable:-

এটি এবং TreeMap এর মধ্য বিশেষ কোন পার্থক্য নেই। সধ মেথড synchronousnized .

LinkedHashMap:-

এটি HashMap এর subclass.

এটি key value যেই order এ রাখা হয় সেই order এ ধরে রাখে।

Collection Sorting/ Ordering:-

Object list sort করার জন্য java.util.Collections class এর sort(List<T>) static method ব্যবহার করা হয়।

এই মেথড ব্যবহার করে দুই ধরণের List sort করা যায়।

1. ArrayList
2. LinkedList

Ex:-

List<Integer> listA=new ArrayList<>();

listA.add(5);

listA.add(3);

Collections.sort(listA);

এবং sorting হয়,

Public interface Comparable<T>

{

Public int compareTo(T o);

}

Integer,String class গুলো এই interface কে inherite করেছে।

Ex:-

List<Person> person=new ArrayList<>();

person.add(new Person(“James “,”Gosling”);

for(Person p:person)

{

System.out.println(p.getFirstName()+” “+p.getLastName());

}

# যখন Comparable interface টি implement করা সম্ভব নয় তখন আরেকটি বিশেষ interface ব্যবহার করা হয়।

Public interface Comparator<T>

{

Int compare(T obj1,T obj2);

}

Ex:-

@Override

Public int compare(Person o1,Person o2)

{

Return o1.getLastName().compareTo(o2.getLastName());

}

Lambda Expression,,

Collections.sort(person,(o1,o2)-> o1.getLastName().compareTo(O2.getLastName());

Java তে primitive type এবং reference type Array sort করার জন্য java.util.Arrays class রয়েছে।

Methods,

* Static void sort(byte array[])
* Static void sort(byte array[],int fromindex,int toIndex);

# Reference type/ Object Array sort করতে হলে object গুলোকে comparable interface implements করতে হবে,

* Void sort(Object[] a)
* Void sort(Object[] a, int fromIndex,int toIndex)
* Void sort(T[] a, comparator(? Super T> c)
* Void sort(T[] a, int fromIndex,int toIndex,Comparator9? Super T> c)

Ex:-

Person[] persons=new Person[]{

New Person(“Person 1”);

New Person(“Person 2”);

};

Array.sort(persons,new comparator<Person>()

{

@Override

Public int compare(Person o1,Person o2)

{

Return o1.getName().compareTo(o2.getName());

});

Search:-

Array তে search করতে Time Complexity বেশী।

তবে Binary search এর টাইম complexity কম।

Methods,

* Int binarySearch(List<? Extends Comparable<?super T>>list,T key)
* Int binarySeacrh(List<? Extends T>list,T key,Comparator,? Super T>c)

এটি index return করে , না পেলে -1 return করে।

অনুশীলনী

১. একটি প্রোগ্রাম লিখ যা একটি string input নেবে অবং এতে কতগুল unique character আসে তা প্রিন্ট করবে।

২. একটি facebook application simulate করার জন্য প্রোগ্রাম লিখ

* এতে এচ্ছে মত user তৈরি করা যাবে।
* একটি ইউজার অন্য ইউজার কে friend request দিতে পারবে।
* একটি নিদিষ্ট সময়ে status দিতে পারবে।
* ইউজার অন্য ইউজার এর status এ like & comments করতে পারবে।
* ইউজার অন্য ইউজার কে poke করতে পারবে।