313. What are Java collections

The java Collections Framework is a collection of Interfaces and classes which helps in storing and processing the data efficiently.

This framework has several useful classes which have tons of useful functions which makes a programmer task super easy.

**Collections**

List

A list is an ordered collection(sometimes called a sequence). List may contains duplicate elements.

Below are classes implement List Interface

Array List

Linked List

Vector

Set

A set is a collection that cannot contain duplicate elements.

How ever it makes no guarantees concerning the order of iteration

Below are classes implement set Interface

Hashset

LinkedHashSet

TreeSet

MAP

A map is an object that maps keys to values. A map cannot contain duplicate keys.

Below are main implementations of MAP interfaces

HashMap

TreeMap

LinkedHashMap

314. Implementation of Arrays List

Create one new class (arrayListexample)

So all this collection framework comes under util package. So without importing this Java util, you cannot work on whole collection interfaces.

So first of all, how do you tell to Array list that what kind of objects you are sending into it?

what kind of data types you are passing into this array list, please mention here.

I passing string That’s why I put it as <String> or <Integer>

once we define the size, you cannot again decrease or increase the size, it's a fixed for array.

whereas in array list you could add or delete elements anytime from the list.

It's a dynamic size array,

Interview que : array list has a dynamic size where you can increase by adding or decrease by removing.

315. Examples of ArrayList

package coreJava;

import java.util.ArrayList;

public class arrayListexample {

// all the classes which implements list interface can accept duplicate values

//ArraList,LinkedList,vector- Implementing List interface - all these three classes can accept duplicate values.

//array have fixed size where as arraylist can grow dynamically

//you can access and insert any value in any index

int i =5;

public static void main(String[] args)

{

// TODO Auto-generated method stub

ArrayList<String> a=new ArrayList<String>();

a.add("rahul");

a.add("java");

a.add("java"); //duplicate value

System.out.println(a);

a.add(0, "student");

System.out.println(a);

/\*a.remove(1); // 1 index no

a.remove("java");

System.out.println(a);\*/

System.out.println(a.get(2)); // 2 is index no

// testing

System.out.println(a.contains("testing")); // this testing string word is present or not

System.out.println(a.indexOf("rahul")); // index no of rahul

System.out.println(a.isEmpty()); // false

System.out.println(a.size()); //3

}

protected void abc() {

// TODO Auto-generated method stub System.out.println("hello"); }

}

317. Implementation of Set interface

Difference b/w set and list is

//Set interface does not accept duplicate values

// there is no guarantee that elements stored in sequential order.(random order)

There is no way you could specifically pick one object based upon its index in set. Whereas in ArrayList, we have control on what index to place on what particular string.

Ur adding one item to cartSo ur add same item again in to your cart. Probably the number one will be changed to two. in that case, you will use ArrayList interface,

because it accepts duplicate. You cannot use Set there.

So there is one more case where you want, you need to recharge your mobile, and you have sent details, you cannot recharge at a time, right, two times So transaction might decline for second time. There, people go with the Set interface. So it all depends upon where exactly you are trying to, based upon the scenario you work upon.

318. Examples of HashSet using Iterator

Iterator method -  if you want to iterate through each and every string present in the Hasset, then there is an interface called iterator, which will help you to Traverse through each and every object present in the set.

Hasnext() - it'll check whether the next index is present or not, If it is present, then it returns true Then it goes inside and print the value. So till the index is present,

it'll keep on loop and it prints the values.

this is how you can iterate to the objects present in the set interface with the help of iterator.

//Hashset, Treeset, LinkedHashset these 3 classes implements set interface

//does not accept duplicate values

//there is guarantee elements stored in sequential order(random order)

HashSet<String> hs = new Hashset<String>();

hs.add(“USA”);

hs.add(“UK”);

hs.add(“INDIA”);

hs.add(“he”);

hs.add(“she”);

hs.add(“INDIA”); // duplicate it should not accept if you print it will display only one India

println(hs);

//println(hs.remove(“UK”));

Println(hs.isEmpty());

Println(hs.size());

Iterator<String> i=hs.iterator(); // highlighted is the written type of the iterator method

While(i.hasNext())

{

Println(i.next());

}

List accept duplicate values, whereas set do not accept the duplicate values.

List guarantees order, but set does not guarantee the sequential order.

It made the return values in a random order. So based upon the your requirement,

you may select which is the best suitable collection for you.

It is a arraylist or hashset or treeset

319. Implementation of Map interface

we are actually placing the values and indexing arrayList input takes care of that IN ARRAYlist but whereas in HashMap, we map key to the value.

MAP Interface takes the values in the form of key value pair.

 when you are actually pushing an object into this HashMap collection, you need to pass also key, that means at what index I need to place this value, key value pair.

put is a method where first arguments take key value, and second arguments takes the value

which will go into that key, basically it's an indexing, when you call the key,

automatically value present in that key will be called

when you use this entry set,this each key and value will be stored as an set index,

when I say entry set, that means I'm saying that make all these key value pairs as an set.

 this set have the HashMap values, and object for this set is sn.

if I want to traverse to each and every index of this set, ( sn.iterator)

Interview que : create a hash table and pass the table to the set collection,

and print the key and value separately using the iterator.

note : If there is no key for the given value then the output will be null.

320. Code explaining hashset and hashmap

package coreJava;

import java.util.HashMap;

import java.util.Iterator;

import java.util.Map;

import java.util.Map.Entry;

import java.util.Set;

public class hashMapexample {

public static void main(String[] args) { // TODO Auto-generated method stub

HashMap<Integer,String> hm=new HashMap<Integer,String>(); // passed two arguments

hm.put(0, "hello"); // key : 0 value : hello

hm.put(1, "Gudbye");

hm.put(42, "morning");

hm.put(3, "evening");

System.out.println(hm.get(42));

hm.remove(42);

System.out.println(hm.get(42)); // null

Set sn= hm.entrySet();

Iterator it =sn.iterator(); //hashtable -pass table set collections

while(it.hasNext()) {

System.out.println(it.next());

// I want to separate key and I need to separate value as well, if I want to separate those two

then you need to cast that with Map.entry class, okay, if you cast in that way

then you could separate both key comma value,

Map.Entry mp=(Map.Entry)it.next();

// System.out.println(mp.getKey());

System.out.println(mp.getValue()); }

}}

321. Difference between HashMap and HashTable (Interview )

synchronous vs asynchronous in java – Async is multi-thread,which means operations or programs can run in parallel. Sync is a single thread, so only one operation or program will run at a time.

When you execute something synchronously, you wait for it to finish before moving on to another task. When you execute something asynchronously, you can move on to another task before it finishes.

Hashtable<Integer,String> hm = new Hashtable<Integer,String>();

so there is one more collection called HashTable but it lies most in the similar lines of HashMap only just in a map you could place the table.

 if multiple programs simultaneously accessing this HashMap then HashMap, keep on updating.

 let's say there are five threads which working on this HashMap that means five different programs are threads, can access this HashMap at a time concurrently, that means there is no synchronization(If any thread fails while running all the threads it won’t stop it execute the remaining threads) but whereas when you work on a HashTable if one program is accessing this HashTable the other program need to wait till the first program releases the HashTable resources. That is the main difference.

So that is the reason we say HashTable as an thread safe. That means the operation will be performed only once the other thread completes its task on HashTable. Whereas in HashMap, the multiple threads or multiple programs can access the HashMap

at a time and made it as non synchronized.

1.Synchronization or Thread Safe : This is the most important difference b/w two.

HashMap is non Synchronized and not thread safe. On the other hand, HashTable is Thread Safe & Synchronized.

When to use HashMap ? - answer is If your application do not require any multi-threading task, in other words hasmap is better for non-threading applications. HashTable should be used in multithreading applications.

2.Null keys and null values : Hashmap allows one null key and any number of null values, while Hashtable do not allow null keys and null values in the HashTable object.

3. Iterating the values – Hashmap object values are iterated by using iterator. HashTable is the only class other than vector which uses enumerator to iterate the values of HashTable Object.

322. Practice Exercise : Printing unique number- Amazon interview ques

! - I will give one not condition here.

Not in the sense here, this true negation, which will turn as a false

323. programming code download

package demopack;

import java.util.ArrayList;

public class collectiondemo {

public static void main(String[] args) {

// TODO Auto-generated method stub

int a[] ={ 4,5,5,5,4,6,6,9,4}; //9

// Print unique number from the list- Amazon  // how many time this number repeated in to the list

//print the string in reverse

ArrayList<Integer> ab =new ArrayList<Integer>();

for(int i=0;i<a.length;i++) { // // intialize the I value will start with zero it will execute until // a.length = 9

int k=0;

if(!ab.contains(a[i])) { // // for i = 0, a[i] = 4 for the 1st

ab.add(a[i]);

k++;

for(int j=i+1;j<a.length;j++) {

if(a[i]==a[j]) { // //here i =0,j=1 then a[i]=4,a[j]=5 for 1st iteration

k++;

} }

// System.out.println(a[i]);

//System.out.println(k);

if(k==1)

System.out.println(a[i]+"is unique number"); // 9 is unique number

} } }

}

324. OOPS Interview questions

**OOPS Interview questions**

1. What are the core concepts of OOPS?

OOPS core concepts are;

* 1. Abstraction
  2. Encapsulation
  3. Polymorphism
  4. Inheritance
  5. Composition
  6. Association
  7. Aggregation

1. What is Abstraction?

Abstraction is an OOPS concept to construct the structure of the real world objects. During this construction only the general states and behaviors are taken and more specific states and behaviors are left aside for the implementers.

1. What is Encapsulation?

Encapsulation is an OOPS concept to create and define the permissions and restrictions of an object and its member variables and methods. A very simple example to explain the concept is to make the member variables of a class private and providing public getter and setter methods. Java provides four types of access level modifiers: public, protected, no modifier and private.

1. What is the difference between Abstraction and Encapsulation?
   1. “Program to interfaces, not implementations” is the principle for Abstraction and “Encapsulate what varies” is the OO principle for Encapsulation.
   2. Abstraction provides a general structure of a class and leaves the details for the implementers. Encapsulation is to create and define the permissions and restrictions of an object and its member variables and methods.
   3. Abstraction is implemented in Java using interface and abstract class while Encapsulation is implemented using four types of access level modifiers: public, protected, no modifier and private.
2. What is Polymorphism?

Polymorphism is the occurrence of something in various forms. Java supports various forms of polymorphism like polymorphic reference variables, polymorphic method, polymorphic return types and polymorphic argument types.

1. What is Inheritance?

A subclass can inherit the states and behaviors of it’s super class is known as inheritance.

1. What is multiple inheritance?

A child class inheriting states and behaviors from multiple parent classes is known as multiple inheritance.

1. What is the diamond problem in inheritance?

In case of multiple inheritance, suppose class A has two subclasses B and C, and a class D has two super classes B and C.If a method present in A is overridden by both B and C but not by D then from which class D will inherit that method B or C? This problem is known as diamond problem.

1. Why Java does not support multiple inheritance?

Java was designed to be a simple language and multiple inheritance introduces complexities like diamond problem. Inheriting states or behaviors from two different type of classes is a case which in reality very rare and it can be achieved easily through an object association.

1. What is Static Binding and Dynamic Binding?

Static or early binding is resolved at compile time. Method overloading is an example of static binding.

Dynamic or late or virtual binding is resolved at run time. Method overriding is an example of dynamic binding.

1. What is a Class?

A class is the specification or template of an object.

1. What is an Object?

Object is instance of class.

325. Java interview questions

**Java interview questions**

What is Runtime Polymorphism?

Runtime polymorphism or dynamic method dispatch is a process in which a call to an overridden method is resolved at runtime rather than at compile-time.

In this process, an overridden method is called through the reference variable of a super class. The

 What is the difference between abstraction and encapsulation?

Abstraction hides the implementation details whereas encapsulation wraps code and data into a single unit.

 What is abstract class?

A class that is declared as abstract is known as abstract class. It needs to be extended and its method implemented. It cannot be instantiated.

 Can there be any abstract method without abstract class?

No, if there is any abstract method in a class, that class must be abstract.

Can you use abstract and final both with a method?

No, because abstract method needs to be overridden whereas you can't override final method.

Is it possible to **instantiate**the abstract class?

No, abstract class can never be instantiated.

**What is interface?**

Interface is a blueprint of a class that have static constants and abstract methods.It can be used to achieve fully abstraction and multiple inheritance.

 Can you declare an interface method static?

No, because methods of an interface is abstract by default, and static and abstract keywords can't be used together.

 Can an Interface be final?

No, because its implementation is provided by another class.

 What is marker interface?

An interface that have no data member and method is known as a marker interface.For example Serializable, Cloneable etc.

 What is difference between abstract class and interface?

|  |  |
| --- | --- |
| **Abstract class** | **Interface** |
| 1)An abstract class can have method body (non-abstract methods). | Interface have only abstract methods. |
| 2)An abstract class can have instance variables. | An interface cannot have instance variables. |
| 3)An abstract class can have constructor. | Interface cannot have constructor. |
| 4)An abstract class can have static methods. | Interface cannot have static methods. |
| 5)You can extends one abstract class. | You can implement multiple interfaces. |

 Can we define private and protected modifiers for variables in interfaces?

No, they are implicitly public.

 When can an object reference be cast to an interface reference?

An object reference can be cast to an interface reference when the object implements the referenced interface.