Just wait. Waiting for some more attendes. Can you see my screen

Java interview Questions

1. Difference between abstract and interface

|  |  |
| --- | --- |
| Can have abstract and non abstract methods | Only abstract methods |
| 0-100% abstraction | Only 100% abstraction |
| Not multiple inheritance | multiple inheritance |
| Can have static/non static and final/non final variables. | Can only have static and final variables |
| Keyword extends | implements |
| Can provide the implementation of interface | Can’t provide the implementation of abstract class |
| Can have static method, main method and constructor | Can’t have static method, main method and constructor |

1. What is encapsulation and abstraction.

**Ensapsulation:**

It is like a wrapper that secure data and code. We can create it by declaring the variables private and providing getter and setters. It hides the functionality from the user too.

public class Employee{

private String name;

private int id;

public String getName(){

return name;

}

public int getId(){

return id;

}

public String setName(String name){

this.name= name;

}

public String setId(int id){

this.id= id;

}

public static void main (String[] args) {

Employee emp1 = new Employee();

Employee emp2 = new Employee();

Emp1.setName(“Geeta”);

Emp1.setId (20);

System.out .println(emp.getId());

}

**Abstraction:** It is hiding the data but functionality is visible to the user. It can be achieved by abstract classes and interfaces.

Abstract Class

* Methods can only be declared but not defined
* Can have variables and methods like normal class.
* Can have constructor but can not be called directly as abstract class can’t be instantiated.
* Even with one abstract method the class need to be called abstract class.
* Cannot create objects of abstract class.

// abstract class

public abstract class machine {

public abstract void start ();

public abstract void stop ();

}

// concrete class

public class Camera extends Machine {

public void start () {

System.out.prinln(“starting car”);

}

public void stop () {

System.out.prinln(“stoping car”);

}

}

**Interface:** It can have only abstract methods and static, final variables. No constructor. We can achieve multiple inheritance through interface.

interface Phone{

public abstract void call();

}

public Iphone implements phone, {

public void call () {

system.out.println(“ Calling”)

}

}

public static void main ( String [] args){

Iphone ph = new Iphone();

ph.call();

}

1. What is polymorphism.

**Polymorphism:**

Different form. Could be 2 types overloading and overriding.

**Overloading :** Different type or no of parameters. Also known as early binding/ static polymorphism/compile time polymorphism

public class Overloading{

public void print(int i){

System.out.println(i);

}

public void print(String s){

System.out.println(s);

}

public void print(int i, String s){

System.out.println(i,s);

}

public static void main ( String [] args ) {

Overloading ov = new Overloading();

ov.print(20,“Geeta”);

}

}

**Overriding:** whenever is a relationship. Also known as Dynamic polymorphism/run time polymorphism.

public class Computer {

public void memory () {

System.out.print (“ any memory”);

}

}

public class Mac extends Computer {

public void memory () {

System.out.print ( “ 16GB”);

}

}

public static void main (String [] args){

Computer comp = new Computer ();

Mac mac = new Mac ();

Comp.memory(); // output is 16GB

1. Difference between overloading and overriding

|  |  |
| --- | --- |
| Overloading | Overriding |
| Always within the class | Between 2 class where there is –a relationship |
| Signature is change | Same signature |
| Can be static, final | No static |
| Compile time polymorphism | Run time polymorphism |
| Return type can be changed | Cannot change the return type |

1. What is inheritance?

It is about the reusability of the class. It shows is-a relationship.

Child inherit all the features of parent class. Java supports single inheritance but no multiple inheritance.

C extends A print

C extends B print

Single level inheritance

Multilevel inheritance

A-------🡪B ----------C

Class A{

Int n1,n2,result;

Public void add(){

result = n1 +n2;

}

}

class B extends A{

public void sub() {

result = n1-n2

}

}

}

Class C extends B{

}

public static void main ( String [] args ) {

B ch = new B();

A pa = new A();

Pa.

ch.n1 = 6;

ch.n2 = 3;

ch.add();

ch.sub();

}

parent class is also called super and base class

child class is called sub and derived

1. Why String is immutable?

If 2 or more string is pointed to the same object, changing on one string value will change everything.

There will be no hashmap if string is mutable

1. What is Static Keyword in java.

It is used for memory management. It belongs to class but not object. We can access static variables and methods using class name. we cannot access non static variables /methods inside a static method because static variables do not need objects and they are loaded at the time your class get loaded. But we can access then by creating a object. We can use it when we are going to use that variable or method multiple times, we don’t want to create the object multiple time eg. Util class.

1. What is final

Final is the keyword in java which is used to restrict the user. It can be used with variable, class or method.

Final variable – once set we can’t change the value of the variable.

Final class – It can’t be inherited.

Final method – it can’t be override.

A constructor can not be final as it can not be inherited.

1. Difference between string, stringBuffer and stringBuilder

|  |  |  |
| --- | --- | --- |
| **String** | StringBuffer | StringBuilder |
| Store in constant string pool | Heap memory | Heap memory |
| immutable | mutable | mutable |
| Thread safe | threadsafe | Not thread |
| fast | Very slow | fast |

1. Difference between == and equals method for string

== is for reference comparison. If both reference points to the same object it will return true. If same or difference object have same content then equals will come true.

String s1 = new String(“hello”);

String s2 = new String(“hello”);

s1 == s2; false

s1.equals(s2); true

1. Diff b/n final, finally and finalize.

**Fina**l: final is a keyword which can be used for class, method or variable.

**Finally**: it is a block which is associated with try/catch. This block is always executed even if an unexpected exception occurs. It is usually a cleanup code where programmer want to avoid the return and break for the code.

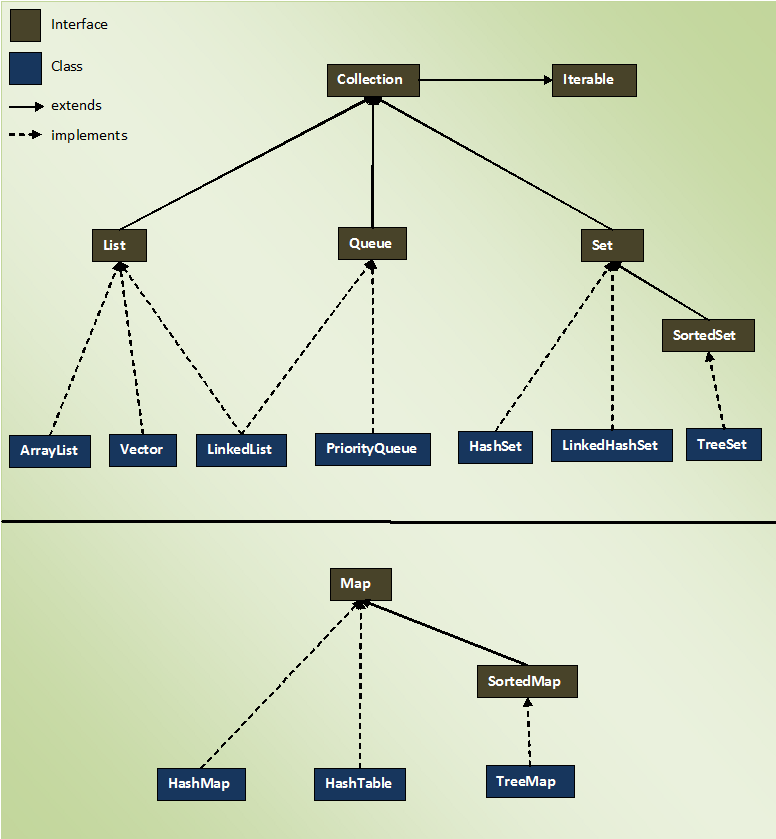
**Finalize**: it is a method. Before an object is garbage collected, the runtime system calls its finalize() method. It is invoked by garbage collector just before destroying an object to perform cleanup activities.

1. What is try/catch/final used for

This is used for exception handling. Comes under checked exceptions

1. What different collections are present in java ?

Collections are group of objects stored in well defined manner.Using collection framework we can store and manipulate the group of objects. All classes and interfaces related to collection are placed in java.util package.



1. Difference between list, set, map

|  |  |  |
| --- | --- | --- |
| list | set | map |
| Can hold duplicate elements . | No duplicate elements in set. | It is key value pair. Value can be duplicate but keys are always unique. |
| List is ordered means it maintains insertion order. | Set is unordered but LinkedHashSet maintains the insertion order. | Map is unordered too but LinkedhashMap maintains insertion order. TreeMap maintains sorting order |
| List allows multiple null elements | Set allows single null element because it contains unique elements. | Map allows one null key but multiple values. But HashTable dosen’t allow null key neither null values. |

1. Difference between arraylist, vector and linkedlist?

|  |  |
| --- | --- |
| **ArrayList** | L**inkedList** |
| It is resizable | Not resizeable |
| Internally uses dynamic array to store the elements | Internally uses doubly linked list to store the elements. |
| Mainly used in case of storing and retrievel of data | Mainly for insertion and deletion of the data. |
| Memory consumption is low | It is high because of doubly linked list |

|  |  |
| --- | --- |
| **ArrayList** | **Vector** |
| Not a legacy class | It is a legacy class |
| It is not synchronized | It is synchronized |
| Can grow up to half of its size | Can be double of its size |
| It is fast as it is not synchronized | Slow |
| Uses iterator interface to iterate | Mainly use enumeration interface but can also use Iterator too. |

1. Diff b/n hashmap, hashtable, treemap and linkedhashmap

|  |  |  |  |
| --- | --- | --- | --- |
| HashMap | HashTable | TreeMap | LinkedHashMAp |
| Doesn’t provide any ordering | Also dosen’t provide any ordering | Treemap provides ordering | It provides ordering. It can be used to maintain insertion order |
| It is fast compared to hashtable as it is not syncronized | It is slow as it is syncronized |  | Not synchronized but slower than hashmap |
| Allows one null key and multiple null values | No null key or value | Treemap sorts elemnts in natural order so also dosen’t allow null keys | Allows null key and null values |
| Not synchronized and thread unsafe | Synchronized and safe | Not synchronized and thread unsafe | Not synchronized and thread unsafe |

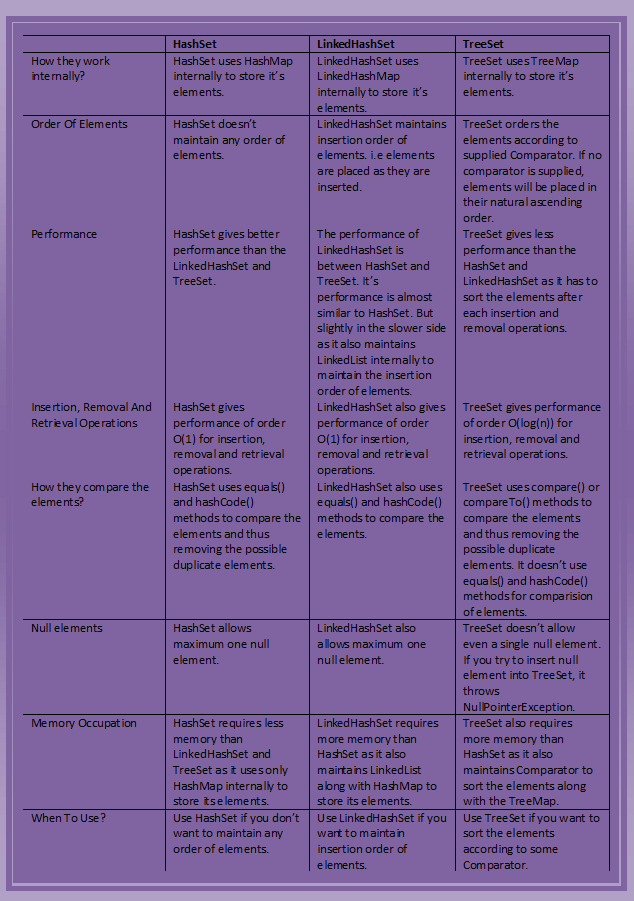
1. What is importance of hashcode and equals with respect to hash collections?

[**hashCode()**](https://docs.oracle.com/javase/7/docs/api/java/lang/Object.html#hashCode%28%29) and [**equals()**](https://docs.oracle.com/javase/7/docs/api/java/lang/Object.html#equals%28java.lang.Object%29) methods have been defined in Object class which is parent class for java objects. For this reason, all java objects inherit a default implementation of these methods.

HashCode() method is used to get a unique integer for given object. This integer is used for determining the bucket location, when this object needs to be stored in some [HashTable](https://en.wikipedia.org/wiki/Hash_table) like data structure. By default, Object’s hashCode() method returns and integer representation of memory address where object is stored.

equals() method, as name suggest, is used to simply verify the equality of two objects. Default implementation simply check the object references of two objects to verify their equality.

1. Difference between hashset, linkedhashset and tree set



1. What is typecasting, explicit and implicit cast eg int -🡪 float, float🡪 int

Type casting in java or simply casting is used to convert data from one data type to another data type. By using casting, data can not be modified but only type of data can be modified.

increasing order of their memory size of primitive data

**byte < short < int < long < float < double.**

**Implicit casting**: also known as Auto widening. It is when converting data from small sized to big sized data type.

Int i;

float j = i;

Return j;

Int will automatically convert to float

**Explicit casting** : or explicit narrowing. Reverse of implicit. When we convert the data from big sized to small sized.

float i;

int j = (int) i; // float is explicitly narrowed to int

1. What is metod to convert string to integer and integer to string.

We can use Integer.parseInt() to convert string to primitive int

String s1 = “15”;

int i = Integer.parseInt(s1); // output will be 15.

To convert string into integer object we can use Integer.valueOf()

String s1 = “15”;

int i = Integer.valueOf(s1);

it will also give you the outcome of 15

If the string dose not contain a parsable integer eg string contains some characters it will throw NumberFormatException .

1. String s1 = “hello”; constant pool

String s2 = “hello”;

String s3 = new String(“hello”); // heap memory

String s4 = s3.intern(); //intern to convert in string constant pool. It checks in pool and creates only reference to existing string

What is the output

1. S.o.p.(s1.equals(s2);

true

1. S.o.p(s1==s2);

True

1. S.o.p(s1.equals(s3));

True

1. S.o.p(s1==s3);

False

1. S.o.p(s2==s4);

True

1. What is the use of finally? When finally block is not executed in java?

Finally – though there is any exception in try catch finally still executes. It is used for clean up or close of resources or for any value assignments

The only times finally won't be called are:

if you call System.exit() *or*

if the JVM crashes first

if there is an infinite loop in the try block

if the power turns off

if system.exit – throws security exception - finally will still execute.

1. Can a class be abstract and final together?

No the class can’t be together because abstract demands to be inherited and final can’t be inherited so both can’t be together.

1. Can we override static method in java?

you can not override private or static method in Java, if you create similar method with same return type and same method arguments that's called method hiding.

1. What is the root class in exception hierarchy?

Throwable is the root class. They are from the java.lang package.