**Q.What is oops concept?**Ans-> Object-Oriented Programming is a methodology or paradigm to design a program using classes and objects.   
-> It simplifies the software development and maintainance by providing some concepts:  
1. Object  
2. Class  
3. Inheritance  
4. Polymorphism  
5. Abstraction  
6. Encapsulation

**Object:**=======  
-> Instance of a class is known as an Object whereas instance is nothing but allocating sufficient amount of memory   
space for the data members of a class.  
-> Any entity that has state and behavior is known as an object.   
-> For example: chair, pen, table, keyboard, bike etc.   
-> It can be physical and logical.

**Class:**======  
-> The process of binding data-members and associated methods in a single unit is called class.  
-> Collection of objects is nothing but class. It is logical entity.

**Inheritance:**===========  
-> When an object acquires all the properties and behaviors of parent class object.   
i.e., known as inheritance. It provides code-reusability. It is used to achieve runtime polymorphism.

**Polymorphism:**==============  
-> When one task is performed by different ways. i.e., known as polymorphism.  
-> The process of representing "one form in multiple forms" whereas One form represents  
original method and it always resides in Base class. Multiple forms represent overriden  
methods and they always resides in derived class.  
Ex: void sum()------>BC void sum()----->DC1 void sum()--------->DC2  
{ { {  
a=10; b=20; f1=10.5f; f2=12.5f; c1='A'; c2='B';  
S.o.p(a+b); S.o.p(f1+f2); S.o.p(c1+c2);  
} } }  
Ex: To convince the customer differently.  
To draw something eg: shape or rectangle.  
-> In java, we use method overloading and method overriding to achieve polymorphism.  
-> Another example can be to speak something e.g. cat speaks meaw, dog barks woof etc.

**Abstraction: (Security Purpose)**===========  
-> Highlighting set of services what we are offering and hiding implementation details is nothing but Abstraction. Example: ATM.  
-> Hiding internal details and showing functionality is known as abstraction.   
-> For example: phone call, we don't know the internal processing.  
-> In java, we use abstract class and interface to achieve abstraction.

**Encapsulation:**=============  
-> Binding (or wrapping) code and data together into a single unit is known as encapsulation.   
-> For example: capsule, it is wrapped with different medicines.  
-> A java class is the example of encapsulation. Java bean is the fully encapsulated class   
-> because all the data members are private here.

**Q.What is Interface:**  
===================  
-> The interface in java is a mechanism to achieve abstraction.   
-> There can be only abstract methods in the java interface not method body.   
-> It is used to achieve abstraction and multiple inheritance in Java.  
-> Java Interface also represents IS-A relationship.  
**Why use Java interface?:**1. It is used to achieve abstraction.  
2. By interface, we can support the functionality of multiple inheritance.  
3. It can be used to achieve loose coupling.  
**Abstraction: (Security Purpose)**===========  
-> Highlighting set of services what we are offering and hiding implementation details is nothing but Abstraction. Example: ATM.  
-> Hiding internal details and showing functionality is known as abstraction.   
-> For example: phone call, we don't know the internal processing.  
-> In java, we use abstract class and interface to achieve abstraction.  
**Interface**=================  
1. If we don't know anything about implementation just we have requirement specification  
then we should go for Interface.  
2. Every methods inside Interface is always public and abstract.  
3. We can't declare Interface methods with the following modifiers: private, protected,   
static, final, synchronized, native and strictfp.  
4. Every variable present inside Interface must be public, static, final whether we are declaring  
or not.  
5. We can't declare Interface variables with the following modifiers private, protected, transient  
and volatile.  
6. For Interface variables compulsory we should perform initialization at the time of declaration,  
otherwise we will get compile time error.  
7. Inside Interface we can't declare static and instance blocks.  
8. Inside Interface we can't take constructor.  
==========================================================================================  
**Abstract Class**=============  
1. If we are talking about implementation but not completely (partial implementation)  
then we should go for Abstract class.  
2. Every method present inside Abstract Class need not be public & abstract, we can  
take concrete methods also.  
3. There are no restrictions on abstract class method modifiers.  
4. Every variable present inside Abstract Class need not be public static and final.  
5. There are no restrictions on Abstract class variable modifiers.  
6. For Abstract Class variables which is not required to perform initialization at the time of declaration.  
7. Inside Abstract Class we can declare static and instance blocks.  
8. Inside Abstract Class we can take constructor.

**Q.What is difference between HashMap and HashTable?**================================================================================  
Ans- HashMap Hashtable

**HashMap  
Hashtable**1. No method present inside HashMap is synchronized.  
2. Every methods inside Hashtable is synchronized.  
3. At a time multiple threads are allowed to operate on HashMap object simultaneously and hence it is not a Threadsafe.  
4. At a time only one thread is allowed to operate onHashtable object and hence it is Threadsafe.  
5. Relatively Performance is high.  
6. Relatively Performance is low.  
7. Null is allowed for both keys and values.

8. Null is not allowed for both keys and values. Otherwisewe will get NullPointerException.

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

**Q.What is difference between HashMap and LinkedHashMap?**

**HashMap LinkedHashMap**=====================================================================================

**HashMap  
LinkedHashMap**1. Underlying data structure is Hashtable.  
1. Underlying data structure is combination of Hashtable Underlying data structure is combination. of Hashtable  
2. Insertion order is not preserved.  
2. Insertion order is preserved.  
3. Introduced in 1.2 version.  
3. Introduced in 1.4 version.

**IdentityHashMap**==================  
-> It is exactly same as HashMap except the following difference.  
-> In case of HashMap, JVM will use .equals(-) method to identify duplicate keys, which is meant for content comparison.  
-> But in case of IdentityHashMap, JVM will use == operator to identify duplicate keys which is meant for reference comparison.

**WeakHashMap**==================  
-> It is exactly same as HashMap except the following difference.  
-> In case of HashMap, if an object associated with HashMap then it is not eligible for garbage collection,  
eventhough it doesn't contain any external references. i.e., HashMap dominates Garbage Collector.  
-> But in case of WeakHashMap, if an object doesn't contain any references then it is always eligible for   
Garbage Collector eventhough it is associated with WeakHashMap. ie., Garbage Collector dominates WeakHashMap.

**Q.How to implement your own HashMap?**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Ans- Hash Map is a implementaion of Map Interface.  
HashMap works on the principal of hashing.  
-> Map.Entry interface - This interface gives a map entry (key-value pair).  
HashMap in Java stores both key and value object, in bucket,   
as an object of Entry class which implements this nested interface Map.Entry.  
-> hashCode() -HashMap provides put(key, value) for storing and get(key) method  
for retrieving Values from HashMap. When put() method is used to store (Key, Value) pair,   
HashMap implementation calls hashcode on Key object to calculate a hash   
that is used to find a bucket where Entry object will be stored.   
When get() method is used to retrieve value, again key object is used to calculate a hash   
which is used then to find a bucket where that particular key is stored.  
-> equals() - equals() method is used to compare objects for equality.  
In case of HashMap key object is used for comparison, also using equals() method   
Map knows how to handle hashing collision (hashing collision means more than   
one key having the same hash value, thus assigned to the same bucket.  
In that case objects are stored in a linked list,   
Where hashCode method helps in finding the bucket where that key is stored,  
equals method helps in finding the right key as there may be more than one key-value pair   
stored in a single bucket.  
\*\* Bucket term used here is actually an index of array, that array is called table in HashMap   
implementation. Thus table[0] is referred as bucket0, table[1] as bucket1 and so on.

So it is implementaion of HashMap

**QHow to implement your own ArrayList?**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
Ans-> ArrayList is a Resizable-array of an Objects.  
In Array we don't have the capability of dynamically increasing the size of the array  
but in ArrayList we can, We know that ArrayList has initial capacity of the element  
to be added in a list, so first things we need is default initial capacity.   
second things we need an Array of Objects to store the element of the ArrayList.

->Define default initial capacity of the CustomArrayList.  
-private static final int DEFAULT\_INITIAL\_CAPACITY = 5;  
->Now declare an empty CustomArrayList which will return while creating empty CustomArrayList  
-private static final Object[] EMPTY\_ELEMENT\_DATA = {};  
->Define the size for the CustomeArrayList  
-private int size;  
->Now we need to define an Array of an Object to store the elements view sourceprint?  
-private transient Object[] customArrayListElementData;

So it is implementaion of ArrayList.

**Q. Synchronized keyword**=========================================================================================================================  
Ans--> Synchronized modifier is applicable only for methods and blocks but not for classes and variables.

-> If multiple threads are operating simultaneously on same java object then there may be a chance of data inconsistency problem.  
-> To overcome this problem we should go for synchronized.  
-> If a method or block declared as synchronized then at a time only one thread is allowed to operate method or block on the given object.  
-> So that we can resolve data inconsistency problem.  
-> The main advantage of synchronized keyword is we can overcome data inconsistency problems.  
-> But the main disadvantage is it increases waiting time of threads and creates performance problems.  
-> Hence if there is no specific requirement then it is never recommanded to use synchronized keyword.  
-> Synchronized method should compulsory contain implementation whereas abstract method should not contain implementation.  
-> Hence [abstract synchronized] combination is illegal for methods.

**Q.4. Diff\_Serialization&Externalization.**Ans-\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
The process of converting system representation to network representation is called an marshalling.  
The process of converting data from network representation to sysytem representation is called un marshalling.  
\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**Serialization:**================  
the process of seprating data from an object is called serialization.  
or  
-> The process of writing state of an object to a file is called Serialization.  
-> But strictly speaking, it is a process of converting an object from java supported form to either file supprted form  
or network supported form or database supported form is called Serialization.   
-> By using FileOutputStream and ObjectOutputStream, we can achieve Serialization.

It is meant for default Serialization.  
Here everything takes care by JVM and Programmer doesn't have any control.  
In Serialization, total object will be serialized always & it is not possible to serialize part of the object.  
Relatively performance is low.  
Serialization is best choice if we want to save total objects to the file.  
Serialization Interface doesn't contain any methods.  
So it is called as Marker Interface.\*\*\*\*\*\*  
transient keyword will play role in Serialization.

**Step to create the serialization:**-----------------------------  
step1 create the serialized class and its object.  
ex:- class employee inplements serializable{  
int emp id=101;  
String emp name=ram;  
----  
-----  
}  
step2 create fileOutput stream.  
FileOutputStream fos=new fileoutputStream("abc.txt");

Step3 create object outputStream  
for that we have to use the following constructor.  
public objectoutputstream(FileOutPutStream fos)  
ex: objectoutputStream oos=new objectoutputStream(fos);

step4 write serializable object in object outputstream:

we have to use this method for that.  
public void writeobject(object obj)  
ex: Employee emp=new employee();  
oos.writeobject(emp);

**Desirialization:  
==================**The process of regenerating the object on the basis of data is called the deserialization.

**Step to perform the des:**-------------------------------  
step1 create fileInputStream.  
ex: fileInputStream fis=new fileInputStream("abc.txt");

Step2 create objectInputStream:  
to create objectInputStream we have to use the following constructor.

public objectInputStream(FileInputStream fis)

ex: objectInputStream ois=new objectInputStream(fis);

Step3 read des obj from objectInputStream:

for that we have to use the method.  
public object readObject()  
ex:  
Employee emp=(Employee)ois.read object();

**Externalization:**=================  
It is meant for customized Serialization.  
Here everything takes care by Programmer and JVM doesn't have any control.  
In Externalization, based on our requirement we can save either total object or part of the object.  
Relatively performance is high.  
Externalization is best choice if we want to save part of the object to the file.  
Externalization Interface contains 2 methods. 1. writeExternal(-) and 2. readExternal(-)  
So it is not a Marker Interface.  
transient keyword will not play any role in Externalization.

**Marker Interface:**==================  
-> Marker interface in Java is interfaces with no field or methods or in simple word empty interface in java is called marker interface.   
-> Example of marker interface is Serializable, Clonnable and Remote interface.  
-> It looks they are used to indicate something to compiler or JVM.   
-> So if JVM sees a class is Serializable it done some special operation on it,   
similar way if JVM sees one class is implement Clonnable it performs some operation to support cloning.   
Same is true for RMI and Remote interface. So in short Marker interface indicate, signal or a command to Compiler or JVM.  
-> After introduction of Annotation on Java5, Annotation is better choice than marker interface and JUnit is a perfect example of using Annotation.  
-> In summary marker interface in Java is used to indicate something to compiler, JVM or any other tool but Annotation is better way of doing same thing.

**Q. equals&==operator.**Ans-1. What is difference between == operator and .equals() method?  
== Opertor .equals() method  
=====================================================================================

= = equals  
**.equlas**1. It is an operator, applicable for both premitives and Object types.  
1. It is a method, applicable only for Object types but not premitives.  
2. In general == operator meant for reference/address comparision.  
2. in general .equals method is used for content comparision.  
3. It is not possible to override == operator for content comparision.  
3. It is possible to override equals(-) method for content comparision.

Note:\*\*.equals method present in object class also meant for reference comparision only based on our requirement we can override fro content comparision.  
\*\* In string class all wrapper class and all collection classes .equals() is overridden for content comparision.

**Q.Singleton Java class:**====================================  
Ans-> The Java class that allows us to create only one object per JVM is called singleton java class.  
-> Situations to take singleton java class:  
a. If class is not having any state.  
b. If class is having only sharable read only state (final variable).  
c. If class is having huge amount of state and performing write operations on that state in synchronized manner (1 Thread at a time).

**Creation of our own Singleton class**=======================  
-> We can create our own singleton class, by using  
private constructor,   
static variables,   
static methods   
we can implement singleton classes.

Example:  
public class Test implements Cloneable  
{  
private static Test t;  
private Test()  
{  
}  
public static Test getInstance()  
{  
if(t==null)  
{  
t=new Test();  
}  
return t;  
}  
public Object clone()  
{  
return this;  
}  
}

Now ,   
Test t1=Test.getInstance();   
Test t2=Test.getInstance();  
Test t3=Test.getInstance();  
|  
|  
Test t100=Test.getInstance();

**Like this, Only one object is created here with multiple references.**========================================================================================  
-> If Constructor is not private then outside person can create object directly by calling constructor.  
In that case they can create multiple objects and also we will miss singleton behavior.  
-> If 't' is not private then after creation of first object outside person can reassign 't' with null. In   
that case a second new object will be created, whenever we call getInstance() method again.  
Test t1=Test.getInstance();  
Test.t=null;  
Test t2=Test.getInstance();  
-> If we are not overriding clone() method & Test class implements Cloneable then Object class   
clone() method will be executed which provides always a separate new object.  
Test t1=Test.getInstance();  
Test t2=(Test)t1.clone();  
========================================================================================

**Advantage of Singleton Class:**==========================  
Instead of creating multiple objects we can run entire show with only one object but with multiple references.   
Hence the main advantage of singleton class is performance will be improved and we know object creation is most precious.  
**Q.7. StringPool V.V.I**Ans--> There is string pool in java which is used for managing the memory efficiently and save the money invested in the memory.  
-> There are two types of declaring strings in java:  
String Literal  
String Object.  
Example:  
String s="ABC"; //String Literal  
String s1="ABC"; //String Literal   
String sq=new String("ABC"); //String Object  
String sq1=new String("ABC"); //String Object  
Both stm are use to create string object in java.  
\*\*  
When we use String s="Abc" then string object will be created in the string constant pool SCP area. it is one of memory in method area.  
when String literal is created in java, JVM checks the string pool if the string content is already present then again memory is not allocated.  
So, It optimizes the use of memory.  
String sq1=new String("ABC");  
But this is not true of u create a string using Option 2. Then the String object is created on heap and each time you create a string object,  
a new object is created irrespective of its content.  
So, same thing is not applies to String objects. String objects shown in example sq and sq1 occupy different location in memory.

\*\*\*\*\*\* Below are the hard requirements of an immutable object.

Make the class final  
make all members final, set them explicitly, in a static block, or in the constructor  
Make all members private  
No Methods that modify state  
Be extremely careful to limit access to mutable members(remember the field may be final but the object can still be mutable.   
ie private final Date imStillMutable). You should make defensive copies in these cases.

**Q.8. Diff\_yield,join&sleep.**Ans-  
**sleep()**========  
-> If a thread doesn't want to perform any operation for a particular amount of time that is just  
pausing is required then we should go for sleep() method.  
-> sleep(n) says "I'm done with my timeslice, and please don't give me another one for at least n milliseconds."  
The OS doesn't even try to schedule the sleeping thread until requested time has passed.  
-> It is static and final method.  
-> It can be overloaded and throws InterruptedException also.  
**join()**========  
-> If a thread wants to wait until completing other threads then we should go for join() method.  
-> It is final method but not static method and native method.  
-> It can be overloaded and throws InterruptedException.

**yield()**=========  
-> It causes to pause current executing thread to give the chance for remaining waiting threads of same priority.  
-> yield() says “I’m done with my timeslice, but I still have work to do.” The OS is free to immediately give the   
thread another timeslice, or to give some other thread or process the CPU the yielding thread just gave up.  
-> It is static method and native method but not final method.  
-> It doesn't throw InterruptedException and it can't be overloaded.  
**wait()**-> wait() method releases the lock.  
-> wait() is the method of Object class.  
-> wait() is the non-static method.  
-> wait() should be notified by notify() or notifyAll() methods.

**Difference between wait and sleep method**===========================================  
-> A wait can be "woken up" by another thread calling notify on the monitor which is being waited on  
whereas a sleep cannot it can only be interrupted.  
-> While sleeping a Thread does not release the locks it holds, while waiting releases the lock on the   
object that wait() is called on.  
-> sleep(n) says "I'm done with my timeslice, and please don't give me another one for at least n milliseconds."  
The OS doesn't even try to schedule the sleeping thread until requested time has passed.  
.wait() says "I'm done my timeslice. Don't give me another timeslice until someone calls notify()."  
-> we can normally use sleep() for time-synchronization and wait() for multi-thread-synchronization.

**Q. Why we use super keyword?**=============================================================  
Ans--> Whenever we inherit the base class features into derived class there is a possibility  
that base class features are similar to derived class features.  
-> Due to this, JVM get an ambiguity problem.  
-> In order to differentiate between Base class features and Derived class features, for  
Base class features must be always proceed by super keyword.

\*\*\*  
-> In other words, super is a keyword which is used for differentiate between Base class  
features and Derived class features.  
\*\*\*\*\*  
-> super keyword is playing an important role in three places. They are:  
1. At variable level  
2. At method level  
3. At constructor level

**Q.What is the difference between Checked and UnChecked Exception?**==============================================================  
-> Checked Exception: The Exceptions which are checked by compiler for smooth execution of the program at runtime are called Checked Exception.  
-> Compiler will check whether we are handling checked exceptin or not. If we are not handling then we will get compile time error.  
For example: HallTicketMissingException, FileNotFoundException, IOException etc...  
-> Unchecked Exception: The Exceptions which are not checked by compiler whether the programmer handling or not are called Unchecked Exception.  
RuntimeException and its child classes, Error and its child classes are unchecked except these the remaining are checked exception.  
For example: BombBlastException, ArithmeticException, NullPointerException etc...  
-> Checked Exception is required to be handled by compile time while Unchecked Exception doesn't.  
-> Checked Exception is direct sub-Class of Exception while Unchecked Exception are of RuntimeException and Error.

**Q.How can you create our own Exception?**===========================================================================  
-> If you are creating our own Exception that is known as custom exception or user-defined exception.   
-> Java custom exceptions are used to customize the exception according to user need.  
-> By the help of custom exception, we can have our own exception and message.  
-> Let's see a simple example of java custom exception.

class TooYoungException extends RuntimeException  
{  
TooYoungException(String s)  
{  
super(s);  
}  
}  
class CustomExceptionTest  
{  
public static void main(String args[])  
{  
int age=17;  
if(age<18)  
throw new TooYoungException("You are too Young..Just wait for "+(18-age)+" years");  
else  
System.out.println("You are eligible to vote.");  
}  
}

**Q. What is the difference between length and length()?**==========================================================================  
-> length is a final variable, applicable only for arrays.   
whereas length() is a method, applicable only for String object.  
-> length variable represents the size of array.  
whereas length() returns the number of characters present in String.

**Q.13.ClassNotFound & ClassNoDefFound.**Ans-  
-> It is an exception. It is of type java.lang.Exception.  
where as It is an error. It is of type java.lang.Error.  
-> It occurs when an application tries to load a class at run time which is not updated in the classpath.  
where as It occurs when java runtime system doesn’t find a class definition,which is present at compile time, but missing at run time.  
-> It is thrown by the application itself. It is thrown by the methods like Class.forName(),  
loadClass() and findSystemClass().  
where as It is thrown by the Java Runtime System.  
-> It occurs when classpath is not updated with required JAR files.  
where as It occurs when required class definition is missing at runtime.

\*\*\*\* ClassNotFoundException \*\*\*   
Ex.

public class MainClass  
{  
public static void main(String[] args)  
{  
try  
{  
Class.forName("oracle.jdbc.driver.OracleDriver");  
}catch (ClassNotFoundException e)  
{  
e.printStackTrace();  
}  
}  
}

\*\*\*\*\*\*\*\* NoClassDefFoundError \*\*\*\*\*\*\*  
Ex....  
class A  
{  
// some code  
}  
public class B  
{  
public static void main(String[] args)  
{  
A a = new A();  
}  
}

**Q.\*\* What is immutable class..??**=================================

-> Immutable class is a class which once created, it’s contents can not be changed.  
-> Immutable objects are the objects whose state can not be changed once constructed.  
e.g. String class.  
-> All wrapper classes in java.lang are immutable –   
String, Integer, Boolean, Character, Byte, Short, Long, Float, Double.

**\*\*\* Why use immutable class..??**=====================================

-> Because java uses the concept of string literal. Suppose there are 5 reference variables,all referes to one object "sachin".  
If one reference variable changes the value of the object, it will be affected to all the reference variables.   
That is why string objects are immutable in java.

-> In case of String object, just because of SCP a single object is referred by multiple references.  
by using one reference if we are allowed to change the content then remaining refereces will be impacted.  
to overcome this problem some people made string object as immutable.  
....  
-> But in case of StringBuffer for every requirment a separate object will be created.  
by using one reference if we are allowed to change the content then remaining refereces won't be impacted.  
hence immutability concept is not required for StringBuffer object.

**\*\*\*How to create an immutable class???**=============================================

-> Create a final class.  
Set the values of properties using constructor only.  
Make the properties of the class final and private.  
Do not provide any setters for these properties.  
If the instance fields include references to mutable objects, don't allow those objects to be changed:  
Don't provide methods that modify the mutable objects.  
Don't share references to the mutable objects.  
Never store references to external, mutable objects passed to the constructor;  
if necessary, create copies, and store references to the copies. Similarly,   
create copies of your internal mutable objects when necessary to avoid returning the originals in your methods.  
-> Ex..  
public final class FinalPersonClass {   
private final String name;   
private final int age;   
public FinalPersonClass(final String name, final int age) {   
this.name = name;   
this.age = age;   
}   
public int getAge() {   
return age;   
}   
public String getName() {   
return name;   
}   
}

**Q.String & StringBuffer & StringBuilder.**Ans-  
\*\*\*\* String \*\*\*\*

-> String is immutable ( once created can not be changed )object.   
The object created as a String is stored in the Constant String Pool .   
Every immutable object in Java is thread safe ,that implies String is also thread safe .  
String can not be used by two threads simultaneously.  
String once assigned can not be changed.

**\*\*\*\* StringBuffer \*\*\***

-> StringBuffer is mutable means one can change the value of the object .   
The object created through StringBuffer is stored in the heap .   
StringBuffer has the same methods as the StringBuilder ,   
but each method in StringBuffer is synchronized that is StringBuffer is thread safe .

**\*\*\*\*\*\* StringBuilder \*\*\*\***

-> StringBuilder is same as the StringBuffer ,  
that is it stores the object in heap and it can also be modified .   
The main difference between the StringBuffer and StringBuilder is that StringBuilder is also not thread safe.   
StringBuilder is fast as it is not thread safe .

**Q.Reflection API:**=================================  
-> Reflection API is used for reading a class at execution time, loading its byte code  
and creating its object and further calling its datamembers and member methods.  
-> Reflection API contains all information about our class that is what types of modifiers,  
constructors, inner classes, methods etc...  
-> Reflection acts as Mirror

**Q.19 Why String class is final .**Ans-5 Reasons of Why String is final or Immutable in Java  
Why String class is made Immutable or Final in JavaThough true reasons of why String class was made final is best known to Java designers,  
and apart from that hint on security by James Gosling, I think following reasons also suggest Why String is Final or Immutable in Java.  
**1) String Pool**Java designer knows that String is going to be most used data type in all kind of Java applications and that's  
why they wanted to optimize from start. One of key step on that direction was idea of storing String literals in String pool.   
Goal was to reduce temporary String object by sharing them and in order to share, they must have to be from Immutable class.  
You can not share a mutable object with two parties which are unknown to each other.   
Let's take an hypothetical example, where two reference variable is pointing to same String object:  
String s1 = "Java";  
String s2 = "Java";  
Now if s1 changes the object from "Java" to "C++", reference variable also got value s2="C++", which it doesn't even know about it.  
By making String immutable, this sharing of String literal was possible. In short,  
key idea of String pool can not be implemented without making String final or Immutable in Java.

**2) Security**Java has clear goal in terms of providing a secure environment at every level of service and String is critical in those whole security stuff.   
String has been widely used as parameter for many Java classes, e.g. for opening network connection,  
you can pass host and port as String, for reading files in Java you can pass path of files and directory as String and for opening database connection,  
you can pass database URL as String. If String was not immutable, a user might have granted to access a particular file in system,   
but after authentication he can change the PATH to something else, this could cause serious security issues.   
Similarly, while connecting to database or any other machine in network, mutating String value can pose security threats.   
Mutable strings could also cause security problem in Reflection as well, as the parameters are strings.  
**3) Use of String in Class Loading Mechanism**Another reason for making String final or Immutable was driven by the fact that it was heavily used in class loading mechanism.   
As String been not Immutable, an attacker can take advantage of this fact and a request to load standard Java classes   
e.g. java.io.Reader can be changed to malicious class com.unknown.DataStolenReader. By keeping String final and immutable,  
we can at least be sure that JVM is loading correct classes.  
**4) Multithreading Benefits**Since Concurrency and Multi-threading was Java's key offering, it made lot of sense to think about thread-safety of String objects.  
Since it was expected that String will be used widely, making it Immutable means no external synchronization,   
means much cleaner code involving sharing of String between multiple threads.   
This single feature, makes already complicate, confusing and error prone concurrency coding much easier.  
Because String is immutable and we just share it between threads, it result in more readable code.

**5) Optimization and Performance**Now when you make a class Immutable, you know in advance that, this class is not going to change once created.   
This guarantee open path for many performance optimization e.g. caching. String itself know that, I am not going to change,   
so String cache its hashcode. It even calculate hashcode lazily and once created, just cache it. In simple world,  
when you first call hashCode() method of any String object, it calculate hash code and all subsequent call to hashCode() returns already calculated,   
cached value. This results in good performance gain, given String is heavily used in hash based Maps e.g.  
Hashtable and HashMap. Caching of hashcode was not possible without making it immutable and final, as it depends upon content of String itself.  
**Q.Difference between Array and ArrayList ?**

**Throw  
Throws**1. Java throw keyword is used to explicitly throw an exception.  
1. Java throws keyword is used to declare an exception  
2. Checked exception cannot be propagated using throw only.  
2. Checked exception can be propagated with throws.  
3. Throw is followed by an instance.  
3. Throws is followed by class.  
4. Throw is used within the method.  
5. You cannot throw multiple exceptions.

4. Throws is used with the method signature.  
5. You can declare multiple exceptions e.g.  
public void method()throws IOException,SQLException.

**Q.Difference between method overloading and method overriding in java ?**

Method Overloading  
Method Overriding  
1) Method Overloading occurs with in the same Method Overriding occurs between  
class two classes superclass and subclass

2) Since it involves with only one class inheritance  
Since method overriding occurs between superclass and subclass inheritance is involved.  
is not involved.  
3)In overloading return type need not be the same  
3) In overriding return type must be same.  
4) Parameters must be different when we do  
4) Parameters must be same.  
overloading  
5) Static polymorphism can be acheived using  
5) Dynamic polymorphism can be acheived using method overriding.  
method overloading  
6) In overloading one method can’t hide the  
6) In overriding subclass method hides that of the superclass method.  
another

**Pros and Cons of String being Immutable or Final in Java**---------------------------------------------------------------  
Apart from above benefits, there is one more advantage that you can count due to String being final in Java.   
It's one of the most popular object to be used as key in hash based collections e.g. HashMap and Hashtable.  
Though immutability is not an absolute requirement for HashMap keys, its much more safe to use Immutable object as key than mutable ones,   
because if state of mutable object is changed during its stay inside HashMap, it would be impossible to retrieve it back,  
given it's equals() and hashCode() method depends upon the changed attribute. If a class is Immutable, there is no risk of changing its state,  
when it is stored inside hash based collections. Another significant benefits, which I have already highlighted is its thread-safety.   
Since String is immutable, you can safely share it between threads without worrying about external synchronization.   
It makes concurrent code more readable and less error prone.

Despite all these advantages, Immutability also has some disadvantages, e.g. it doesn't come without cost.   
Since String is immutable, it generates lots of temporary use and throw object, which creates pressure for Garbage collector.  
Java designer has already thought about it and storing String literals in pool is their solution to reduce String garbage.   
It does help, but you have to be careful to create String without using constructor e.g. new String() will not pick object from String pool.   
Also on average Java application generates too much garbage. Also storing Strings in pool has a hidden risk associated with it.   
String pool is located in PermGen Space of Java Heap, which is very limited as compared to Java Heap.   
Having too many String literals will quickly fill this space, resulting in java.lang.OutOfMemoryError: PermGen Space. Thankfully,   
Java language programmers has realized this problem and from Java 7 onwards, they have moved String pool to normal heap space,   
which is much much larger than PermGen space. There is another disadvantage of making String final, as it limits its extensibility.   
Now, you just can not extend String to provide more functionality, though more general cases its hardly needed,   
still its limitation for those who wants to extend java.lang.String class.  
**Q.Final vs Finally vs Finalize.**Ans-Final--> it is modifier applicable for classes,methods,and variables.  
Class----> if will declae the class as final then we cannot extend that class.or say we cant create child class for that.  
Variable--->That will be constant and we canot perform re-assignment for that variable.  
Methods----> that method we cant overdide in child class.

**Q. Difference between ArrayList and LinkedList ?**

ArrayList  
LinkedList  
1.For retrieval operation ArrayList is  
preferable  
1.For Middle insertion or middle modification  
LinkedList is preferable  
2.Memory shifting or Memory management is  
complicated  
2.Memory shifting or Memory management is  
not required  
3. ArrayList internally uses dynamic  
array to store the elements.  
3. LinkedList internally uses doubly  
linked list to store the elements.

**Q.Difference between Array and ArrayList ?**

Array  
ArrayList  
1. Array is not Grow able in nature.  
1.ArrayList is grow able in nature  
2.It allow only homogeneous object  
2.It allow both homogeneous and  
heterogeneous object  
3.Inbuilt methods are not there in Array  
3.Inbuilt methods are there provided by utile  
class  
4.Performance wise Array is best  
5.Memory wise Array is not good

4.Performance wise ArrayList is preferable  
5.Memory wise ArrayList is not good in comparison Array

**Q. How can we take list into map?**Example: public Map<List<String>, List<Double>> getData() { Map<List<String>, List<Double>> dataMap = new HashMap<>();  
List<String> l1Key = new ArrayList<>();  
l1Key.add("Basant");  
l1Key.add("Babul");

List<String> l2Key = new ArrayList<>();  
l2Key.add("Manoj"); l2Key.add("Amit"); l2Key.add("Saroj");

List<Double> l1Value = new ArrayList<>();  
l1Value.add(50.000);

l1Value.add(40.000);

List<Double> l2Value = new ArrayList<>();  
l2Value.add(90.000); l2Value.add(60.000); l2Value.add(30.000);

dataMap.put(l1Key, l1Value); dataMap.put(l2Key, l2Value); return dataMap;

}

**Q. How can we take Map into List?**Example:

public List<Map<String, Integer>> setData() {  
List<Map<String, Integer>> listMap = new ArrayList<>(); Map<String, Integer> map = new HashMap<>();  
map.put("A", 34);  
map.put("D", 90); map.put("B", 12); map.put("C", 91); listMap.add(map);

return listMap;

}

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

**Q.RequestDispatcherVsSendRedirect.**  
Ans-requestDispatcher - forward() method  
====================================  
-> When we use forward method, request is transfer to other resource within the same server for further processing.  
-> In case of forward, web container handle all process internally and client or browser is not involved.  
-> When forward is called on requestdispatcher object we pass request and response objects so our old request object   
is present on new resource which is going to process our request.  
-> Visually we are not able to see the forwarded address, it is transparent.  
-> Using forward () method is faster than send redirect.  
-> When we redirect using forward and we want to use same data in new resource we can use request.setAttribute () as we have request object available.

**SendRedirect**====================================  
-> In case of sendRedirect, request is transfer to another resource to different domain or different server for further processing.  
-> When you use sendRedirect, container transfers the request to client or browser so URL given inside the sendRedirect method   
is visible as a new request to the client.  
-> In case of sendRedirect call, old request and response objects are lost because it’s treated as new request by the browser.  
-> In address bar, we are able to see the new redirected address. It’s not transparent.  
-> sendRedirect is slower because one extra round trip is required, because completely new request is created and old request object is lost.   
Two browser request required.  
-> But in sendRedirect, if we want to use we have to store the data in session or pass along with the URL.  
sWhich one is good?  
=====================  
-> Its depends upon the scenario that which method is more useful.  
-> If you want control is transfer to new server or context and it is treated as completely new task then we go for Send Redirect.   
Generally, forward should be used if the operation can be safely repeated upon a browser reload of the web page will not affect the result.  
**Q.What are the difference between ServletConfig and ServletContext Object?**=====================================================================================  
-> The ServletConfig parameters are specified for a particular servlet and are unknown to other servlets.   
-> It is used for initialising purpose.  
-> The ServletContext parameters are specified for an entire application outside of any particular servlet   
and are available to all the servlets within that application.  
-> So we can say ServletConfig object is one per Servlet class.  
-> ServletContext object is global to entire web application.  
-> Object of ServletConfig will be created during initialization process of the servlet.  
-> Object of ServletContext will be created at the time of web application deployment.  
-> In web.xml, <init-param> tag will be appear under <servlet-class> tag.  
-> In web.xml, <context-param> tag will be appear under <web-app> tag.  
-> ServletConfig object will be destroyed once the servlet execution is completed.  
-> ServletContext object will be destroyed once the application is removed from the server.  
**Q.WebServer&AppServer.**Ans- Web Server vs Application Server  
=====================================================================================  
-> I have web application and I have to run web application , Web Server is required.  
-> WebServer provides environment to run only Web application.  
-> Web Server can provide support for Servlet, JSP, Html, etc...  
it support servlet and jsp only.  
-> I have enterprise application and i have to run enterprise application, Application Server is required.  
-> Application Server provides environment to run both web application and enterprise application.  
-> Application Server is superior server. Every Application Server contain inbuilt Web server to provide  
the support for Web application.  
-> Any technology, we can use to develop enterprise application then we can go for Application Server.  
For ex: Servlet, Jsp, Html, + EJB, JMS etc...  
-> J2EE compatible server is Application Server.  
it support distributed txn and EjB.