**7. What are different ways to create String Object?**

There arev arious ways you can create a STRING OBJECT in java:

We can create String objects by using String **literal** .

for ex: String Str = "Hello world";

By using "new" Keyword ,this is the common way to create a String object in java.

for ex : String str 1 = new String ("Hello");

Using character Array - Charater Array can ne converted in string .

for ex: char ch[] = { 'H','E','L','L','O',};

String str1 = new String (ch);

**10. What is String subSequence method? \***

String subsequence method is a built in function in Java that returns a char Sequence . Char Sequence is the subsequence of this sequence. The subsequence starts with the char value at the specifies index and ends with the char value at( end -1)

The syntax of the subsequence methods is:

sub Sequence()

the subsequence method takes two parameters

start index - the starting index

end index - the ending index

the subsequence returns a char sequence.

**Q11. How to Split String in java? \***

The String has a built-in method for splitting strings:

String[] split​(String regex) - splits the string around matches of the given regular expression

String[] split​(String regex, int limit) - splits this string around matches of the given regular expression

The method returns an array of split strings.

for ex: class StrSplit{

public static void main(String []args){

String strMain = "Sunnyvale, Cupertino, Mountian view, Fremont, Newark";

String[] arrSplit = strMain.split(", ");

for (int i=0; i < arrSplit.length; i++)

{

System.out.println(arrSplit[i]);

}

}

}

Output:

Sunnyvale

Cupertino

Mountian view

Fremont

Newark

**Q37. What is JVM and explain me the Java memory allocation \***

Java Virtual Machine (JVM) is a engine that provides runtime environment to drive the Java Code or applications. It converts Java bytecode into machines language. JVM is a part of Java Run Environment (JRE). In other programming languages, the compiler produces machine code for a particular system. However, Java compiler produces code for a Virtual Machine known as Java Virtual Machine.

Memory Allocation in Java

Memory Allocation in Java is the process in which the virtual memory sections are set aside in a program for storing the variables and instances of structures and classes. However, the memory isn’t allocated to an object at declaration but only a reference is created. For the memory allocation of the object, new() method is used, so the object is always allocated memory on the heap.

The Java Memory Allocation is divided into following sections :

Heap

Stack

Code

Static

This division of memory is required for its effective management.

The code section contains your bytecode.

The Stack section of memory contains methods, local variables, and reference variables.

The Heap section contains Objects (may also contain reference variables).

The Static section contains Static data/methods.

**Q38. What is Polymorphism and encapsulation?**

Single task can be done in different way.

Method overloading(compile time polymorphism),method overriding(run time polymorphism)

**method overriding**

Specific implementation of a method for child class.

**method overloading**

If a class have multiple methods by same name but different parameters, it is known as Method Overloading.

**Encapsulation**

 The process of binding data and corresponding methods (behavior) together into a single unit is called encapsulation in Java. In other words, encapsulation is a programming technique that binds the class members (variables and methods) together and prevents them from being accessed by other classes, thereby we can keep variables and methods safes from outside interference and misuse.  
  
If a field is declared private in the class then it cannot be accessed by anyone outside the class and hides the fields within the class. Therefore, Encapsulation is also called data hiding.

**Q39. What is method overloading and Method over riding?**

There are many differences between method overloading and method overriding in java. Method Overloading Method Overriding

1) Method overloading is used to increase the readability of the program. Method overriding is used to provide the specific implementation of the method that is already provided by its super class.

2) Method overloading is performed within class.

Method overriding occurs in two classes that have IS-A (inheritance) relationship.

3) In case of method overloading, parameter must be different.

In case of method overriding, parameter must be same.

4) Method overloading is the example of compile time polymorphism.

Method overriding is the example of run time polymorphism.

5) In java, method overloading can't be performed by changing return type of the method only. Return type can be same or different in method overloading. But you must have to change the parameter.

Return type must be same or covariant in method overriding.

**Q40. Why string is Immutable?**

The String class is immutable, so that once it is created a String object cannot be changed.

If it is modified then new object will be created If there is a necessity to make a lot of modifications to Strings of characters, then you should use String Buffer & String Builder Classes.

**Q41. What is the difference between String and String buffer? \***

**String**

String is a Class in java and defined in java.lang package. It’s not a primitive data type like int and long. String class represents character Strings. String is used in almost all the Java applications and there are some interesting facts we should know about String. String in immutable and final in Java and JVM uses String Pool to store all the String objects.

String buffer n string builder StringBuffer and StringBuilder are mutable classes. StringBuffer operations are thread-safe and synchronized where StringBuilder operations are not thread-safe. So in a multi-threaded environment, we should use StringBuffer but in the single-threaded environment, we should use StringBuilder.  
StringBuilder performance is fast than String Buffer because of no overhead of synchronization.

**Q42. What is the difference between array and array list?**

Array: Simple fixed sized arrays that we create in Java, like below

int arr[] = new int[10]

ArrayList : Dynamic sized arrays in Java that implement List interface.

ArrayList<Type> arrL = new ArrayList<Type>();

Here Type is the type of elements in ArrayList to

be created

Differences between Array and ArrayList

An array is basic functionality provided by Java. ArrayList is part of collection framework in Java. Therefore array members are accessed using [], while ArrayList has a set of methods to access elements and modify them.

Array is a fixed size data structure while ArrayList is not. One need not to mention the size of Arraylist while creating its object. Even if we specify some initial capacity, we can add more elements.

Array can contain both primitive data types as well as objects of a class depending on the definition of the array. However, ArrayList only supports object entries, not the primitive data types.

Note: When we do arraylist.add(1); : it converts the primitive int data type into an Integer object.

Since ArrayList can’t be created for primitive data types, members of ArrayList are always references to objects at different memory locations (See this for details). Therefore in ArrayList, the actual objects are never stored at contiguous locations. References of the actual objects are stored at contiguous locations.

In array, it depends whether the arrays is of primitive type or object type. In case of primitive types, actual values are contiguous locations, but in case of objects, allocation is similar to ArrayList.

Java ArrayList supports many additional operations like indexOf(), remove(), etc. These functions are not supported by Arrays.

**Q43. What is the difference between hash map and Hash table?**

**Hashmap vs Hashtable**

HashMap and Hashtable store key/value pairs in a hash table. When using a Hashtable or HashMap, we specify an object that is used as a key, and the value that you want linked to that key. The key is then hashed, and the resulting hash code is used as the index at which the value is stored within the table.

1. HashMap is non synchronized. It is not-thread safe and can’t be shared between many threads without proper synchronization code whereas --Hashtable is synchronized. It is thread-safe and can be shared with many threads.

2. HashMap allows one null key and multiple null values whereas --Hashtable doesn’t allow any null key or value.

3. HashMap is generally preferred over HashTable if thread synchronization is not needed

Why HashTable doesn’t allow null and HashMap does?

To successfully store and retrieve objects from a HashTable, the objects used as keys must implement the hashCode method and the equals method. Since null is not an object, it can’t implement these methods. HashMap is an advanced version and improvement on the Hashtable. HashMap was created later.

**Q46.What is a vector in Java?**

The java.util.Vector class implements a growable array of objects. Similar to an Array, it contains components that can be accessed using an integer index. Following are the important points about Vector −

The size of a Vector can grow or shrink as needed to accommodate adding and removing items.

Each vector tries to optimize storage management by maintaining a capacity and a capacityIncrement.

As of the Java 2 platform v1.2, this class was retrofitted to implement the List interface.

Unlike the new collection implementations, Vector is synchronized.

This class is a member of the Java Collections Framework.

**Q45. What is set in java? \***

A Set is a Collection that cannot contain duplicate elements. It models the mathematical set abstraction.

The Set interface contains only methods inherited from Collection and adds the restriction that duplicate elements are prohibited.

Set also adds a stronger contract on the behavior of the equals and hashCode operations, allowing Set instances to be compared meaningfully even if their implementation types differ.

The methods declared by Set are summarized in the following table −

Sr.No. Method & Description

1 add( )

Adds an object to the collection.

2 clear( )

Removes all objects from the collection.

3 contains( )

Returns true if a specified object is an element within the collection.

4 isEmpty( )

Returns true if the collection has no elements.

5 iterator( )

Returns an Iterator object for the collection, which may be used to retrieve an object.

6 remove( )

Removes a specified object from the collection.

7 size( )

Returns the number of elements in the collection.

**Q46. What is an abstract class?**

Abstract class: is a restricted class that cannot be used to create objects (to access it, it must be inherited from another class).

**Q47. What is an interface?**

An interface is a completely "abstract class" that is used to group related methods with empty bodies.

Like abstract classes, interfaces cannot be used to create objects (in the example above, it is not possible to create an "Animal" object in the MyMainClass)

Interface methods do not have a body - the body is provided by the "implement" class

On implementation of an interface, you must override all of its methods

Interface methods are by default abstract and public

Interface attributes are by default public, static and final

An interface cannot contain a constructor (as it cannot be used to create objects)

**Q48. Why Java is Platform independent?**

It is a platform independent language i.e. the compiled code can be run on any java supporting platform. It runs on the logic of “Write once, run anywhere”.

The platform that helps to run and develop the programs written in java script is called java platform. Java platform consists of execution engine, a compiler, and a set of libraries. Java programs can run on any OS or processor.

**Q49. What are access modifiers? Give me an example? \***

The access modifiers in Java specifies the accessibility or scope of a field, method, constructor, or class. We can change the access level of fields, constructors, methods, and class by applying the access modifier on it.

There are four types of Java access modifiers:

Private: The access level of a private modifier is only within the class. It cannot be accessed from outside the class.

Default: The access level of a default modifier is only within the package. It cannot be accessed from outside the package. If you do not specify any access level, it will be the default.

Protected: The access level of a protected modifier is within the package and outside the package through child class. If you do not make the child class, it cannot be accessed from outside the package.

Public: The access level of a public modifier is everywhere. It can be accessed from within the class, outside the class, within the package and outside the package.

There are many non-access modifiers, such as static, abstract, synchronized, native, volatile, transient, etc. Here, we are going to learn the access modifiers only.

**Q50. What are java exceptions? Give me an example**

An exception is an unwanted or unexpected event, which occurs during the execution of a program i.e at run time, that disrupts the normal flow of the program’s instructions.

For ex :

class ThrowsExecp{

public static void main(String args[]){

String str = null;

System.out.println(str.length());

}

}

Output :

**Exception in thread "main" java.lang.NullPointerException**

**at ThrowsExecp.main(File.java:8)**

**Q51. What is the difference between throws and throwable?**

**Throws** is also a keyword in java which is used in the method signature to indicate that this method may throw mentioned exceptions. The caller to such methods must handle the mentioned exceptions either using try-catch blocks or using throws keyword. Below is the syntax for using throws keyword.

**Throwable** is a super class for all types of errors and exceptions in java. This class is a member of java.lang package. Only instances of this class or it’s sub classes are thrown by the java virtual machine or by the throw statement. The only argument of catch block must be of this type or it’s sub classes. If you want to create your own customized exceptions, then your class must extend this class.

**Q52. What is the difference between Error and exception? \***

Exceptions and errors both are subclasses of Throwable class.

**Error-** The error indicates a problem that mainly occurs due to the lack of system resources and our application should not catch these types of problems. Some of the examples of errors are system crash error and out of memory error. Errors mostly occur at runtime that's they belong to an unchecked type.

**Exceptions** are the problems which can occur at runtime and compile time. It mainly occurs in the code written by the developers. Exceptions are divided into two categories such as checked exceptions and unchecked exceptions.

**Q53. What is the difference between Error, throwable and exception? \***

All exception and errors types are sub classes of class Throwable, which is base class of hierarchy.One branch is headed by Exception. This class is used for exceptional conditions that user programs should catch. NullPointerException is an example of such an exception.Another branch,Error are used by the Java run-time system(JVM) to indicate errors having to do with the run-time environment itself(JRE). StackOverflowError is an example of such an error.

**Q54. What are collection APIs, give me an example**

The Collection API is a set of classes and interfaces that support operation on collections of objects. These classes and interfaces are more flexible, more powerful, and more regular than the vectors, arrays, and hashtables if effectively replaces. Classes are HashSet, HashMap, ArrayList, LinkedList, TreeSet and TreeMap. Interfaces – Collection, Set, List and Map. java.util.ArrayList class provides resizable-array and implements the List interface. java.util.HashSet class implements the Set interface, backed by a hash table. java.util.BitSet class implements a vector of bits that grows as needed.

**Q55. What is the difference between final and finally?**

**Final** is used to apply restrictions on class, method and variable. Final class can't be inherited, final method can't be overridden and final variable value can't be changed.Final is a keyword.

for ex:

class FinalExample{

public static void main(String[] args){

final int x=100;

x=200;//Compile Time Error

}}

**Finally** is used to place important code, it will be executed whether exception is handled or not.

for ex:

class FinallyExample{

public static void main(String[] args){

try{

int x=300;

}catch(Exception e){System.out.println(e);}

finally{System.out.println("finally block is executed");}

**Q57. What are the different types of interface? (Ans List, set, Queue)** \*

Set, List and Map are three important interfaces of Java collection

The Set interface provides an unordered collection of unique objects, i.e. Set doesn't allow duplicates, while Map provides a data structure based on key-value pair and hashing.

**Q56. Will java supports multiple inheritance?**

When one class extends more than one classes then this is called multiple inheritance. For example: Class C extends class A and B then this type of inheritance is known as multiple inheritance**. Java doesn’t allow multiple inheritance**. Java doesn’t allow multiple inheritance to avoid the ambiguity caused by it.

**Q58. What are wrapper class? Give me an example \***

A Wrapper class is a class whose object wraps or contains primitive data types. When we create an object to a wrapper class, it contains a field and in this field, we can store primitive data types. In other words, we can wrap a primitive value into a wrapper class object.

import java.util.ArrayList;

class Autoboxing

{

public static void main(String[] args)

{

char ch = 'a';

// Autoboxing- primitive to Character object conversion

Character a = ch;

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

// Autoboxing because ArrayList stores only objects

arrayList.add(25);

// printing the values from object

System.out.println(arrayList.get(0));

}

}

Output: 25

**Q59. What is boxing and unboxing in Java? Explain with an example \***

The basic difference between Boxing and Unboxing is that Boxing is the conversion of the value type to an object type whereas, on other hands, the term Unboxing refers to the conversion of the object type to the value type.

Primitive type Wrapper class

boolean Boolean

byte Byte

char Character

float Float

int Integer

long Long

short Short

double Double

**Q60. Explain for each loop \***

For-each is another array traversing technique like for loop, while loop, do-while loop introduced in Java5.It starts with the keyword for like a normal for-loop.

Instead of declaring and initializing a loop counter variable, you declare a variable that is the same type as the base type of the array, followed by a colon, which is then followed by the array name.

In the loop body, you can use the loop variable you created rather than using an indexed array element. It’s commonly used to iterate over an array or a Collections class (eg, ArrayList)

**Q61. What are iterators, explain with an example \***

Iterator is used for iterating (looping) various collection classes such as HashMap, ArrayList, LinkedList etc. In this tutorial, we will learn what is iterator, how to use it and what are the issues that can come up while using it. Iterator took place of Enumeration, which was used to iterate legacy classes such as Vector.

import java.util.ArrayList;

import java.util.Iterator;

public class IteratorDemo1 {

public static void main(String args[]){

ArrayList names = new ArrayList();

names.add("Jyoti");

names.add("Steve");

names.add("Jack");

Iterator it = names.iterator();

while(it.hasNext()) {

String obj = (String)it.next();

System.out.println(obj);

}

}

}

**Q 63. What is multithreading, serialization and Generics in Java \***

**Multithreading** is a Java feature that allows concurrent execution of two or more parts of a program for maximum utilization of CPU. Each part of such program is called a thread. So, threads are light-weight processes within a process.

**Generics** was added in Java 5 to provide compile-time type checking and removing risk of ClassCastException that was common while working with collection classes. The whole collection framework was re-written to use generics for type-safety.

**Serialization** is a mechanism to convert an object into stream of bytes so that it can be written into a file, transported through a network or stored into database. De-serialization is just a vice versa. In simple words serialization is converting an object to stream of bytes and de-serialization is rebuilding the object from stream of bytes. Java Serialization API provides the features to perform serialization & de-serialization. A class must implement java.io .Serializable interface to be eligible for serialization. is a mechanism to convert an object into stream of bytes so that it can be written into a file, transported through a network or stored into database. De-serialization is just a vice versa. In simple words serialization is converting an object to stream of bytes and de-serialization is rebuilding the object from stream of bytes. Java Serialization API provides the features to perform serialization & de-serialization. A class must implement java.io.Serializable interface to be eligible for serialization.

Q35. Write test cases for how to test just the withdrawing functionality from ATM ( Minimum 10 test cases required ) \*

1.Test the ATM machine accepts card and PIN details  
2.Test the error message by inserting a card incorrectly  
3.Test the error message by inserting an invalid card (Expired Card)  
4. Test the error message by entering an incorrect PIN  
5.Test that the user is asked to enter the PIN after inserting a valid ATM Card  
6.Test that PIN is encrypted

7.Test the machine logs out of the user session immediately after successful withdrawal  
8.Test the message when there is no money in the ATM

displayed.  
9.Test the ATM machine successfully takes out the money.  
10.Test the ATM machine takes out the balance printout after the withdrawal

Q36. Write to test scenarios to test Pencil \*

1. Test that the text written with the pencil is readable/legible.
2. Test that the user can write smoothly on different types/quality of paper surfaces.
3. Test that the darkness/color of the text written by pencil is as per the specifications.
4. Test the strength of the lead, it should not break when a specified(normal human) pressure is applied.
5. Test that the text written by pencil can be erased by normal erasers.
6. Test that the quality and strength of the pencil’s wood.
7. Test whether the outer body of the pencil is circular or some polygon shape.
8. Test that the length and radius of the pencil are as per the specification.
9. Test that the weight of the application is as per the specification.
10. Test that the pencil can be sharpened easily by a normal sharpener.