Interview questions

Q1. Explain JDK, JRE and JVM?

JDK vs JRE vs JVM		
JDK	**JRE**	**JVM**
It stands for Java Development Kit.	It stands for Java Runtime Environment.	It stands for Java Virtual Machine.
It is the tool necessary to compile, document and package Java programs.	JRE refers to a runtime environment in which Java bytecode can be executed.	It is an abstract machine. It is a specification that provides a run-time environment in which Java bytecode can be executed.
It contains JRE + development tools.	It's an implementation of the JVM which physically exists.	JVM follows three notations: Specification, **Implementation,** and **Runtime Instance**.

Q2. Explain public static void main(String args[]) in Java.

main() in Java is the entry point for any Java program. It is always written as **public static void main(String[] args)**.

- **public**: Public is an access modifier, which is used to specify who can access this method. Public means that this Method will be accessible by any Class.
- **static**: It is a keyword in java which identifies it is class-based. main() is made static in Java so that it can be accessed without creating the instance of a Class. In case, main is not made static then the compiler will throw an error as **main**() is called by the JVM before any objects are made and only static methods can be directly invoked via the class.
- **void**: It is the return type of the method. Void defines the method which will not return any value.
- **main**: It is the name of the method which is searched by JVM as a starting point for an application with a particular signature only. It is the method where the main execution occurs.
- **String args[]**: It is the parameter passed to the main method.

Q3. Why Java is platform independent?

Java is called platform independent because of its byte codes which can run on any system irrespective of its underlying operating system.

Q4. Why Java is not 100% Object-oriented?

Java is not 100% Object-oriented because it makes use of eight primitive data types such as boolean, byte, char, int, float, double, long, short which are not objects.

Q5. What are wrapper classes in Java?

Wrapper classes convert the Java primitives into the reference types (objects). Every primitive data type has a class dedicated to it. These are known as wrapper classes because they "wrap" the primitive data type into an object of that class. Refer to the below image which displays different primitive type, wrapper class and constructor argument.

Q6. What are constructors in Java?

In Java, constructor refers to a block of code which is used to initialize an object. It must have the same name as that of the class. Also, it has no return type and it is automatically called when an object is created.

There are two types of constructors:

1. **Default Constructor:** In Java, a default constructor is the one which does not take any inputs. In other words, default constructors are the no argument constructors which will be created by default in case you no other constructor is defined by the user. Its main purpose is to initialize the instance variables with the default values. Also, it is majorly used for object creation.
2. **Parameterized Constructor:** The parameterized constructor in Java, is the constructor which is capable of initializing the instance variables with the provided values. In other words, the constructors which take the arguments are called parameterized constructors.

Q7. What is singleton class in Java and how can we make a class singleton?

Singleton class is a class whose only one instance can be created at any given time, in one JVM. A class can be made singleton by making its constructor private.

Q8. What is the difference between Array list and vector in Java?

ArrayList	Vector
Array List is not synchronized.	Vector is synchronized.
Array List is fast as it's non-synchronized.	Vector is slow as it is thread safe.
If an element is inserted into the Array List, it increases its Array size by 50%.	Vector defaults to doubling size of its array.

Array List does not define the increment size.	Vector defines the increment size.
Array List can only use Iterator for traversing an Array List.	Vector can use both Enumeration and Iterator for traversing.

Q9. What is the difference between equals() and == in Java?

Equals() method is defined in Object class in Java and used for checking equality of two objects defined by business logic.

"==" or equality operator in Java is a binary operator provided by Java programming language and used to compare primitives and objects. *public boolean equals(Object o)* is the method provided by the Object class. The default implementation uses == operator to compare two objects. For example: method can be overridden like String class. equals() method is used to compare the values of two objects.

Q10. When can you use the super keyword?

In Java, the super keyword is a reference variable that refers to an immediate parent class object.

When you create a subclass instance, you're also creating an instance of the parent class, which is referenced to by the super reference variable.

The uses of the Java super Keyword are-

1. To refer to an immediate parent class instance variable, use super.
2. The keyword super can be used to call the method of an immediate parent class.
3. Super() can be used to call the constructor of the immediate parent class.

Q11. What makes a HashSet different from a TreeSet?

HashSet	**TreeSet**
It is implemented through a hash table.	TreeSet implements SortedSet Interface that uses trees for storing data.
It permits the null object.	It does not allow the null object.
It is faster than TreeSet especially for search, insert, and delete operations.	It is slower than HashSet for these operations.
It does not maintain elements in an ordered way.	The elements are maintained in a sorted order.
It uses equals() method to compare two objects.	It uses compareTo() method for comparing two objects.
It does not permit a heterogenous object.	It permits a heterogenous object.

Q12. What are the differences between HashMap and HashTable in Java?

HashMap	Hashtable
It is non synchronized. It cannot be shared between many threads without proper synchronization code.	It is synchronized. It is thread-safe and can be shared with many threads.
It permits one null key and multiple null values.	It does not permit any null key or value.
is a new class introduced in JDK 1.2.	It was present in earlier versions of java as well.
It is faster.	It is slower.
It is traversed through the iterator.	It is traversed through Enumerator and Iterator.
It uses fail fast iterator.	It uses an enumerator which is not fail fast.
It inherits AbstractMap class.	It inherits Dictionary class.

Q13. What is the importance of reflection in Java?

Reflection is a runtime API for inspecting and changing the behavior of methods, classes, and interfaces. Java Reflection is a powerful tool that can be really beneficial. Java Reflection allows you to analyze classes, interfaces, fields, and methods during runtime without knowing what they are called at compile time. Reflection can also be used to create new objects, call methods, and get/set field values. External, user-defined classes can be used by creating instances of extensibility objects with their fully-qualified names. Debuggers can also use reflection to examine private members of classes.

Q14. How to not allow serialization of attributes of a class in Java?

The NonSerialized attribute can be used to prevent member variables from being serialized. You should also make an object that potentially contains security-sensitive data nonserializable if possible. Apply the NonSerialized attribute to certain fields that store sensitive data if the object must be serialized. If you don’t exclude these fields from serialisation, the data they store will be visible to any programmes with serialization permission.

Q15. Can you call a constructor of a class inside another constructor?

Yes, we can call a constructor of a class inside another constructor. This is also called as constructor chaining. Constructor chaining can be done in 2 ways-

1. **Within the same class:** For constructors in the same class, the this() keyword can be used.
2. **From the base class:** The super() keyword is used to call the constructor from the base class.
   The constructor chaining follows the process of inheritance. The constructor of the

sub class first calls the constructor of the super class. Due to this, the creation of sub class's object starts with the initialization of the data members of the super class. The constructor chaining works similarly with any number of classes. Every constructor keeps calling the chain till the top of the chain.

Q16. Contiguous memory locations are usually used for storing actual values in an array but not in ArrayList. Explain.

An array generally contains elements of the primitive data types such as int, float, etc. In such cases, the array directly stores these elements at contiguous memory locations. While an ArrayList does not contain primitive data types. An arrayList contains the reference of the objects at different memory locations instead of the object itself. That is why the objects are not stored at contiguous memory locations.

Q17. How is the creation of a String using new() different from that of a literal?
When we create a string using new(), a new object is created. Whereas, if we create a string using the string literal syntax, it may return an already existing object with the same name.

Q18. Why is synchronization necessary? Explain with the help of a relevant example.

Java allows multiple threads to execute. They may be accessing the same variable or object. Synchronization helps to execute threads one after another.
It is important as it helps to execute all concurrent threads while being in sync. It prevents memory consistency errors due to access to shared memory. An example of synchronization code is-

```
public synchronized void increment()
{
a++;
}
```

As we have synchronized this function, this thread can only use the object after the previous thread has used it.

Q19. Explain the term "Double Brace Initialization" in Java?

Double Brace Initialization is a Java term that refers to the combination of two independent processes. There are two braces used in this. The first brace creates an anonymous inner class. The second brace is an initialization block. When these both are used together, it is known as Double Brace Initialization. The inner class has a reference to the enclosing outer class, generally using the 'this' pointer. It is used to do both creation and initialization in a single statement. It is generally used to initialize collections. It reduces the code and also makes it more readable.

Q20. Why is it said that the length() method of String class doesn't return accurate results?

The length() method of String class doesn't return accurate results because it simply takes into account the number of characters within in the String. In other words, code points outside of the BMP (Basic Multilingual Plane), that is, code points having a value of U+10000 or above, will be ignored.

The reason for this is historical. One of Java's original goals was to consider all text as Unicode; yet, Unicode did not define code points outside of the BMP at the time. It was too late to modify char by the time Unicode specified such code points.

Q21. What are the differences between Heap and Stack Memory in Java?

The major difference between Heap and Stack memory are:

Features	Stack	Heap
Memory	Stack memory is used only by one thread of execution.	Heap memory is used by all the parts of the application.
Access	Stack memory can't be accessed by other threads.	Objects stored in the heap are globally accessible.
Memory Management	Follows LIFO manner to free memory.	Memory management is based on the generation associated with each object.
Lifetime	Exists until the end of execution of the thread.	Heap memory lives from the start till the end of application execution.
Usage	Stack memory only contains local primitive and reference variables to objects in heap space.	Whenever an object is created, it's always stored in the Heap space.

Q22. What is a package in Java? List down various advantages of packages.

Packages in Java, are the collection of related classes and interfaces which are bundled together. By using packages, developers can easily modularize the code and optimize its reuse. Also, the code within the packages can be imported by other classes and reused. Below I have listed down a few of its advantages:

- Packages help in avoiding name clashes
- They provide easier access control on the code
- Packages can also contain hidden classes which are not visible to the outer classes and only used within the package
- Creates a proper hierarchical structure which makes it easier to locate the related classes

Q23. Why pointers are not used in Java?

Java doesn't use pointers because they are unsafe and increases the complexity of the program. Since, Java is known for its simplicity of code, adding the concept of pointers will be contradicting. Moreover, since JVM is responsible for implicit memory allocation, thus in order to avoid direct access to memory by the user, pointers are discouraged in Java.

Q24. What is JIT compiler in Java?

JIT stands for Just-In-Time compiler in Java. It is a program that helps in converting the Java bytecode into instructions that are sent directly to the processor. By default, the JIT compiler is enabled in Java and is activated whenever a Java method is invoked. The JIT compiler then compiles the bytecode of the invoked method into native machine code, compiling it "just in time" to execute. Once the method has been compiled, the JVM summons the compiled code of that method directly rather than interpreting it. This is why it is often responsible for the performance optimization of Java applications at the run time.

Q25. What are access modifiers in Java?

In Java, access modifiers are special keywords which are used to restrict the access of a class, constructor, data member and method in another class. Java supports four types of access modifiers:

1. *Default*
2. *Private*
3. *Protected*
4. *Public*

Modifier	Default	Private	Protected	Public
Same class	YES	YES	YES	YES
Same Package subclass	YES	NO	YES	YES
Same Package non-subclass	YES	NO	YES	YES
Different package subclass	NO	NO	YES	YES
Different package non-subclass	NO	NO	NO	YES

Q26. Define a Java Class.

A class in Java is a blueprint which includes all your data. A class contains fields (variables) and methods to describe the behavior of an object. Let's have a look at the syntax of a class.

```
class Abc {
```

```
member variables // class body
methods}
```

Q27. What is an object in Java and how is it created?

An object is a real-world entity that has a state and behavior. An object has three characteristics:

1. State
2. Behavior
3. Identity

An object is created using the ‘new’ keyword. For example:

ClassName obj = new ClassName();

Q28. What is Object Oriented Programming?

Object-oriented programming or popularly known as OOPs is a programming model or approach where the programs are organized around objects rather than logic and functions. In other words, OOP mainly focuses on the objects that are required to be manipulated instead of logic. This approach is ideal for the programs large and complex codes and needs to be actively updated or maintained.

Q29. What are the main concepts of OOPs in Java?

Object-Oriented Programming or OOPs is a programming style that is associated with concepts like:

1. *Inheritance:* Inheritance is a process where one class acquires the properties of another.
2. *Encapsulation:* Encapsulation in Java is a mechanism of wrapping up the data and code together as a single unit.
3. *Abstraction:* Abstraction is the methodology of hiding the implementation details from the user and only providing the functionality to the users.
4. *Polymorphism:* Polymorphism is the ability of a variable, function or object to take multiple forms.

Q30. What is the difference between a local variable and an instance variable?

In Java, a **local variable** is typically used inside a method, constructor, or a **block** and has only local scope. Thus, this variable can be used only within the scope of a block. The best benefit of having a local variable is that other methods in the class won’t be even aware of that variable.

Example

```
if(x > 100)
{
```

```
String test = "Edureka";
}
```

Whereas, an **instance variable** in Java, is a variable which is bounded to its object itself. These variables are declared within a **class**, but outside a method. Every object of that class will create it's own copy of the variable while using it. Thus, any changes made to the variable won't reflect in any other instances of that class and will be bound to that particular instance only.

```
class Test{
public String EmpName;
public int empAge;
}
```

Q31. Differentiate between the constructors and methods in Java?

Methods	Constructors
1. Used to represent the behavior of an object	1. Used to initialize the state of an object
2. Must have a return type	2. Do not have any return type
3. Needs to be invoked explicitly	3. Is invoked implicitly
4. No default method is provided by the compiler	4. A default constructor is provided by the compiler if the class has none
5. Method name may or may not be same as class name	5. Constructor name must always be the same as the class name

In case you are facing any challenges with these Core Java interview questions, please comment on your problems in the section below.

Q32. What is final keyword in Java?

final is a special keyword in Java that is used as a non-access modifier. A final variable can be used in different contexts such as:

- **final variable**

When the final keyword is used with a variable then its value can't be changed once assigned. In case the no value has been assigned to the final variable then using only the class constructor a value can be assigned to it.

- **final method**

When a method is declared final then it can't be overridden by the inheriting class.

- **final class**

When a class is declared as final in Java, it can't be extended by any subclass class but it can extend other class.

Q33. What is the difference between break and continue statements?

break	continue
1. Can be used in switch and loop (for, while, do while) statements	1. Can be only used with loop statements
2. It causes the switch or loop statements to terminate the moment it is executed	2. It doesn't terminate the loop but causes the loop to jump to the next iteration
3. It terminates the innermost enclosing loop or switch immediately	3. A continue within a loop nested with a switch will cause the next loop iteration to execute

Example break:

```
for (int i = 0; i < 5; i++)</div>
<div>
<pre>{
if (i == 3)
{
break;
}
System.out.println(i);
}
```

Example continue:

```
for (int i = 0; i < 5; i++)
{
if(i == 2)
{
continue;
}
System.out.println(i);
}
```

Q34. What is an infinite loop in Java? Explain with an example.

An infinite loop is an instruction sequence in Java that loops endlessly when a functional exit isn't met. This type of loop can be the result of a programming error or may also be a deliberate action based on the application behavior. An infinite loop will terminate automatically once the application exits.

For example:

```
public class InfiniteForLoopDemo
{
public static void main(String[] arg) {
for(;;)
System.out.println("Welcome to Edureka!");
// To terminate this program press ctrl + c in the console.
}
}
```

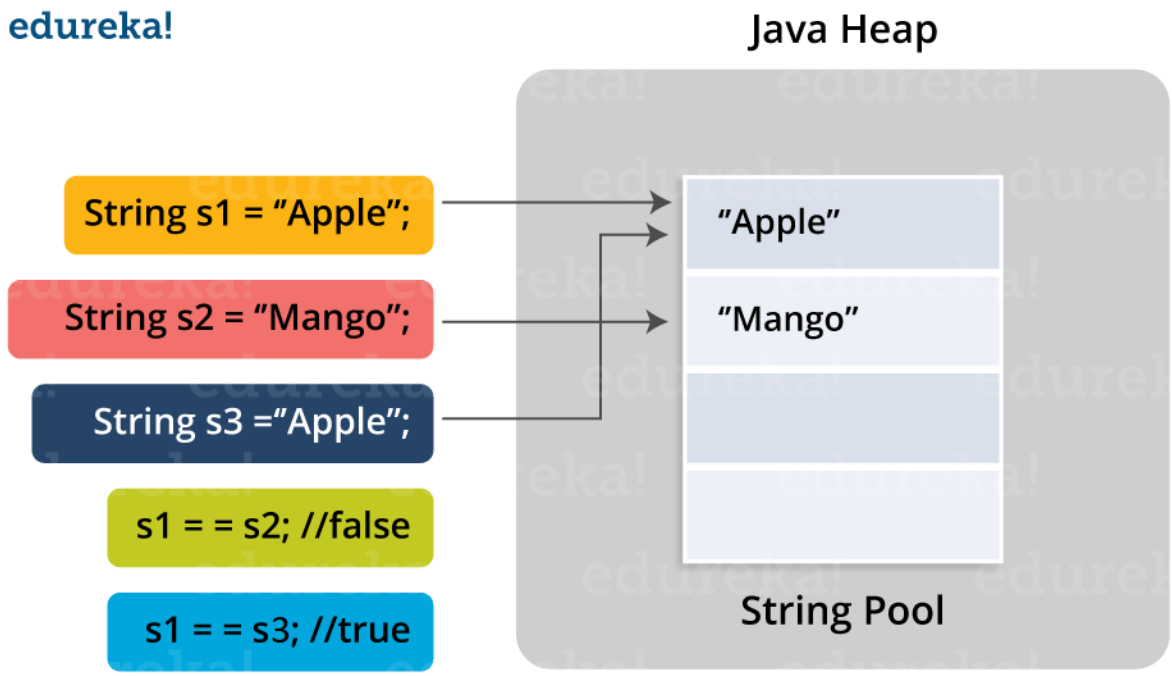
Q35. What is the difference between this() and super() in Java?

In Java, super() and this(), both are special keywords that are used to call the constructor.

this()	super()
1. this() represents the current instance of a class	1. super() represents the current instance of a parent/base class
2. Used to call the default constructor of the same class	2. Used to call the default constructor of the parent/base class
3. Used to access methods of the current class	3. Used to access methods of the base class
4. Used for pointing the current class instance	4. Used for pointing the superclass instance
5. Must be the first line of a block	5. Must be the first line of a block

Q36. What is Java String Pool?

Java String pool refers to a collection of Strings which are stored in heap memory. In this, whenever a new object is created, String pool first checks whether the object is already present in the pool or not. If it is present, then the same reference is returned to the variable else new object will be created in the String pool and the respective reference will be returned.



Q37. Differentiate between static and non-static methods in Java.

Static Method	Non-Static Method
1. *The static* keyword must be used before the method name	1. No need to use the *static* keyword before the method name
2. It is called using the class (className.methodName)	2. It is can be called like any general method
3. They can't access any non-static instance variables or methods	3. It can access any static method and any static variable without creating an instance of the class

Q38. Explain the term "Double Brace Initialisation" in Java?

Double Brace Initialization is a Java term that refers to the combination of two independent processes. There are two braces used in this. The first brace creates an anonymous inner class. The second brace is an initialization block. When these both are used together, it is known as Double Brace Initialisation. The inner class has a reference to the enclosing outer class, genertally using the 'this' pointer. It is used to do both creation and initialization in a single statement. It is generally used to initialize collections. It reduces the code and also makes it more readable.

Q39. What is constructor chaining in Java?

In Java, constructor chaining is the process of calling one constructor from another with respect to the current object. Constructor chaining is possible only through legacy where a subclass

constructor is responsible for invoking the superclass' constructor first. There could be any number of classes in the constructor chain. Constructor chaining can be achieved in two ways:

1. Within the same class using this()
2. From base class using super()

Q40. Difference between String, StringBuilder, and StringBuffer.

Factor	String	StringBuilder	StringBuffer
Storage Area	Constant String Pool	Heap Area	Heap Area
Mutability	Immutable	Mutable	Mutable
Thread Safety	Yes	No	Yes
Performance	Fast	More efficient	Less efficient

If you think this article on Java Interview Questions is helpful, you can check out Edureka's Java Training in Chennai as well.

Q41. What is a classloader in Java?

The **Java ClassLoader** is a subset of JVM (Java Virtual Machine) that is responsible for loading the class files. Whenever a Java program is executed it is first loaded by the classloader. Java provides three built-in classloaders:

1. Bootstrap ClassLoader
2. Extension ClassLoader
3. System/Application ClassLoader

Q42. Why Java Strings are immutable in nature?

In Java, string objects are immutable in nature which simply means once the String object is created its state cannot be modified. Whenever you try to update the value of that object instead of updating the values of that particular object, Java creates a new string object. Java String objects are immutable as String objects are generally cached in the String pool. Since String literals are usually shared between multiple clients, action from one client might affect the rest. It enhances security, caching, synchronization, and performance of the application.

Q43. What is the difference between an array and an array list?

Array	ArrayList
Cannot contain values of different data types	Can contain values of different data types.
Size must be defined at the time of declaration	Size can be dynamically changed

Need to specify the index in order to add data	No need to specify the index
Arrays are not type parameterized	Arraylists are type
Arrays can contain primitive data types as well as objects	Arraylists can contain only objects, no primitive data types are allowed

Q44. What is a Map in Java?

In Java, Map is an interface of Util package which maps unique keys to values. The Map interface is not a subset of the main Collection interface and thus it behaves little different from the other collection types. Below are a few of the characteristics of Map interface:

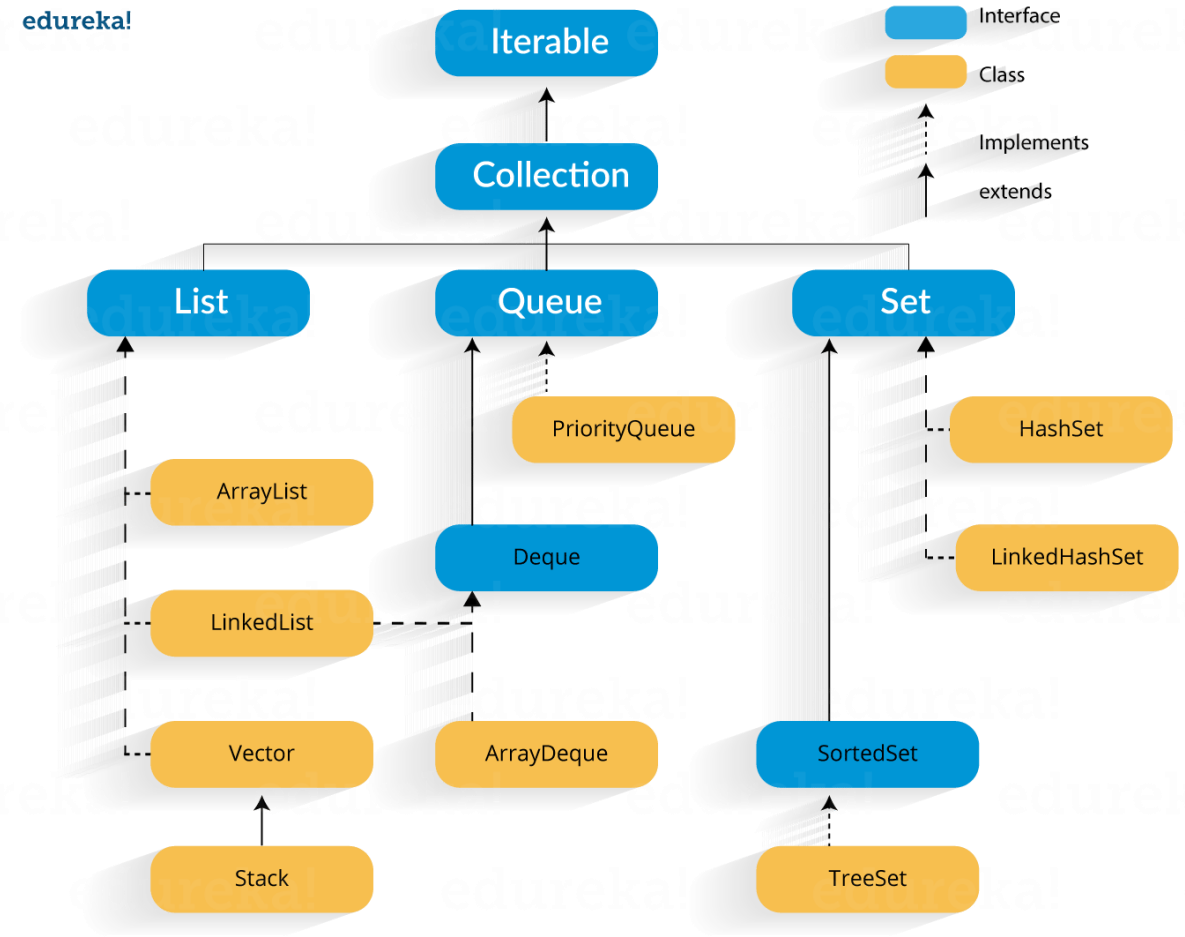
1. Map doesn't contain duplicate keys.
2. Each key can map at max one value.

Q45. What is collection class in Java? List down its methods and interfaces.

In Java, the collection is a framework that acts as an architecture for storing and manipulating a group of objects. Using Collections you can perform various tasks like searching, sorting, insertion, manipulation, deletion, etc. Java collection framework includes the following:

- Interfaces
- Classes
- Methods

The below image shows the complete hierarchy of the Java Collection.



Want to upskill yourself to get ahead in Career? Check out this video

Core Java Interview Quesions And Answers for Freshers and Experienced

This 120+ core java interview question and answer will help freshers and experience to crack the interview in 1st attempt