Q1. What is JVM ?

- JVM, i.e., Java Virtual Machine.
- JVM is the engine that drives the Java code.
- Mostly in other Programming Languages, compiler produce code for a particular system but Java compiler produce Bytecode for a Java Virtual Machine.
- When we compile a Java program, then bytecode is generated. Bytecode is the source code that can be used to run on any platform.
- Bytecode is an intermediary language between Java source and the host system.
- It is the medium which compiles Java code to bytecode which gets interpreted on a different machine and hence it makes it Platform/Operating system independent.

Why is Java called the Platform Independent Programming Language?

- Java is called platform independent because of Java Virtual Machine. As different computers with the different operating system have their JVM, when we submit a **.class** file to any operating system, JVM interprets the bytecode into machine level language.

Q2. What does the "static" keyword mean ?

The **static keyword** in Java is mainly used for memory management. The static keyword in Java is used to share the same variable or method of a given class. The users can apply static keywords with variables, methods, blocks, and nested classes. The static keyword belongs to the class than an instance of the class. The static keyword is used for a constant variable or a method that is the same for every instance of a class.

Can you override private or static method in Java ?

No, we cannot override private or static methods in Java.

Private methods in Java are not visible to any other class which limits their scope to the class in which they are declared.

Q3. What is a Constructor

A constructor in Java is a **special method** that is used to initialize objects. The constructor is called when an object of a class is created. It can be used to set initial values for object attributes.
In Java, a constructor is a block of codes similar to the method. It is called when an instance of the class is created

Constructor Overloading and Copy-Constructor in java ?

Overloaded constructor is called based upon the parameters specified when new is executed. Sometimes there is a need of initializing an object in different ways. This can be done using constructor overloading.

In Java, a ***copy constructor*** is a special type of constructor that creates an object using another object of the same Java class. It returns a duplicate copy of an existing object of the class.

Q4. What is the difference between an Interface and an Abstract class ?

Abstract class	Interface
1) Abstract class can **have abstract and non-abstract** methods.	Interface can have **only abstract** methods. Since Java 8, it can have **default and static methods** also.
2) Abstract class **doesn't support multiple inheritance**.	Interface **supports multiple inheritance**.
3) Abstract class **can have final, non-final, static and non-static variables**.	Interface has **only static and final variables**.
4) Abstract class **can provide the implementation of interface**.	Interface **can't provide the implementation of abstract class**.
5) The **abstract keyword** is used to declare abstract class.	The **interface keyword** is used to declare interface.
6) An **abstract class** can extend another Java class and implement multiple Java interfaces.	An **interface** can extend another Java interface only.

7) An **abstract class** can be extended using keyword "extends".	An **interface** can be implemented using keyword "implements".
8) A Java **abstract class** can have class members like private, protected, etc.	Members of a Java interface are public by default.

9)**Example:**
public abstract class Shape{
public abstract void draw();
}
	Example:
	public interface Drawable{
	void draw();
	}

Q5. What is the difference between processes and threads ?

Process	Thread
A process is an instance of a program that is being executed or processed.	Thread is a segment of a process or a lightweight process that is managed by the scheduler independently.
Processes are independent of each other and hence don't share a memory or other resources.	Threads are interdependent and share memory.
Each process is treated as a new process by the operating system.	The operating system takes all the user-level threads as a single process.
If one process gets blocked by the operating system, then the other process can continue the execution.	If any user-level thread gets blocked, all of its peer threads also get blocked because OS takes all of them as a single process.
Context switching between two processes takes much time as they are heavy compared to thread.	Context switching between the threads is fast because they are very lightweight.
The data segment and code segment of each process are independent of the other.	Threads share data segment and code segment with their peer threads; hence are the same for other threads also.
The operating system takes more time to terminate a process.	Threads can be terminated in very little time.

New process creation is more time taking as each new process takes all the resources.	A thread needs less time for creation.

Q6. What is an Iterator ?

An `Iterator` is an object that can be used to loop through collections, like ArrayList and HashSet. It is called an "iterator" because "iterating" is the technical term for looping.

What differences exist between Iterator and ListIterator ?

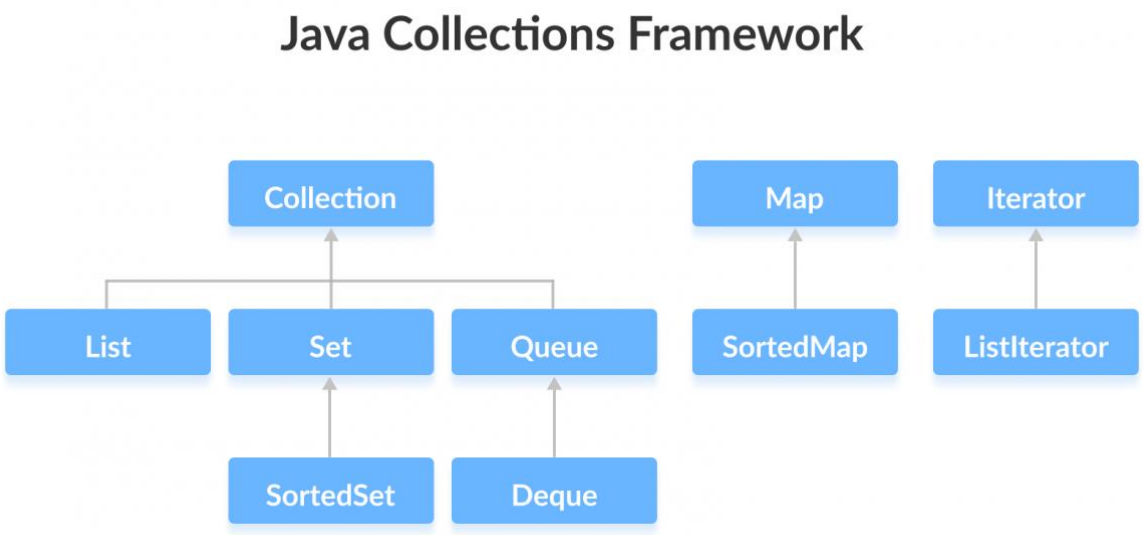
BASIS FOR COMPARISON	ITERATOR	LISTITERATOR
Basic	Iterator can traverse the elements in a collection only in forward direction.	ListIterator can traverse the elements in a collection in forward as well as the backwards direction.
Add	Iterator is unable to add elements to a collection.	ListIteror can add elements to a collection.
Modify	Iterator can not modify the elements in a collection.	ListIterator can modify the elements in a collection using set().

BASIS FOR COMPARISON	ITERATOR	LISTITERATOR
Traverse	Iterator can traverse Map, List and Set.	ListIterator can traverse List objects only.
Index	Iterator has no method to obtain an index of the element in a collection.	Using ListIterator, you can obtain an index of the element in a collection.

Q7. Why Collection doesn't extend Cloneable and Serializable interfaces ?

The Collection interface in Java specifies a group of objects called elements. **The maintainability and ordering of elements is completely dependent on the concrete implementations provided by each of the Collection**. Thus, there is no use of extending the Cloneable and Serializable interfaces.

Q8. What are the basic interfaces of Java Collections Framework ?



Q9. What is the importance of hashCode() and equals() methods ?

Equals () and Hashcode () in Java The equals () and hashcode () are the two important methods provided by the Object class for **comparing objects**. Since the Object class is the parent class for all Java objects, hence all objects inherit the default implementation of these two methods.

Q10. What is difference between Array and ArrayList ?

Base	Array	ArrayList
Dimensionality	It can be single-dimensional or multidimensional	It can only be single-dimensional
Traversing Elements	For and for each generally is used for iterating over arrays	Here iterator is used to traverse riverArrayList
Length	length keyword can give the total size of the array.	size() method is used to compute the size of ArrayList.

Base	Array	ArrayList
Size	It is static and of fixed length	It is dynamic and can be increased or decreased in size when required.
Speed	It is faster as above we see it of fixed size	It is relatively slower because of its dynamic nature
Primitive Datatype Storage	Primitive data types can be stored directly unlikely objects	Primitive data types are not directly added unlikely arrays, they are added indirectly with help of autoboxing and unboxing
Generics	They can not be added here hence type unsafe	They can be added here hence makingArrayList type-safe.
Adding Elements	Assignment operator only serves the purpose	Here a special method is used known as add() method

When will you use Array over ArrayList ?

Q11. What is difference between ArrayList and LinkedList ?

Parameter of Comparison	ArrayList	LinkedList
Usage	A dynamic array is used to store elements internally.	A double-linked list is used to store elements internally.
Manipulation	Manipulation is slow and takes more time.	Manipulation is faster and takes the least time.
Implementation	ArrayList implements only List.	LinkedList implements List and Queue.
Access	ArrayList is better when an application wants to store and access data.	LinkedList works faster in the manipulation of the stored data.
Performance	ArrayList performs 0(1).	LinkedList performs 0(n).

Q12. What is Comparable and Comparator interface ? List their differences.

BASIS FOR COMPARISON	COMPARABLE	COMPARATOR
Basic	The Comparable interface allows only single sorting sequence.	The Comparator interface allows multiple Sorting sequences.
Packages	The Comparable interface is present in the java.lang package.	The Comparator interface is present in the java.util package.
Methods	The Comparable interface contains only single method public int compareTo(Object obj);	The Comparator interface contains two methods public int compare(Object obj1, Object obj2) boolean equals(Object obj)
Implementation	Comparable interface is implemented by the class whose objects are to be compared.	Comparator interface is implemented by a sperate class instead to the class whose objects are to be compared.

BASIS FOR COMPARISON	COMPARABLE	COMPARATOR
Comparison	The compareTo(Object obj) method compares the object which is used to invoke the method with the specified object passes to the method.	The compare(Object obj1, Object obj2) method compare the both the specified objects that are passed to the method.
List/Array	When a list of the object of Comparable type has to be compared the Collection class provides a method i.e. Collections.sort(List lst).	When a list of objects of Comparable type has to be compared the Collection class provides a method i.e. Collections.sort(List, Comparator).

Q13. What is the purpose of garbage collection in Java, and when is it used ?

The **purpose of garbage collection** is to identify and discard those objects that are no longer needed by the application, in order for the resources to be reclaimed and reused.

The **gc**() **method is used** to invoke the **garbage collector** to perform cleanup processing. The **gc**() is found in System and Runtime classes.

Q14. What are the two types of Exceptions in Java ? Which are the differences between them ?

Checked Exception

Checked exceptions are called **compile-time** exceptions because these exceptions are checked at compile-time by the compiler.

Unchecked Exceptions

The **unchecked** exceptions are just opposite to the **checked** exceptions. The compiler will not check these exceptions at compile time.

Checked Exception	Unchecked Exception
Checked exceptions occur at compile time.	Unchecked exceptions occur at runtime.
The compiler checks a checked exception.	The compiler does not check these types of exceptions.
These types of exceptions can be handled at the time of compilation.	These types of exceptions cannot be a catch or handle at the time of compilation, because they get generated by the mistakes in the program.
They are the sub-class of the exception class.	They are runtime exceptions and hence are not a part of the Exception class.
Here, the JVM needs the exception to catch and handle.	Here, the JVM does not require the exception to catch and handle.
Examples of Checked exceptions:	Examples of Unchecked Exceptions:
• File Not Found Exception • No Such Field Exception • Interrupted Exception • No Such Method Exception • Class Not Found Exception	• No Such Element Exception • Undeclared Throwable Exception • Empty Stack Exception • Arithmetic Exception • Null Pointer Exception • Array Index Out of Bounds Exception • Security Exception

Q15. what is final finally and finalize in java?

- Final: The Final is a Keyword that allows declaring a class, data members, and method to be unalterable.
- Finally: Finally is a code block in Exception Handling in Java. ...
- Finalize: Finalize is the method in Java that is used to perform memory clean-up before the object got collected by JVM garbage collection.

Q16. what is exception handling?

Exception Handling is a mechanism to handle runtime errors such as ClassNotFoundException, IOException, SQLException, RemoteException, etc.

explain error and exception.

Error refers to an illegal operation performed by the user which results in the abnormal working of the program.

exceptions in java refer to 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.

Explain difference between throw and throws

Sr. no.	Basis of Differences	throw	Throws
1.	Definition	Java throw keyword is used throw an exception explicitly in the code, inside the function or the block of code.	Java throws keyword is used in the method signature to declare an exception which might be thrown by the function while the execution of the code.
2.	Type of exception Using throw keyword, we can only propagate unchecked exception i.e., the checked exception cannot be propagated using throw only.	Using throws keyword, we can declare both checked and unchecked exceptions. However, the throws keyword can be used to propagate checked exceptions only.	
3.	Syntax	The throw keyword is followed by an instance of Exception to be thrown.	The throws keyword is followed by class names of Exceptions to be thrown.
4.	Declaration	throw is used within the method.	throws is used with the method signature.
5.	Internal implementation	We are allowed to throw only one exception at a time i.e. we cannot throw multiple exceptions.	We can declare multiple exceptions using throws keyword that can be thrown by the method. For example, main() throws IOException, SQLException.

Q16. What is the difference between Exception and Error in java ?

Basis of Comparison	Exception	Error
Recoverable/ Irrecoverable	Exception can be recovered by using the try-catch block. An error cannot be recovered.	
Type	It can be classified into two categories i.e. checked and unchecked.	All errors in Java are unchecked.
Occurrence	It occurs at compile time or run time.	It occurs at run time.
Package	It belongs to java.lang.Exception package.	It belongs to java.lang.Error package.
Known or unknown	Only checked exceptions are known to the compiler.	Errors will not be known to the compiler.
Causes	It is mainly caused by the application itself.	It is mostly caused by the environment in which the application is running.

Example	**Checked Exceptions:** SQLException, IOException **Unchecked Exceptions:** ArrayIndexOutOfBoundException, NullPointerException, ArithmaticException	Java.lang.StackOverFlow, java.lang.OutOfMemoryError