1. What do you know about JVM, JRE and JDK?

→ JDK :-

JDK is an abbreviation for Java Development Kit. It is an environment of software development used for developing applets and Java applications. JDK has a physical existence, and it contains JRE + development tools. JDK assists them in coding and running the Java Programs. JDK consists of a private JVM along with a few other resources, java (a loader/interpreter) like javac (a compiler), Javadoc, jar (an archiver).

JRE :-

JRE stands for Java Runtime Environment - also written as Java RTE. It is a set of software tools designed for running other software. It is an implementation of JVM, and JRE provides a runtime environment. User needs JRE to run any Java program.

JVM :-

JVM stands for Java Virtual Machine. It provides a runtime environment for driving Java application or code. JVM is an abstract machine that converts the Java bytecode into a machine language (compiled to Java bytecode). JVM is also known as virtual machine as it does not exist physically.

All three, JDK, JRE and JVM are dependent.
JVM has 3 notions:- implementation
instance
Specifications.

2. Is JRE platform dependant or independent?

→

Java as a programming language is platform-independent.
Java bytecode generated from Java source code is platform-independent.
The JRE itself (the runtime environment that executes Java bytecode) is platform-dependent as you need different versions of the JRE for different platforms.
So, in pratice, Java applications can be platform-independent, but the JRE is platform-specific.

3. What is ultimate base class in java class hierarchy? List the name of methods of it?

→ In the java class hierarchy, the ultimate base class is the `java.lang.Object` class. All classes in Java, whether they are explicitly defined by the programmer or implicitly through inheritance, ultimately inherit from the `Object` class. The `Object` class is located in the `java.lang` package & it contains several important methods, including:-

- `toString()`
- `equals(Object obj)`
- `hashCode()`
- `notify()`
- `notifyAll()`
- `wait()`
- `wait(long timeout)`

4. Which are the references type in java?

→ Reference datatypes in java are those which contains reference (address of dynamically created objects. These are not predefined like primitive data types. following are the reference types in Java.

class types:-

This reference types points to an object of a class.

Array types:-

This reference types points to an array.

interface types:-

This reference type points to an object of a class with implements an interface.

5. Explain narrowing and widening?

→ Narrowing:-

Also known as downcasting (casting) is a conversion that is explicitly performed in the following situations -

- Narrowing a wider (bigger) primitive type value to a smaller primitive type value.
- Narrowing a superclass reference to a subclass reference, during inheritance.

Widening:-

Also known as upcasting is a conversion that implicitly takes place in the following situations -

- Widening takes place when a smaller primitive type value is automatically accomodated in a larger (wider) primitive data type.
- Widening also a superclass reference to a subclass reference, during inheritance.

6. How will you print "Hello CDAC" statement on screen, without semicolon?

→
```
Java
class HelloWorld {
    public static void (String []args){
        if (System.out.printf("Hello CDAC\n")==null){
        }
    }
}
```

We use 'System.out.printf("Hello CDAC\n")' to print the desired text. The '\n' character is used for a line break.
We enclose the 'System.out.printf' statement within an 'if' block.
The 'if' condition compares the results of 'System.out.printf' to 'null'. Since System.out.printf returns a 'PrintStream' (which is not 'null'), then cond" will always be false, and the code block inside the 'if' statement will not be executed.

7. Can you write java application without main function? If yes, how?

→ Yes, we can execute a java program without a main method by using a static block. Static Block in Java is a group of statements that gets executed only once when the class is loaded into memory by Java ClassLoader. It is also known as a Static initialization block. Static initialization block is going directly into the stack memory.

8. What will happen, if we call main method in static block?

→ If you call the 'main' method from within a static block (a 'static' initializer block) in a Java class, it will run the 'main' method like any other method call. It can lead to an unexpected behaviour. First the static block gets print and then calls the 'main' method with an empty array of strings.

9. In System.out.println, Explain meaning of every word?

→ Java System.out.println() is used to print an argument that is passed to it. The statement can be broken into 3 parts which can be understood seperately as:-

System:- It is a final class defined in the java.lang package.

out:- This is an instance of PrintStream type, which is a public and static member field of the System class.

println:- As all instances of PrintStream class have a public method println().

10. How will you pass object to the function by reference?

→ Although Java is strictly passed by value, the precise effect differs betⁿ whether a primitive type or a reference type is passed. When we pass a primitive type to a method, it is passed by value. Objects are passed by what is effectively call-by-reference

11. Explain constructor chaining? How can we achieve it in c++?

→ Constructor chaining is a concept in object-oriented programming where one constructor of a class can call another constructor within the same class. This allows you to reuse the initialization logic of one constructor in another, reducing code duplication and ensuring consistent initialization of objects. Constructor chaining is supported in both Java and c++.

In c++, constructor chaining can be achieved using member initializer lists and multiple constructors in a class.

12. Which are the rules to overload method in subclass?

→ Method overloading allows you to define multiple in a class with the same name but with different parameter lists. When you override a method in a subclass there are specific rules you should follow to ensure correct method overloading.

Rules to overload methods in a subclass:-

1) Method Name Must Be Same
2) Parameter List must Differ
3) Return Type Can Be Different
4) Access Modifiers
5) Expectations.

18. Explain the difference among finalize and dispose?

→

Features	Dispose Method()	Finalize Method()
Defined	It is defined in the interface IDisposable interface.	It is defined in java.lang.object class.
Basic	It is used to close or release unmanaged resources stored by an object, like files or streams.	It is used to clear up unmanaged resources owned by the current obj. before it is destroyed.

Syntax	The syntax of dispose() method is:
	public void Dispose()
	{
	// Dispose code here
	}
		The syntax of finalize() method is:
		protected void finalize()
		{
		// Code for finalization
		}

Access Specifiers	It is declared as public.	It is declared as private
Invoked	By user Very Quickly	By garbage collector(GC) Very slowly.
Performance	Executes immediate action and has no impact on performance.	It has an impact on the performance of the site.

14. Explain the difference among final, finally & finalize ?

→

Key	final	finally	finalize
Definition	final is the keyword and access modifier which is used to apply restrictions on a class, method	finally is the block in Java Exception to execute the important code whether the exception	finalize is the method in Java is used to perform clean up processing just before object.
Applicable to	final keyword is used with the classes, methods and variables	finally block is always related to the try & catch	finalize method is used with the objects.
Functionality	1] finale variable becomes constant and cannot be modified. 2] final method cannot be overridden by subclass	1] finally block runs the important code even if exception 2] finally block cleans up all the resources used in try block.	1] finalize method performs the cleaning activities with respect to the object before its destruction.
Execution	final method is executed only when we call it.	finally block is executed as soon as the try-catch	finalize method is executed just before the obj.

15. Explain the diff^n among checked & unchecked exception?

→

Checked Exception	Unchecked Exception
1) checked expectations happen at compile time when the source code is transformed into an executable code.	Unchecked expectations happen at runtime when the executable program starts running.
2) They checked exception is checked by compiler.	These types of exceptions are not checked by compiler.
3) Checked exception can be created manually.	They can also be created manually.
4) This exception is counted as a sub-class of class.	This exception happens in runtime & hence exception class.
5) JVM requires the exception to be caught or handled.	JVM does not need the exception to be caught.

16. Explain exception chaining?

→ In Java, a chained exception is a technique that enables programmers to associate one Exception with another. By providing additional information about an specific exception, debugging can be made easier. A chained exception is created by wrapping an existing in new exception, which becomes the root cause of new Exception.

Various Constructors:-

Throwable (Throwable cause)

Throwable (String desc, Throwable cause)

getCause()

initCause() method

17. Explain the diffn among throw & throws?

→

Key Difference	throw	throws
Point of Usage	The throw keyword is used inside a function. It is used when it is required to throw an Exception logically.	The throws keyword is used in function signature. It is used when the function has some statements that can lead to exceptions.
Exceptions Thrown	The throw keyword is used to throw an exception explicitly. It can throw only exception at a time.	The throws keyword can be used to declare multiple exception seperated by comma.
Syntax	Syntax of throw keyword includes the instances of the Exception to be thrown.	Syntax of throws keyword includes the class names of the exceptions to be thrown.
Propogation of Exceptions	throw keyword cannot propogate checked exceptions. It is only used to propogate the unchecked Exceptions.	throws keyword is used to propogate the checked Exceptations only.

18. In which case, finally block doesn't execute?

→ 'finally' block associated with a try-catch-finally construct is designed to execute under most circumstances.

few Scenarios:-

1) System.exit()
2) Infinite Loop or Hang

19. Explain Upcasting?

→ Upcasting is a type of object typecasting in which a child object is typecasted to a parent class object. By using Upcasting, we can easily access the variables and methods of the parent class to child class.

Upcasting is also known as Generalization and Widening.

20. Explain dynamic method dispatch?

→ Dynamic method dispatch is a mechanism in object-oriented programming languages - such as Java that enables the selection of a method to be executed at runtime rather than compile time.

Dynamic method dispatch is a fundamental concept in achieving polymorphism in object-oriented programming.

Key Points:-

Polymorphism

Inheritance and Overridding

Base & Derived Classes

21. What do you know about final method?

→

When a method is declared as final, it cannot be overridden by a subclass.

This is useful for methods that are part of a class's public API and should not be modified by subclasses.

22. Explain fragile base class problem and how can we overcome it?

→

Fragile Base Class problem is a software design issue that occurs in object-oriented programming when a class is widely used as a base class for other classes, and changes made to base class can unintentionally break or introduce bugs in the derived classes. This problem arises when subclasses rely on the behaviour and implementation details of the base class, and any changes to the base class can have unintended consequences.

To overcome the fragile base class problem, you can follow these design principles:-

- Minimize Coupling
- Use Abstraction
- Liskov Substitution Principle (LSP)
- Design by Contract.
- Interfaces and Abstract classes.
- Testing & Regression Testing
- Documentation.

28. Why java does not support multiple implementation inheritance?

→ In Java, a class cannot extends more than one class. Therefore following is illegal -

Ex. public class extends Animal, Mammal {}

The reason behind this is to prevent ambiguity. In this java compiler cannot decide, which display method it should inherit. To prevent such situation, multiple inh. is not allowed in java.

24. Explain maker interface? List the name of some maker interfaces?

→ An iterface that does not contain methods, fields and constants is known as marker interface.
An empty interface is known as marker interface or tag interface.
It delivers the run-time type information about an object.
It is the reason that the JVM and compiler have additional information about an object.
Ex :-
Serializable
Cloneable
It signals or command to the JVM.
Two alternatives of maker interface:-
Internal flags
Annotations
Built-in Maker Interfaces:- Already present in JDK.
Clonable Interface
Serializable Interface
Remote Interface

25. Explain the significance of maker interface?

→ The main use of Maker Interface in Java is to convey to the JVM that the class implementing some interface on this category has to be granted some special behaviour.
When a class implements the serializable interface, which is a marker interface, then this is an addiction to the JVM that the objects of this class can be serialized. Similarly objects of this class can be cloned.