1 what do you know about JVM, JRE & JDK ?

JVM

Java virtual machine is an abstract machine that execute Java Bytecode

Their are different JVM for different hardware & software problems

JRE

Java Runtime Environment. This is included in JDK. JRE provider libraries & JVM that is required to run java program

JDK

Java Development kit. It contains the tools & libraries for development of Java programs. It also contains compiler & debugger required to run java program.

2 Is JRE dependent or Independent ?

JRE is dependent

It is dependent because in JVM, JRE & JDK the configuration of each os is different from each other

JRE is platform dependent It has the responsibility of creating an environment for the execution of codes. It provides the system with minimum requirement needed for execution of Java Program in a system. Since the configurations for different OS varies, therefore JRE is platform dependent

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

Ans Object class is present in java.lang package Every class in Java is directly or indirectly derived from the Object class. If a class does not extend any other class then it is a direct child class of Object & if extends another class then it is indirectly derived therefore object class methods are available to all Java classes. Hence object class acts as a root of the inheritance hierarchy in any Java program

The object class provides following methods:
toString() method
hashCode() method
equals(Object obj) method
finalize() method

getClass() method
done() method
wait(), notify(), notifyAll() methods

Q What are the reference types in Java?

Sol" The reference types in Java are:-
① Class ② Interface ③ Array Reference ④ String

Unlike primitive types, which are stored directly in memory reference types are stored as reference to other object. It means that a referenced type can only be used to access an object if the object exists.

Classes are blueprints for creating objects. They define the properties & methods of an object. Interface are similar to classes but they do not define any properties or method.

Interface are used to specify a contract that objects must meet

Arrays are used to store a collection of objects. They can be created from any type of object including primitive types & reference types.

Strings are immutable reference type that represent sequence of characters Strings are used to represent text such as the name of a person or the title of a book

5 Explain narrowing & widening

Soln Primitive data types can be converted into one another implicitly or explicitly. This process is called as Type casting
There are two types of type casting:-
① widening ② narrowing

Widening Typecasting

- converting lower data type into higher one
- is called it is implicit conversion
- It is safe because no data loss

```
int num1 = 10;
float num2 = 15.5f;

long num3 = num1 //ok

double num4 = num2 //ok
```

Narrowing Typecasting

- Converting higher data type into a lower data type
- It is explicit conversion
- There is some data loss

```
int a = 1234
// byte b = a // error : lossy conversion
byte b = (byte) a; // ok
```

6 Print "Hello CDAC" statement on screen without semicolon ;.

Sol: 1st way: using if statement

```
if (System.out.println ("Hello CDAC") == null);
```

Output :- Hello CDAC

2nd way: using append() method of String Builder class

```
if (System.out.append ("Hello CDAC") == null)
{ }
```

Output :- Hello CDAC

3rd way :- using equals method of string class

```
if (System.out.append("Hello CDAC").equals(null))
{
}
```

Output :- Hello CDAC

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

Sol^n Yes, by using a static block.
Because static block are executed at the time of class loading i.e before execution of main

```
class Program {
    static {
        System.out.println("Hello");
        System.exit(0);
    }
}
```

Output :- Hello

9. In System.out.println explain meaning of every word.

Sol^n In Java, System.out.println() is a statement that prints the arguments passed to it. It prints a message to standard output (console) & appends a newline character.

- System is a built-in final class in java.lang package. It cannot be instantiated & provide access to standard input, output & error streams.

- out is an instance of system class & represents the standard output stream. It is of type printstream & is responsible for writing data to output stream.

- println() method is an overloaded method of printstream class. It is used when we want to print our result in two separate lines. It throws the cursor ~~at the~~ to the next line after displaying the result.

10 How will you pass object to function by references ?

ans In Java, objects always passed by reference to the object reference but it is important to understand that Java is bit different from some other languages like C++.

Example

```
class myclass {
    int value;

    myclass (int value) {
        this.value = value;
    }
}

public class PassObjectByReference {
    public static void modifyObject (myclass obj) {
        obj.value = 42;
    }

    public static void main (String args[]) {
        myclass myobject = new myclass (10);
        System.out.println (object.value);

        modify Object (my object);

        System.out.println (myobject.value);
    }
```

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

Ans Constructor chaining in Java is concept wherein one constructor can call other constructor in the same class or its superclass.

Within the same class (using `this()`); in this scenario one constructor of the class can call constructor

Example in cpp :-

```
class myclass {
public:
    myclass(int value) : data(value) {
    }
    myclass() : myclass(0) {
    }
private:
    int data;
};
```

12 What are the rules to overload method in subclass ?

Soln ① Method Name :- The overloaded method in

subclass must have to must same name of method in the superclass that you want to overload

② Parameter list :- The parameter list of the overloaded method in the subclass must be different from the parameter list of method in the superclass

③ Return type - The return type of the overloaded in subclass can be same as or a subtype of the return type method in the superclass

13 Explain difference between finalize & dispose

Dispose()	Finalize()
① It is defined interface disposable interface	It is defined in Java.lang.Object class
② The syntax of dispose() method is public void dispose() { }	The syntax of finalize() is protected void finalize()
③ It is declared as public	It is declared by private

④ It is invoked by user — It is invoked by garbage collector

⑤ It is invoked very quickly — It is invoked more slowly than dispose() method.

Q4: Explain difference among final, finally & finalize?

Ans final → final is the keyword & access modifier which is used to apply restriction on class

- finally keyword is used with the classes, methods & variables

- final method is executed only when we call it

finally - finally is the block Java exception handling to execute the important code

- finally block is always related to the try & catch block in exception handling

- Its execution is not dependent on the execution

finalize - finalize method in Java which is use to perform clean up processing

- finalize method is used with the object

- finalize method is executed just before the object destroyed

Ques15 Explain the difference between checked & unchecked exception.

Ans

Checked Exceptions in Java

These are the exceptions that are checked at compile time. If some code within a method throws a checked exception, then the method must either handle the exception or it must specify the exception using the throws keyword

Unchecked Exceptions in Java

These are the exceptions that are not checked at compile time. In C++, all exceptions are unchecked, so it is not forced by the compiler's to either handle or specify the exception. It is up to the programmers to be civilized & specify or catch the exceptions

Checked Exception	Unchecked Exception
• Checked by the compiler	Not checked by the compiler
~~It can be created manually~~	
• JVM requires the exception to be caught or handled.	JVM does not need the exception to be caught or handled
• It is the sub-class of the exception class.	They are runtime exceptions & hence are not a part of the Exception class
• Occurs at compile time	Occurs at runtime
~~Examples :- NullPointerException etc~~	
• Example :- SocketException, SQLException, IOException, FileNotFoundException	Example :- NullPointerException, ClassCastException, ArithmeticException, ArrayStoreException,

Ques 16 Explain Exception chaining. 9

Ans Exception chaining occurs when one exception causes another exception. The original exception is the cause of the second exception

Often we need to throw a custom exception & want to keep the details of an original exception that is such scenarios we can use the chained exception mechanism

Advantage:-
It can help debug, as it can help us track down the root cause of an error

Disadvantage:-
. chaining can make our code more difficult to read & understand. Therefore, we should use exception chaining sparingly & only when necessary

17 Difference between throw & throws

Ans Throw

- Throw keyword is used inside a function
- It is used when it required to throw exception logically
- Throw keyword is used to throw exception explicitly
- It can throw only one exception at a time
- Syntax of throw keyword include the instance of the exception to be thrown

Throws

Throws keyword is used in function signature
It is used when the function has some statement that can lead to exception

The throws keyword can be used to declare multiple exception seperated by commas, whichever exception

Syntax of throws keyword include the class name of the Exception throws

18 In which case finally block doesn't execute

Any the finally block in try catch finally execute is typically execute under most circumstances.

(1) system.exit() - In your program calls system.exit(int status) the JVM terminate immediately & no further code, including the finally block is executed

(2) Infinite loop - If your program enters infinite loop or encounter a condition where it never leaves a particular code block the finally block will not execute until that loop or condition is interrupted or resolved so such cases, the program may appear to be stuck

(3) Uncaught Error - If your program encounters an uncaught error, such as a hardware failure or critical runtime error it may terminate abruptly without executing the finally block

19 What is upcasting?

ans Upcasting is a type of casting in object oriented programming where a reference to a subclass object is treated as a reference to superclass

This means you are moving up the class hierarchy from a more specialized type / subclass to a more general type superclass

20 Explain dynamic method dispatch

ans Dynamic method dispatch is the mechanism which call to an overridden method is resolved at runtime

This is an important concept because of how Java implements runtime polymorphism

Java uses the principle of a superclass reference variable can refer to a subclass object to resolve call overridden method at runtime

2| What do you know about final method?

ans It is a keyword which is used to restrict the user.
If we make any method as final, we cannot override it.

When to use?
When we want that no one can change the definition of final method in derived classes

Example:
Let's take an example of sensitive data of the office IN the office, there may be some data that are not accessible to each employee

Advantages
- Improving performance
- making code easier to understand
- Enhancing security
- Promoting code reuse

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

ans The improper design of parent class can lead sub classes of a super class to use the super class in unexpected ways.

Example :-

```
class Rectangle {
    private int len;
    private int breadth;
    public Rectangle(int len, int breadth) {
        this.len = len;
        this.breadth = breadth;
    }
    public int calculateArea() {
        return len * breadth;
    }
    // setters & getters
}
class square extends Rectangle {
    public square(int side) {
        super(side, side);
    }
@Test
public void testSquare() {
    Square sq = new Square(5);
    assertEquals("Area of square", 25, square.
                                    CalculateArea());
```

```
square.setBreadth(9);
assertEquals("Area of square", sq.square.
                                calculateArea());
}
```

output:

Expected : 81
Actual : 45

Soln

There is no straight forward solution to this problem because this is all about following best practices

→ A/C to Joshua Bloch,
programmer should "design & document for inheritance or else prohibit it"

→ If there is a breakable superclass, it is better to prohibit inheritance by labelling a declaration of class or method, with keyword "final"

→ use either constructor or setter but not both

23 Why Java does not support multiple implementation inheritance.

ans Java does not support multiple implementation inheritance for several reasons, primarily related to the complexity & practical issue

① Diamond problem —
multiple inheritance can lead to a problem known as diamond problem. This occurs when a class inherited from two classes that have common ancestor

② Complexity
Multiple inheritance introduced complexity in the term of method resolution object initialization & memory corrupt

③ Fragile Base class problem
Multiple inheritance can lead to the fragile Base class problem

24 Explain marker interface ? List of the name of some marker interface

ans A marker interface in Java is an interface that does not declare any

method to implement but serve as a marker to tag for classes that implement it

- marker interface are used to indicate that class possesses some special characteristics or capabilities

- ~~Marker interface are used to indicate that class possess some~~

- Marker interface are often the following purpose

→ Java.io.Serializable - Indicates that do object state can be serialized & deserialized

→ Java.lang.Cloneable - Indicate that object can cloned to be the clone ()

→ Java.rmi.Remote - Used in JVM Remote method Invocation (RMI) Framework to identify remote object that can be accessed over a network

→ Java.util.EventListener - Used in Java event handling mechanism identify object that can handle events

25 Explain the significance of marker interface

ans → marker interface are interface that don't have fields, methods or content

⇒ In other term, a marker interface or tag interface is an empty interface.
It provides information about to object's runtime type

→ The JVM & compiles have additional information about object because of this

→ The interface for marker must be empty by the declaration is the same for ae interface in Java

→ syntax of marker interface in Java:

```
public interface my_markerInterface {
}
```