JAVA INTERVIEW QUESTIONS

Page No.
Date

☆ What do you mean by JDK, JRE & JVM.

1] JDK

⇒ JDK is an acronym for JAVA Development Kit.

⇒ It is a Software development environment used for developing JAVA applications and applets

⇒ It includes JAVA Runtime Environment, an interpreter/Loader (Java), a compiler (javac), an archiver (jar), a documentation generator (Javadoc) and other tools needed in java development.

⇒ It physically exists.

2] JRE

⇒ JRE is an acronym for JAVA Runtime environment.

⇒ It is a ~~program~~ software that allows the JAVA program to run smoothly

⇒ It communicates between the JAVA program and the operating System i.e it acts as a software layer on top of the operating System.

⟹ It is an open-access software distribution that has a Java class library, specific tools and a seperate JVM.

⟹ JAVA source code is compiled and converted to byte code. You need JRE, if you want to run this bytecode on any platform.

⟹ The JRE loads classes, checks memory access and get system resources.

3] JVM

⟹ JAVA Virtual Machine acts as a run-time engine to run Java applications.

⟹ It is the one that calls the main method in a JAVA code. It is a part of JRE.

⟹ JVM makes it possible to develop JAVA code once and Run it anywhere.

⟹ When we compile a .java file, .class file (contains byte-code) with the same class name present in .java file are generated by the JAVA compiler. This .class file goes into various steps when we run it. These steps together describe the whole JVM.

Q.2. Is JRE platform dependent or independent?

Ans. JRE is platform dependent. because different operating system have different requirements or configurations and every diff. operating system have their own specific JRE.

Page No.
Date

Q.3. Which is the ultimate base class in JAVA class hierarchy? List the name of methods of it?

→ object class is the Ultimate base class in JAVA.

→ object class is present in java.lang package

→ Every class is directly or indirectly derived from object class.

→ We can say that object class is the parent class of all the classes in JAVA by default i.e topmost class.

→ Some methods of Object class:-

1] toString();
2] hashCode();
3] equals(obj);
4] finalize();
5] clone();
6] getClass()

Q.4. Which are the reference types in java?

Ans. Reference types hold the references to objects and provide a means to access those objects stored somewhere in memory.

→ List of JAVA reference types:

* Array: Provides a fixed size data structure that stores data elements of same type.

* class: It provides Inheritance, polymorphism and encapsulation.
Consists of a set of ~~something~~ values that holds data and a set of methods that operates on the data.

* Enumeration: A reference for a set of objects that represent a related set of choices.

* Interface: Provides a public API and is "implemented" by JAVA classes.

* Annotation: Provides a way to associate meta-data (data about data) with program elements.

Q.5 Explain narrowing and widening?

Ans Converting a value from one primitive data type to another primitive data type is known as type casting.

There are two types of type casting :-

1) Narrowing
2) Widening

1) Narrowing

→ Converting a higher data type into a lower one is called narrowing type casting.

→ Also known as, explicit conversion or casting up.

→ It is to be done by the programmer explicitly.

→ If the programmer does not type cast then the compiler will show a Compile time error.

example

```
double d = 156.24
int i = d          // error : lossy conversion
int i = (int) d    // OK
```

Date

2] Widening

→ Converting a lower data type into higher data type is called widening.

→ Also known as implicit conversion or casting down.

→ It is done automatically

→ No chance of data loss

eg.

```
int i = 188
double d = i;
```

output: 188.0

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

→ 1] If-else method

```
class Intro {
    public static void main (String [] args) {
        if (System.out.print ("Hello World") == null) {
        }
    }
}
```

→ 2] using ~~equals~~ append methods

```
import java.util.*;
class Intro {
public static void main (String [] args) {

if (System.out.~~print~~ append ("Hello World") == null) {
        }
    }
}
```

→ 3] using equals methods

```
import java.util.*;
class Intro {
public static void main (String [] args) {
if (System.out.append ("Hello World").equals(null))
    {
    }
}
}
```

Q.7 Can ~~How will~~ you write java application without main function? If yes, how?

→ Yes, by using static ~~method~~ block we can execute a java program which doesnt have a main method.

→ Static block in JAVA is a group of statements that gets executed only once when the class is loaded into the memory by JAVA Class Loader. It is also known as static initialization block.

→ class Program Without Main {

static {

System.out.println ("class without main");

System.exit (0);

}

}

→ only allowed till JAVA 1.6 version

→ JAVA 7 and newer version JVM's check the presence of main method before initializing.

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

Ans. When we call the 'main' method from a static block, we are actually trying to execute the 'main' method as if it is just any other method.

→ When the 'main' method is executed from the inside of a static block it doesn't behave as a main method should.

→ This might lead to unexpected behaviour in running your JAVA application.

→ It also leads to non-standard program flow and makes our code less readable and harder to maintain.

→ Main method is the entry point from the program starts executing so we should avoid calling it from any other parts of the code.

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

Ans. System - It is a final keyword defined in the Java.lang package

out - It is an instance of print stream type.

println - All instances of the class printstream have a public method which displays result on the monitor.

Also
System :- It is a built in class in java.lang package. It provides access to standard input, output and error streams.

out :- A static member of the System class that represents the standard output stream. It is responsible for writing data to an output stream

println :- It is a method of the print stream class that writes a message to the output stream and appends a new line character.

Q:10 How will you pass an object to the function by reference?

→ Technically, Java is always pass by value, because even though a variable might hold a reference to an object, that object reference is a value that represent the object's location in memory.

~~"Class Name" obj = new "Classname"();~~

~~Class~~

~~int = obj.method();~~

// Passing the object as a value using pass by reference

```
public class Example {
  int a = 10;
  void call (Example eg ) {
    eg.a = eg.a +10;
  }
public static void main (String[] args) {
    Example eg = new Example();
  System.out.println("Before call by reference"+eg
```

// Passing the object as a value using pass by reference

```
    eg.call (eg);
  Syso ("After call by reference" + eg.a);
}
}
```

Q.11. Explain Constructor chaining? How can we achieve it in C++?

→ Constructor chaining is the process of calling one constructor from another constructor with respect to current object.

→ One of the main use of constructor chaining is to avoid duplicate codes while having multiple constructor and make code more readable.

→ Constructor chaining can be done in two ways:

1] Within same class:-
this() Keyword (Pointer) must be used for constructors in same class.

In JAVA 2] From base class:-
Super() Keyword must be used to call the constructor from base class.

→
```
#include <iostream>
using namespace std;
class Test {
private : int x;
public : void setx (int x)
{
  this -> x = x; }
Void print ()
{ cout << "x =" << x << endl; }
};
int main ()
{
Test obj;
int x = 20;
obj. setX (x);
obj. print ();
return 0;
}
```

Q.12 Which are the rules to overload method in sub class?

Ans→ 1] Method Signature:

→ The overloaded methods in subclass must have a different method signature than the methods in the superclass.

→ It includes method name and parameter list.

Ex.
1] Different parameter types.
2] Different number of parameters.
3] Different order of parameters.

2] Access Modifiers

→ The access level of the overloaded method in the subclass cannot be more restrictive than the method in the superclass.

→ It can be equally or less restrictive.

→ The order of access modifiers from most to least restrictive is private, default, protected, public.

→ If the superclass method is 'public' the subclass method can be 'public' 'protected'.

Q.12 (2)

Page No.
Date

3] Return type :

→ The return type can be the same type, or a subtype of the return type of the superclass method.

Q.13 Explain the difference among finalize and dispose?

Ans → 'Finalize' Method:

- 'Finalize' is a method in the 'java.lang.object' class, which is the superclass of all Java classes.
- It is a method that gets called by the garbage collector just before an object is destroyed.
- The purpose of the 'finalize' method is to allow an object to perform cleanup operations before it is garbage collected.
- It can be overridden in a class and its use is discouraged.

→ 'dispose' Method:

- 'dispose' is not a standard method or feature in Java language.
- It is more commonly seen in libraries or frameworks.
- It is custom method defined by the library or framework developers to release and clean up resources.
- It is not part of a Java standard library and its behaviour is specific to the library in which it is used.

Q.14. What is the difference between final, finally and finalize ?

→ Definition :

* final is a keyword and access modifier which is used to apply restrictions on a class or method or variable.

* finally is a block in java Exception handling to execute the important code, whether the exception occurs or not.

* finalize is the method in JAVA which is used to perform clean up processing just before object is garbage collected.

→ Applicable :

* final Keyword is used with the classes, methods and variable.

* finally block is always related to the try and catch block in exception handling.

* finalize() method is used with the object.

Q.14.(2)

→ Functionality :

* final : (1) once declared, final variable becomes constant and cannot be modified.
(2) final method cannot be overridden by sub class.
(3) final class cannot be inherited.

* finally : (1) finally block runs the important code even if exception occurs or not.
(2) finally block cleans up all the resources used in try block.

* finalize : 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 and catch block is executed. Its execution is not dependent on exception

* finalize method is executed just before the object is destroyed.

Q.15 Explain the difference among checked and Unchecked exception?

	Checked Exception	Unchecked Exception
*	It occurs at Compile time.	* It occurs at run-time.
*	The Compiler checks a checked exception	* The Compiler does not check these exceptions.
*	These exceptions can be handled at the time of Compilation.	* These types of exception cannot be a catch or handle at the time of Compilation, because they get generated by mistake
*	They are Sub-class of the exception class	* They are run-time exceptions and not part of Exception class.
*	JVM needs the exception to catch and handle.	* JVM does not require exception to catch and handle.
*	examples: • File not found exceptn • No such field exceptn • Interrupted exceptn • No Such method.	* examples • No Such element • Arithmetic • Security • Null pointer

Q.16 Explain exception chaining ?

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

→ It allows to relate one exception with another exception i.e one exception describes cause of another exception.

→ In JAVA, a chained exception is an exception that is caused by another exception. Chained exceptions are associated such that the previous exception causes each exception in the chain.

Exception chaining is helpful for several reasons

1] Providing Context : You can add context or additional information to the exception message.

2] Error diagnosis : Preserving the original exception as the cause allows for more effective debugging and analysis.

3] Consistency : It maintains a consistent pattern for error handling in your code, allowing you to centralize.

Q.17. Explain the difference among throw and throws?

→

throw	throws
1] The throw keyword is used inside a function. It is used when it is required to throw an exception logically.	1] The throws keyword is used in function signature. It is used for some statements that can lead to exceptions.
2] It is used to throw an exception explicitly. It can only be throw one exception a time	2] It is used to declare multiple exceptions, seperated by a comma. whichever exception occurs, if matched with the declared ones, is thrown automatically
3] Syntax includes instance of the exception to be thrown. Syntax wise throw is followed by instance variable	3] Syntax includes the class name of the exceptions to be thrown. throws keyword is followed by exception class names.
4] Only used to propogate the unchecked Exceptions that are not checked using the throws Keyword.	4] Throws Keyword is used to propogate the checked Exceptions only.

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

→ Finally block is not executed when the System.exit() method is called in the try block before the execution of finally block.

eg. try
{
System.exit(0);
}
catch(exception e)
{
}
finally
{
}

→ When an exception occurs in the code written in the finally block. In this case, finally block does not complete normally.

Q.19 Explain Up Casting?

Ans → In JAVA, like data types even objects can be type casted.

→ There are two types of objects i.e Parent and child objects

→ Typecasting Parent object to child object is known as ~~Upcasting~~ Down casting.

→ Typecasting child object to Parent object is known as ~~Down casting~~ Upcasting.

→ Up casting is done so that we can easily access the variables and methods of the parent class to the child class.

→ It is done implicitly.

→ It is not possible to access all the child class variables and methods only the overridden ones.

```
public class "Any name" {
    psvm (String [] args) {
        // upcasting
"ParentClass_name" obj = new "childclass_name"();
        obj.variable = " ";
{
}
```

Q.120 Explain Dynamic Method dispatch?

→ Runtime polymorphism or Dynamic Method Dispatch is a process in which a call to an overridden method is resolved at runtime rather than compile time.

→ A superclass reference variable can refer to a subclass object to resolve calls to overridden methods at runtime. When a superclass reference is used to call an overridden method, JAVA determines which version of the method to execute based on the type of the object being referred to at the time call.

☆ Advantages

→ It supports overriding of methods that are mandatory for Runtime polymorphism.

→ It allows subclasses to incorporate their own methods and define their implementation.

Q. 27. What do you know about final method?

Ans: Final is a Keyword in JAVA.

It is used to restrict the user.

A method when made final cannot be changed by any outsider.

or

If you make any method final, then you cannot override it.

Q. 22 Fragile base class problem and how we can overcome it?

→ Fragile base class problem is a problem of OOP where base classes (superclasses) are considered "fragile" because, some modifications made in the base class will be inherited by the derived class and may cause the derived class to malfunction

→ For example If you have a base class Shape() that defines a method draw() and a subclass Circle that overrides this method. Now if you add a new method rotate() to the Shape class and call it inside draw() method, the circle class will inherit it too. However this might not be what you want, as rotating a circle has no change in its appearance. This change in the base class can cause unexpected results or errors

→ This problem can be avoided by declaring the methods in the superclass as final, which would make it impossible for a subclass to override them. But this is not desirable or possible. Therefore, it is good practice for super classes to avoid changing calls to dynamically-bound methods.

Q.23 Why does not java support multiple implementation inheritance?

Ans Multiple inheritances will lead to ambiguity

~~If one then two~~ classes are inherited

If one class is inherited from two diff classes and if ~~the~~ both the parent classes happen to contain the method with a same name then the Compiler will not be able to decide which method to choose if called! So to avoid such conditions, Java does not allow multiple inheritance

Q.24. Explain Marker Interface? List the name of some marker interfaces?

Ans
→ An interface 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 gives run-time type information about objects.

→ It helps compiler and JVM by providing additional information about object.

→ Types of marker interface :-
1) Serializable interface.
2) Cloneable interface.
3) Remote interface.

Q.25. What is the significance of marker interface?

Ans → Marker interface has modes like Serializable and cloneable which are pre defined in Java library and which have methods and functions of their own which can be called explicitly. This provides ease to the programmer.