**JAVA - TECHNICAL**

**1.What is oops ?**

Object-oriented programming is a core of Java Programming, which is used for designing a program using classes and objects.

**2.Concepts of oops :**

1. Abstraction
2. Encapsulation
3. Inheritance
4. Polymorphism

**3. What is Class ?**

A class is a user-defined blueprint or prototype from which objects are created. It represents the set of properties or methods that are common to all objects of one type.

**4.What is objects ?**

An object is an instance of a class, Objects are allocated memory space whenever they are created.

**5.What is access modifiers and its types ?**

**Access Modifier**: Defines the access type of the method i.e. from where it can be accessed in your application. In Java, there are 4 types of access specifiers:

**public**: Accessible in all classes in your application.

**protected**: Accessible within the package in which it is defined and in its subclass(es) (including subclasses declared outside the package).

**private**: Accessible only within the class in which it is defined.

**default** (declared/defined without using any modifier): Accessible within the same class and package within which its class is defined.

**6.What is Abstraction ?**

**Abstraction in Java** is the process in which we only show essential details/functionality to the user. The non-essential implementation details are not displayed to the user.   
In Java, abstraction is achieved by**interfaces**and **abstract classes**. We can achieve 100% abstraction using interfaces.

**7.What is Encapsulation ?**

It is defined as the wrapping up of data under a single unit. It is the mechanism that binds together the code and the data it manipulates.

**8.What is Polymorphism and its types ?**

Polymorphism allows us to perform a single action in different ways. In other words, polymorphism allows you to define one interface and have multiple implementations. The word “poly” means many and “morphs” means forms, So it means many forms.

**Types of Java Polymorphism**

In Java Polymorphism is mainly divided into two types:

Compile-time Polymorphism

Runtime Polymorphism

**9. What is Compile-time Polymorphism ?**

In Java, Method Overloading allows different methods to have the same name, but different signatures where the signature can differ by the number of input parameters or type of input parameters, or a mixture of both.

Method overloading in Java is also known as Compile-time Polymorphism, Static Polymorphism, or Early binding.

**Example:**

public class Sum {

// Overloaded sum(). This sum takes two int parameters

public int sum(int x, int y) { return (x + y); }

// Overloaded sum(). This sum takes three int parameters

public int sum(int x, int y, int z)

{

return (x + y + z);

}

// Overloaded sum(). This sum takes two double

// parameters

public double sum(double x, double y)

{

return (x + y);

}

// Driver code

public static void main(String args[])

{

Sum s = new Sum();

System.out.println(s.sum(10, 20));

System.out.println(s.sum(10, 20, 30));

System.out.println(s.sum(10.5, 20.5));

}

}

**10.What is Runtime Polymorphism ?**

In Java, Overriding is a feature that allows a subclass or child class to provide a specific implementation of a method that is already provided by one of its super-classes or parent classes. When a method in a subclass has the same name, the same parameters or signature, and the same return type(or sub-type) as a method in its super-class, then the method in the subclass is said to override the method in the super-class.  
Method overriding is one of the ways by which Java achieves Run Time Polymorphism.

Example:

// method overriding in java

// Base Class

class Parent {

void show() { System.out.println("Parent's show()"); }

}

// Inherited class

class Child extends Parent {

// This method overrides show() of Parent

@Override void show()

{

System.out.println("Child's show()");

}

}

// Driver class

class Main {

public static void main(String[] args)

{

// If a Parent type reference refers

// to a Parent object, then Parent's

// show is called

Parent obj1 = new Parent();

obj1.show();

// If a Parent type reference refers

// to a Child object Child's show()

// is called. This is called RUN TIME

// POLYMORPHISM.

Parent obj2 = new Child();

obj2.show();

}

}

**11.What is Inheritance?**

It is the mechanism in Java by which one class is allowed to inherit the features (fields and methods) of another class. We are achieving inheritance by using extends keyword. Inheritance is also known as “is-a” relationship.

Example:

//base class or parent class or super class

class A{

//parent class methods

void method1(){}

void method2(){}

}

//derived class or child class or base class

class B extends A{ //Inherits parent class methods

//child class methods

void method3(){}

void method4(){}

}

**12.Types of Inheritance :**

1. Single Inheritance
2. Multilevel Inheritance
3. Hierarchical Inheritance
4. Multiple Inheritance (Through Interfaces)
5. Hybrid Inheritance

**13.what is Interface?**

The interface in Java is a mechanism to achieve abstraction. Traditionally, an interface could only have abstract methods (methods without a body) and public, static, and final variables by default. It is used to achieve abstraction and multiple inheritances in Java.

**14.what is Constructors?**

In Java, a Constructor is a block of codes similar to the method. It is called when an instance of the class is created. At the time of calling the constructor, memory for the object is allocated in the memory. It is a special type of method that is used to initialize the object. Every time an object is created using the new() keyword, at least one constructor is called.

**15.Types of Constructors :**

Default Constructor

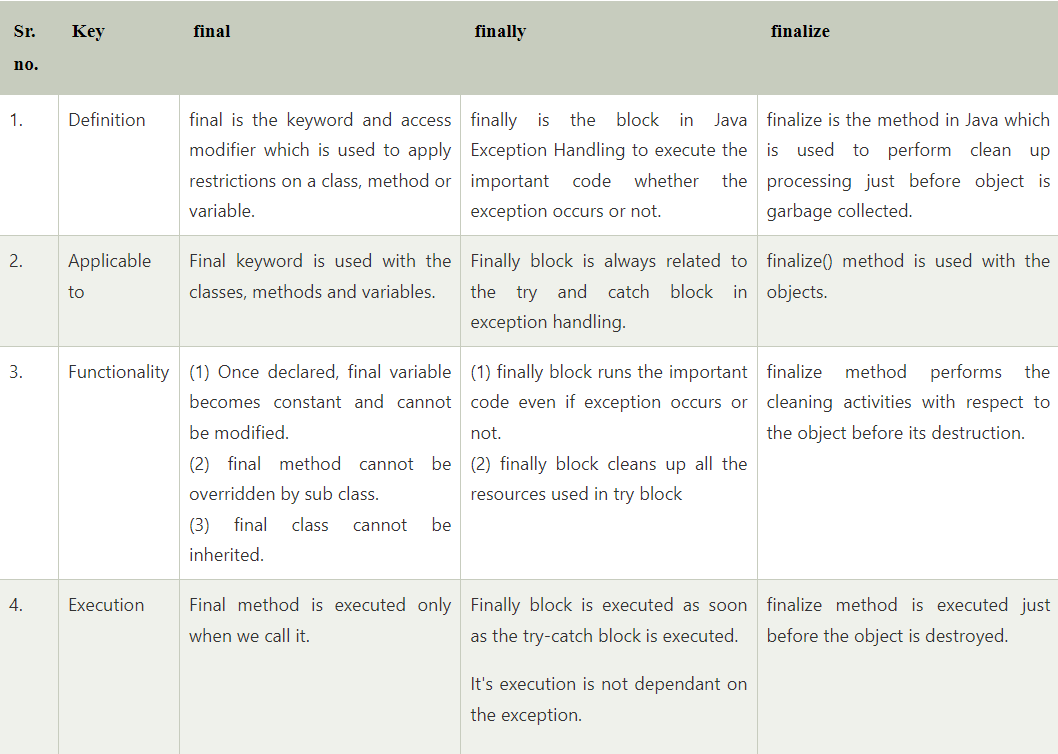
Parameterized Constructor

Copy Constructor

**16.What is copy constructors?**

Unlike other constructors copy constructor is passed with another object which copies the data available from the passed object to the newly created object.

**17. Final Finally Finalize in java ?**



**18. JVM, JRE, JDK :**

**JVM**

JVM (Java Virtual Machine) is an abstract machine. It is called a virtual machine because it doesn't physically exist. It is a specification that provides a runtime environment in which Java bytecode can be executed. It can also run those programs which are written in other languages and compiled to Java bytecode.

**JRE**

JRE is an acronym for Java Runtime Environment. It is also written as Java RTE. The Java Runtime Environment is a set of software tools which are used for developing Java applications. It is used to provide the runtime environment. It is the implementation of JVM. It physically exists. It contains a set of libraries + other files that JVM uses at runtime.

**JDK**

JDK is an acronym for Java Development Kit. The Java Development Kit (JDK) is a software development environment which is used to develop Java applications and applets. It physically exists. It contains JRE + development tools.

**19. What is Exception handling ?**

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

The core advantage of exception handling is to maintain the normal flow of the application.

**20. Types of Exception :**

1. Checked Exception

The classes that directly inherit the Throwable class except RuntimeException and Error are known as checked exceptions. For example, IOException, SQLException, etc. Checked exceptions are checked at compile-time.

1. Unchecked Exception

The classes that inherit the RuntimeException are known as unchecked exceptions. For example, ArithmeticException, NullPointerException, ArrayIndexOutOfBoundsException, etc. Unchecked exceptions are not checked at compile-time, but they are checked at runtime.

**21. What is Static Keyword ?**

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.

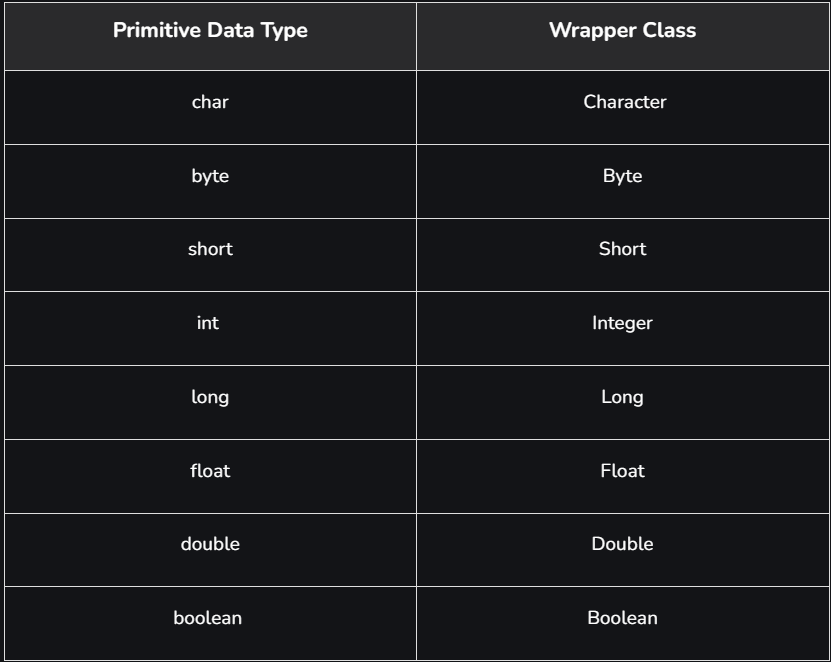
**22. why java is not a 100% oops ?**

No, Java is not a fully object-oriented language as it supports primitive data types like int, byte, long, short, etc., which are not objects.  
  
**23. What are the primitive and non-primitive datatype ?**

Primitive data types - includes byte , short , int , long , float , double , boolean and char. Non-primitive data types - such as String , Arrays and Classes.

**24.What is Wrapper Class ?**

A Wrapper class in Java is a class whose object wraps or contains primitive data types. When we create an object to a wrapper class, it contains a field and in this field, we can store primitive data types.



**Autoboxing and Unboxing**

**1. Autoboxing**

The automatic conversion of primitive types to the object of their corresponding wrapper classes is known as autoboxing. For example – conversion of int to Integer, long to Long, double to Double, etc.

**2. Unboxing**

It is just the reverse process of autoboxing. Automatically converting an object of a wrapper class to its corresponding primitive type is known as unboxing. For example – conversion of Integer to int, Long to long, Double to double, etc.

**25. What is Enums ?**

An enum is a special "class" that represents a group of constants (unchangeable variables, like final variables).