OOPS — OOPS Stands for object-oriented programming system/structure.

The main principles of object-oriented programming are Abstraction, Encapsulation,Inheritance and Polymorphism.

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

1. Inheritance :
inheritance.
One class acquires property of another class with the help of extends keywords is known as

Importance of the inheritances
The most important use of inheritance in java is code reusability. The code that is present in parent class can be directly used by the child class

Types of inheritance

1. Single level inheritance
2. Multilevel inheritance
3. Multiple inheritance
4. Hybrid inheritance
5. Hierarchical inheritance

1. Single level inheritance
It is an operation where inheritance takes place between 2 classes. To perform single level inheritance only two classes are required.
The class from where properties are acquiring/inheriting is called super class/base class/parent class
The class to where properties are inherited/delivered is called sub class/child class.

2. Multilevel inheritance
Multilevel Inheritance takes place between 3 or more than 3 classes.
In Multilevel Inheritance 1 sub class acquires properties of another super class & that class acquires properties of another super class & phenomenon continuous.

3. Hierarchical Inheritance:
Multiple sub classes can acquire properties of 1 super class is known as hierarchical Inheritance.

4. Multiple Inheritance:
1 subclass acquiring properties of 2 super classes at the same time is known as Multiple Inheritance.
Java doesn't support Multiple Inheritance using class because of diamond ambiguity problem.
By using interface we can achieve Multiple Inheritance.

Note: object class is the super most class in all java

5.Hybrid inheritance :
The hybrid inheritance is the combination of two or more than two types of inheritance.

Question - Why multiple & hybrid inheritance is not supported in java?

To reduce the complexity and simplify the language, multiple inheritance is not supported in java.
Example - Consider a scenario where A, B, and C are three classes. The C class inherits A and B classes. If A and B classes have the same method and you call it from child class object, there will be ambiguity to call the method of A or B class.

Since compile-time errors are better than runtime errors, Java renders compile- time error if you inherit 2 classes. So whether you have same method or different, there will be compile time error.

Access Modifiers or specifiers

The access modifiers in Java specifies the accessibility or scope of a field, method, constructor, or class. We can change the access level of fields, constructors, methods, and class by applying the access modifier on it.

There are four types of Java access modifiers:

1. **Private**: The access level of a private modifier is only within the class. It cannot be accessed from outside the class.

2. **Default**: The access level of a default modifier is only within the package. It cannot be accessed from outside the package. If you do not specify any access level, it will be the default.

3. **Protected**: The access level of a protected modifier is within the package and outside the package through child class. If you do not make the child class, it cannot be accessed from outside the package.

4. **Public**: The access level of a public modifier is everywhere. It can be accessed from within the class, outside the class, within the package and outside the package.

Access Modifier	Same Class	Same Package (Non-Subclass)	Subclass (Different	Different Package (Non-Subclass)	Description
public	✔ Allow	✔ Allowed	✔ Allowed	✔ Allowed	Accessible everywhere.
protected	✔ Allowed	✔ Allowed	✔Allowed (through inheritance)	❌ Not accessible without inheritance	Visible within same packa and subclasses.
default *(no modifier)*	✔ Allowed	✔ Allowed	❌ Not allowed	❌ Not allowed	Package-private: only within the same package.
private	✔ Allow	❌ Not allowed	❌ Not allowed	❌ Not allowed	Accessible only inside the same class.

2. Polymorphism

It is one of the oops principle where one object showing different Behaviour at different stage is known as **Polymorphism**.

Types of Polymorphism

1 Compile time Polymorphism
2.Run time Polymorphism

Compile time Polymorphism

In Compile time Polymorphism method declaration and Definition are binded during the

compilation time based on argument or input parameter is known as compile time poly.

Example - Method overloading

Method overloading

When the method name is same with different argument/input param with different data types within the same class is called as method overloading.

It is also known as early binding.
We can overload the non static and static method. We can overload the main method
You can any number of method in a class but JVM class main () methods which receives string array as arguments only.

Run Time Polymorphism

In Run time Polymorphism method declaration and Definition are binded during the run time or execution time based on input parameter or argument.

Example - Method overriding

When the method is present in parent class as well as in child class with same name and same number of arguments/input parameter is called as method overriding.
In overriding method resolution always take care by JVM based on
Run time object no based on reference type that why it is also known as run time polymorphism or dynamic Polymorphism or late binding.
The process of compiler trying to resolve the method call from given overloaded method definitions is called overload resolution.