**1.What are the core OOPs concepts or main principles of object-oriented programming or stuffs in java**

**Class**

**Object**

**Abstraction:** Protecting data of a class from being accessed by members of another class

**Encapsulation:** Hiding data of class from other classes

**Inheritance:** Using code written in a class inside other classes

**Polymorphism:** Using various methods with same name

QWhat is OOPS in java?

Object-oriented programming aims to implement real-world entities like inheritance, hiding, polymorphism etc. in programming. The main aim of OOP is to bind together the data and the functions that operate on them so that no other part of the code can access this data except that function. OOPs make the Java program reusable, easier to modify and debug, make the program more structural and modular, and makes the code short and simple. OOps in java is to improve code readability and reusability by defining a Java program efficiently

Q why the java is called object programming language?

Java Language is considered an object-oriented language because it is based on the concept of objects and classes. Without the creation of objects and classes, it is ‘/ c impossible to write any code in Java. Java supports the concepts of OOPS - Inheritance, Data abstraction, polymorphism, and data encapsulation.

Q.Why the java is not pure object programming language?

The first reason is that the Object-oriented programming language should only have objects whereas java contains 8 primitive data types like **char, boolean, byte, short, int, long, float, double** which are not objects. These primitive data types can be used without the use of any object. Even using [**Wrapper classes**](https://www.javaguides.net/2018/08/wrapper-classes-in-java.html) does not make Java a pure OOP language, as internally it will use the operations like [**Unboxing and Autoboxing**](https://www.javaguides.net/2018/08/autoboxing-and-unboxing-in-java-with.html) (Autoboxing is the automatic conversion that the Java compiler makes between the primitive types and their corresponding object wrapper classes. For example, converting an int to an Integer, a double to a Double, and so on. If the conversion goes the other way, this is called unboxing.)The second reason is related to the static keyword. In a pure object-oriented language, we should access everything by message passing (through objects). But java contains static variables and methods which can be accessed directly without using objects.

Q.what is class in java?

A class is a group of objects which have common properties. It is a template or blueprint from which objects are created. It is a logical entity. It can't be physical. Inside a class, we define variables, constants, member functions, and other functionality. It does not consume memory at run time. It is user defined data type

Q what is object? An object is a real-world entity that has attributes, behavior, and properties. It is referred to as an instance of the class. It contains member functions, variables that we have defined in the class. It occupies space in the memory. Different objects have different states or attributes, and behaviors. An object is *a runtime entity*. We can cereate object using new keyword

* An object is *a runtime entity*.
* The object is *an entity which has state and behavior*.
* The object is *an instance of a class*.
* **State:** represents the data (value) of an object.
* **Behavior:** represents the behavior (functionality) of an object such as deposit, withdraw, etc.
* **Identity:** An object identity is typically implemented via a unique ID. The value of the ID is not visible to the external user. However, it is used internally by the JVM to identify each object uniquely.

|  |  |  |
| --- | --- | --- |
| **SN** | **‘this’ keyword** | **‘super’ keyword** |
| 1. | “this” is a reference variable that contains current class objects. | “super” is a reference variable that contains immediate super class objects. |
| 2. | Any member of the current class object from within an instance method or a constructor can be referred by using this keyword. | If the method overrides one of its super class’s method, the overridden method can be called through the use of super keyword. |
| 3. | ‘this’ keyword is used to call another constructor from within a constructor in the same class. | ‘super’ keyword is used to call the super class’s constructor from within a constructor of the subclass. |
| 4. | JVM never put automatically this() keyword like super() in Java. | By default JVM automatically put the super() keyword at first line inside the constructor. |

**Q.Expalin Super Keyword?**

The **super** keyword in Java is a reference variable which is used to refer immediate parent class object.

Whenever you create the instance of subclass, an instance of parent class is created implicitly which is referred by super reference variable.

## **Usage of Java super Keyword**

1. super can be used to refer immediate parent class instance variable.
2. super can be used to invoke immediate parent class method.
3. super() can be used to invoke immediate parent class constructor.

Question on Inheritance

1)What is inheritance in java?

In Java, Inheritance means creating new classes based on existing ones. A class that inherits from another class can reuse the methods and fields of that class. In addition, you can add new fields and methods to your current class as well.  It is the mechanism in Java by which one class is allowed to inherit the features(fields and methods) of another class.Child class can access the property of parent class Inheritance represents the **IS-A relationship** which is also known as a parent-child relationship

### Q. Why use inheritance in java?

* **Code Reusability:**The code written in the Superclass is common to all subclasses. Child classes can directly use the parent class code.
* **Method Overriding:**[Method Overriding](https://www.geeksforgeeks.org/overriding-in-java/) is achievable only through Inheritance. It is one of the ways by which Java achieves Run Time Polymorphism.
* nheritance provides code reusability. The derived class does not need to redefine the method of base class unless it needs to provide 0piu7rf the specific implementation of the method.
* Runtime polymorphism cannot be achieved without using inheritance.
* We can simulate the inheritance of classes with the real-time objects which makes OOPs more realistic.
* Inheritance provides data hiding. The base class can hide some data from the derived class by making it private.
* Method overriding cannot be achieved without inheritance. By method overriding, we can give a specific implementation of some basic method contained by the base class.

**Q.What is Super Class/Parent class?**

**Super Class/Parent Class:**The class whose features are inherited is known as a superclass(or a base class or a parent class).

Q.What is child class, derived class, subclass?

**Sub Class/Child Class:** The class that inherits the other class is known as a subclass(or a derived class, extended class, or child class). The subclass can add its own fields and methods in addition to the superclass fields and methods

## Q. **How to Use Inheritance in Java?**

The **extends keyword**is used for inheritance in Java. Using the extends keyword indicates you are derived from an existing class. In other words, “extends” refers to increased functionality.

Q.Explain Type of Inheritance in details?

1)Single level inheritance:In single level inheritance child class/sub class inherits features /properties of parent class

Ex: class A{

}

Class B extends class A{

}

2)Multilevel inheritance :

In Multilevel Inheritance, a derived class will be inheriting a base class, and as well as the derived class also acts as the base class for other classes. In the below image, class A serves as a base class for the derived class B, which in turn serves as a base class for the derived class C. In Java, a class cannot directly access the[grandparent’s members](https://www.geeksforgeeks.org/g-fact-91/).

Ex: class A{

}

Class B extends class A{

}

Class C extends class B{

}

### ****3. Hierarchical Inheritance****

In Hierarchical Inheritance, one class serves as a superclass (base class) for more than one subclass. In the below image, class A serves as a base class for the derived classes B, C, and D.

Ex: Class A{

}

Class B extends Class A{

}

Class C extends class A{

}

Class D Extends class A{

}

4)Multiple Inheritance: Multiple inheritance in java is the capability of creating a single class with multiple superclasses **Multiple Inheritance** is the process in which a subclass inherits more than one superclass.

Q.Why the multiple inheritance is not allowed in java?

Because there is an ambiguity **Multiple inheritance faces problems when there exists a method with the same signature in both the superclasses**. To reduce the complexity and simplify the language, Due to such a problem, java does not support multiple inheritance.

Exaple Consider a case where class B extends class A and Class C and both class A and C have the same method display().

Now java compiler cannot decide, which display method it should inherit. To prevent such situation, multiple inheritances is not allowed in java.

Question On Plymorphism

Q.What is polymorphism?

Polymorphism means taking many forms or shapes. In Java, polymorphism occurs when multiple child classes inherit the methods of a parent class. For example, a super class named vehicles with method wheels(); will have sub-classes such as Car, Scooter, etc. Each of these will have its own implementation of the method wheels.

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. Polymorphism allows us to create consistent code

Ex: The print() method is also an example of polymorphism. It is used to print values of different types like char , int , string , etc. Polymorphism is the ability of an object to take on many forms.

real world example is your mobile. Sometime your mobile behaves as a phone, sometime as a camera, sometime as a radio etc. Here the same mobile phone has different forms, so we can say the mobile object is polymorphic in nature.

Q.How The Polymorphism achieve in java? it describes the concept that you can access objects of different types through the same interface

Polymorphism can be achieved by method overloading, method overriding and operator overloading.

QWhat is compile Time Polymorphism?// Method Overloading?

Compile time polymorphism is known as early binding and static binding

Static method can be overloaded is an example of Compile time polymorphism **Compile-time polymorphism allows us to use many methods with the same name but differing signatures and return types**.

Method Overloading is the one the class having the multiple method with same name but different parameter /signature

Method overloading can be done in two ways

1)By changing the argument

2)By changing the data type

Return type of method can be different

Ex: public class ABC{

Public void test(){

}

Public void test(int a){

}

Public void test(Flaot b){

}

Public int test (int a, int b){

Return statement;

}

}

Private method can be overloaded in java , final method can be overloaded in java, the main method can be overloaded in java , both stati and instance method can be overloaded in java

**Q.When to use method overloading in Java?**

🅰 Method overloading is used for the following purpose. They are:

1. Method overloading is used when we need to perform same task with different parameters.
2. Method overloading is done to reuse the same method name.
3. It is used to achieve the compile-time polymorphism in Java.

**Q.Can we overload the main() method?**

The entry point for the JVM to begin its execution is "public static void main(String[] args)". Without the original main() method, the class will compile but won't execute. The overloaded main methods won't execute on their own, we need to call it from the actual original method only. The answer is, **yes, we can overload the main() method**. But remember that the JVM always calls the original main() method. It does not call the overloaded main() method

Q**Why method overloading is not possible by changing return type of method?**

🅰 In Java, Method overloading cannot be done when the return type, method name, and argument list are the same because there may occur ambiguity.

**25. Can we overload main() method in Java?**

🅰 Yes, we can overload the main() method in Java. A class can have any number of main() methods but the execution always starts from public static void main(String[] args) only.

**Q. What is upcasting and downcasting in Java?**

🅰 When the reference variable of super class refers to the object of sub class, it is known as upcasting.

When subclass reference refers to super class object, it is called downcasting.

Q.What is run time polymorphism?//Method overiiding?

Runtime polymorphism, also known as the Dynamic Method Dispatch, is a process that resolves a call to an overridden method at runtime. The process involves the use of the reference variable of a superclass to call for an overridden method

Runtime polymorphism is done by using method overriding in inheritance .

Method overriding means If a subclass of class i.e child class have the same method as declared in parent class For method overriding inheritance is required

In run time polymorphism there should be one parent class and one child class

For method overriding return type of method should be same , method should be same, type and no of argument should be same

* The method must have the same name as in the parent class.
* The method must have the same signature as in the parent class.
* Two classes must have an IS-A relationship between them.

Q.Why the method overriding is called runtime polymorphism?

 method binding between  **method call and method definition happens at run time and it depends on the object of the class** (object created at runtime and goes to the heap).

### Q.Can we override the static method?

No, you can't override the static method because they are the part of the class, not the object.

### Q.Can we override the overloaded method?

### Yes we can override the overload method because overriding means method name should be same with same signature both parent and child class and Overloading occurs when two or more methods in the same class have the same name but different parameters. Yes, we can override a method which is overloaded in super class.

### Q. Can we override the private methods?

No, we cannot override the private methods because the scope of private methods is limited to the class and we cannot access them outside of the class.

**Overriding occurs when the method signature is the same in the superclass and the child class**. Overloading occurs when two or more methods in the same class have the same name but different parameters

### Q.Difference Between method overloading and method overriding

|  |  |
| --- | --- |
| **Method Overloading** | **Method Overriding** |
| 1) Method overloading increases the readability of the program. | Method overriding provides the specific implementation of the method that is already provided by its superclass. |
| 2) Method overloading occurs within the class. | Method overriding occurs in two classes that have IS-A relationship between them. |
| 3) In this case, the parameters must be different. | In this case, the parameters must be the same. |

### Q. Which class is the superclass for all the classes?

The object class is the superclass of all other classes in Java.

### Q.Can we override the private methods?

### No, we cannot override the private methods because the scope of private methods is limited to the class and we cannot access them outside of the class.

### Q.What is use of final keyword?

n Java, the final keyword can be used to indicate that something cannot be changed. It can be used in several contexts, such as **to declare a variable as a constant, to declare a method as final, or to declare a class as final. Once any data member (a variable, method, or class) gets declared as final, it can only be assigned once.**

* **The final variable cannot be reinitialized with another value.**
* **A final method cannot be overridden by another method.**
* **A final**[**class**](https://www.shiksha.com/online-courses/articles/oops-concepts-in-java/#s2)**cannot be extended or inherited by another child class.**

### Q. What is the final class?

### If we make any class final, we can't inherit it into any of the subclasses.

### Q.  What is the final method?

### If we change any method to a final method, we can't override it. By anither method

### Q.Can you declare the main method as final?

Yes, We can declare the main method as public static final void main(String[] args){}.

### Question on Abstraction

Q.What is Abstraction ?

Abstraction is the process of hinding implementation details and showing the functionality to user

There are two ways to achieve abstraction

1)Abstract class

2)Interface

Q.What is abstract class?

A class which declared with yhe abstract keyword .It can have abstract and non abstract method .Abstract method do not have body. We can not make object of abstract class. Abstract method have onl method signature or parameter but don’t have body

A class that is declared as abstract is known as an abstract class. It needs to be extended and its method implemented. It cannot be instantiated. It can have abstract methods, non-abstract methods, constructors, and static methods. It can also have the final methods which will force the subclass not to change the body of the method

Q. It is possible to create abstract class without abstract method

Ans: Yes

Q.Why Abstract class has constructor even though you cannot create object?

We cannot create object of of abstract class but we create object of subclass of abstract class When we create object of subclass of abstract class it calls the constructor of subclass

This subclass constructor has a super keyword in the first line that calls constructor of an abstract class This constructor of an abstract class are used from its subclass

If the abstract class doesn’t have constructor a class that extends that abstract class will not get compiled

Q.To Achieve abstraction we have abstract class so why we use interface ?

Because in interface all method are abstract so if we want 100 % abstraction we use interface and if we wann 0-100% abstract method and non abstract method we want to achieve then use abstract class

### Q.Can you use abstract and final both with a method?

No, because we need to override the abstract method to provide its implementation, whereas we can't override the final method.

### Q.Is it possible to instantiate the abstract class?

### No

Q Difference betwwn abstract and interface?

|  |  |
| --- | --- |
| **Abstract class** | **Interface** |
| An abstract class can have a method body (non-abstract methods). | The interface has only abstract methods. |
| An abstract class can have instance variables. | An interface cannot have instance variables. |
| An abstract class can have the constructor. | The interface cannot have the constructor. |
| An abstract class can have static methods. | The interface cannot have static methods. |
| You can extend one abstract class. | You can implement multiple interfaces. |
| The abstract class **can provide the implementation of the interface**. | The Interface **can't provide the implementation of the abstract class**. |
| The **abstract keyword** is used to declare an abstract class. | The **interface keyword** is used to declare an interface. |
| An **abstract class** can extend another Java class and implement multiple Java interfaces. | An **interface** can extend another Java interface only. |
| An **abstract class** can be extended using keyword **extends** | An **interface class** can be implemented using keyword **implements** |
| A Java **abstract class** can have class members like private, protected, etc. |  |

No, the abstract class can never be instantiated even if it contains a constructor and all of its methods are implemented.

Q.What is the Encapsulation?

It is the process of wrapping the code and data together into single unit

Encapsulation in Java can be defined as **a mechanism using which the data and the methods that work on that data is wrapped to form a single**

Q.What is fully encapsulated class?

A class is said to be tightly encapsulated **if and only if, all the data members(variables) of that class is declared as private**. if class has all variables declared as private and if it's inherited by other class which too has all private data members, then the later one is also said to be tightly encapsulated class.

### Q.What is interface?

An **interface in Java** is a blueprint of a class. It has static constants and abstract methods.

The interface in Java is a mechanism to achieve [*abstraction*](https://www.javatpoint.com/abstract-class-in-java). There can be only abstract methods in the Java interface, not method body. It is used to achieve abstraction and multiple [inheritance in Java](https://www.javatpoint.com/inheritance-in-java).

In other words, you can say that interfaces can have abstract methods and variables. It cannot have a method body.

## **Why use Java interface?**

There are mainly three reasons to use interface. They are given below.

* It is used to achieve abstraction.
* By interface, we can support the functionality of multiple inheritance.
* It can be used to achieve loose coupling.