**Java Questions**

1. **What is OOPS?**

Object Oriented programming is a programming style that is associated with the concept of OBJECTS. It defines in terms of attributes and activities on Object. Objects are instances of classes and are used to interact amongst each other to create application.

1. **What are OOPS principles?**

Abstraction, Encapsulation, Inheritance and Polymorphism.

1. **What is Inheritance?**

Inheritance is the process by which one object acquires the properties of another object.

A class whose properties are inherited by another class is known as Parent class.

A class which inherits the properties is known as Child class.

1. **Why we need to use Inheritance?**

For Code Re usability AND For Extensibility.

1. **How can you achieve code reusability or example of Inheritance?**

Let’s consider a superclass **Vehicle**. Different vehicles have different features and properties however there few of them are common to all. Speed, color, fuel used, size are few which are common to all. Hence we can create a class ‘Vehicle’ with states and actions that are common to all vehicles. The subclass of this superclass can be any type of vehicle. Example: **Class Ca**r  A has all the features of a vehicle. But it has its own attributes which makes it different from other subclasses. By using inheritance we need not rewrite the code that we’ve already used with the Vehicle. The subclass can also be extended. We can make a class ‘**Sports Car**’ which extends ‘Car’. It inherits the features of both ‘Vehicle’ and ‘Car’.

NOTE:

1. The derived class cannot inherit a member of the base class if the derived class declares another member with the same name. (also known as Data hiding. We can call base class attribute in derived class using super keyword.)
2. Base class – private members cannot be inherited.
3. **What are types of inheritance supported in java ?**
4. Single Inheritance.
5. Multilevel Inheritance
6. Hierarchical Inheritance
7. **Why java does not support Multiple Inheritance?**

It leads to diamond problem. Example:

Class A : drive();

Class B : drive();

Class C extends A and B.

C c = new C();

c.drive(); // Compiler will give error at this line. Because it is not sure which drive() method will be called which leads to ambiguity.

1. **How can you implement Inheritance in java?**

Inheritance can be implemented in JAVA using below two keywords.  
1.**extends** – developing inheritance between two classes or two interfaces

2.**implements** - to develop inheritance between interface and class.

1. **What is Encapsulation?**

Encapsulation means hiding the internal details and allowing a simple interface which ensures that the object can be used without having to know how it works.

1. Example of Encapsulation?

POJO class.. having variables as private and provide getter setter for variables.

1. Where you should use encapsulation?
2. Where we want to control the access of data to outside world.
3. We can control the access of data using access modifiers- private, public , protected or default.
4. What is Abstraction?

Abstraction is the process of identifying essential details to be known and ignoring the non-essential details from perspective of each user.

1. **What is the difference between Abstract class and Interface**
2. Abstract class is a class which contain one or more abstract methods, which has to be implemented by sub classes. An abstract class can contain no abstract methods also i.e. abstract class may contain concrete methods.

A Java Interface can contain only method declarations and public static final constants and doesn't contain their implementation. The classes which implement the Interface must provide the method definition for all the methods present.

1. Abstract classes are useful in a situation when some general methods should be implemented and specialization behavior should be implemented by subclasses.

Interfaces are useful in a situation when all its properties need to be implemented by subclasses

1. **What is meant by "Abstract Interface"?**

Firstly, an interface is abstract. That means you cannot have any implementation in an interface.   
All the methods declared in an interface are abstract methods or signatures of the methods.

1. **If interface & abstract class have same methods and those methods contain no implementation, which one would you prefer?**

Obviously one should ideally go for an interface, as we can only extend one class. Implementing an interface for a class is very much effective rather than extending an abstract class because we can extend some other useful class for this subclass

1. **When do you favor abstract class over interface?**  
   Since it’s almost impossible to add a new method on a published interface, it’s better to use abstract class, when evolution is concern. Abstract class in Java evolves better than interface. Similarly, if you have too many methods inside interface, you are creating pain for all it’s implementation, consider providing an abstract class for default implementation
2. **Can abstract class have constructors in Java?**

Yes, abstract class can declare and define constructor in Java. Since you can not create instance of abstract class,  constructor can only be called during [constructor chaining](http://javarevisited.blogspot.com/2012/12/constructor-chaining-in-java-calling-another-constructor.html), i.e. when you create instance of concrete implementation class. Now some interviewer, ask what is the purpose of constructor, if you can not instantiate abstract class? Well, it can still be used to initialize common variables, which are declared inside abstract class, and used by various implementation. Also even if you don’t provide any constructor, compiler will add [default no argument constructor](http://javarevisited.blogspot.com/2012/12/what-is-constructor-in-java-example-chainning-overloading.html) in abstract class, without that your subclass will not compile, since first statement in any constructor implicitly calls super(), default super class constructor in Java.

1. **Can abstract class implements interface in Java? does they require to implement all methods?**  
   Yes, abstract class can implement interface by using implements keyword. Since they are abstract, they don’t need to implement all methods. It’s good practice to provide an abstract base class, along with an interface to declare Type. One example of this is java.util.List interface and corresponding java.util.AbstractList abstract class. Since AbstractList implements all common methods,  concrete implementations like [LinkedList](http://javarevisited.blogspot.com/2012/02/difference-between-linkedlist-vs.html) and [ArrayList](http://javarevisited.blogspot.com/2012/03/how-to-loop-arraylist-in-java-code.html) are free from burden of implementing all methods, had they implemented List interface directly. It’s best of both world, you can get advantage of interface for declaring type, and flexibility of abstract class to implement common behavior at one place.

### Can abstract class be final in Java?

No, abstract class cannot be final in Java. Making them final will stop abstract class from being extended, which is the only way to use abstract class. They are also opposite of each other, abstract keyword enforces to extend a class, for using it, on the other hand, [final keyword](http://javarevisited.blogspot.com/2011/12/final-variable-method-class-java.html) prevents a class from being extended. In real world also, abstract signifies incompleteness, while final is used to demonstrate completeness. Bottom line is, you can not make your class abstract and final in Java, at same time, it’s a compile time error.

1. **Can abstract class have static methods in Java?**  
   Yes, abstract class can declare and define [static methods](http://javarevisited.blogspot.com/2011/11/static-keyword-method-variable-java.html), nothing prevents from doing that. But, you must follow guidelines for making a method static in Java, as it’s not welcomed in a object oriented design, because [static methods can not be overridden in Java](http://javarevisited.blogspot.com/2013/03/can-we-overload-and-override-static-method-java.html). It’s very rare, you see static methods inside abstract class, but as I said, if you have very good reason of doing it, then nothing stops you.
2. **Can abstract class contains main method in Java ?**Yes, abstract class can contain [main method](http://javarevisited.blogspot.sg/2011/12/main-public-static-java-void-method-why.html), it just another static method and you can execute Abstract class with main method, until you don’t create any instance.
3. **What is Association?**

Association is a relationship where all object have their own lifecycle and there is no owner. Let's take an example of Teacher and Student. Multiple students can associate with single teacher and single student can associate with multiple teachers but there is no ownership between the objects and both have their own lifecycle. Both can create and delete independently.

1. **What is Aggregation?**

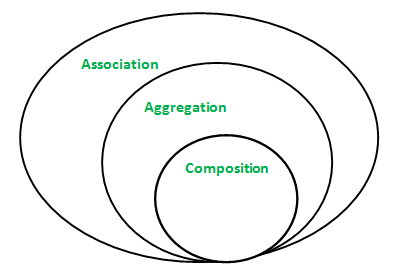
Aggregation is a specialize form of Association where all object have their own lifecycle but there is ownership and child object can not belongs to another parent object. Let's take an example of Department and teacher. A single teacher can not belongs to multiple departments, but if we delete the department teacher object will not destroy. We can think about "has-a" relationship.

1. **What is Composition ?**

Composition is again specialize form of Aggregation and we can call this as a "death" relationship. It is a strong type of Aggregation. Child object dose not have their lifecycle and if parent object deletes all child object will also be deleted. Let's take again an example of relationship between House and rooms. House can contain multiple rooms there is no independent life of room and any room can not belongs to two different house if we delete the house room will automatically delete.

NOTE:

If cow(container) dies, the objects(legs) are also dead. This is composition. That is not the case in aggregation. If car(container) dies, not necessary that the object(driver) should die. Hence this is aggregation. Hope this helps.



1. **What is Polymorphism?**

Polymorphism means many forms. It refers to the ability of an object to behave differently in different situations.

1. **How Polymorphism implemented in java?**

Method Overloading and Method Overriding.

1. **How will it work? Is it compile time binding or runtime? Which polymorphism is it?**

CASE 1:

Class A: method1();

Class B extends A: method1();

A obj = new B();

obj.method1();

CASE 2:

Cass A

Class B extends A: method1();

A obj = new B();

obj.method1();

CASE 3:

Cass A

Class B extends A: method1();

B obj = new B();

obj.method1();

CASE 4:

Cass A: private method1();

Class B extends A: private method1();

B obj = new B();

obj.method1();

And

A obj = new B();

Obj.method1();

CASE 5:

Cass A: public method1();

Class B extends A: private method1();

B obj = new B();

obj.method1();

And

A obj = new B();

Obj.method1();

CASE 6:

Cass A: public method1(){}

Class B extends A: public method1(){} public method2(){}

A obj = new B();

obj.method2();

// Compile time error. Cannot call B’s method from A’s reference.

CASE 7:

class A

{

void m1() throws ArrayIndexOutOfBoundsException

{

System.out.println("In m1 A");

}

}

class B extends A

{

void m1() throws IndexOutOfBoundsException

{

System.out.println("In m1 B");

}

}

public class Test {

public static void main(String[] args) {

A a=new B();

a.m1();

}

}

// Output : In m1 B. ArrayIndexOutOfBoundsException and IndexOutOfBoundsException are Runtime exceptions and there is no rule for runtime exceptions while method overriding.

CASE 8:

class A

{

void m1() throws IOException

{

System.out.println("In m1 A");

}

}

class B extends A

{

void m1() throws Exception

{

System.out.println("In m1 B");

}

}

public class Test {

public static void main(String[] args) {

A a=new B();

try {

a.m1();

} catch (IOException e) {

e.printStackTrace();

}

}

}

Output: Compile time error. As IOException and Exception are checked exception, so you can not broaden the scope of Exception while method overriding.

1. **Real time example of Polymorphism? How can you code it in java?**

Suppose we have to pay electricity bill through any application say Paytm, there we do have option to pay your bill through card or netbanking or Paytm Balance.

So, we have 3 ways to perform the same task. In this case we can create three methods with same name and different parameters. This is example of method overloading.

1. **Features of Java 8**
2. Functional Interface
3. Static and default methods in interface
4. Lambda expressions
5. Stream API
6. **Can we declare class as static?**

Cannot declare top level class as static but can make inner class as static.

1. **What is static import ?**

To import static variable and method of another class.

import static java.lang.Math.PI;

//no need to refer class now

double test = PI \* 5;

1. **When Static block gets executed?**

At time of class loading

1. **Difference between Comparator and Comparable**

Comparable interface can be used to provide single way of sorting whereas Comparator interface is used to provide different ways of sorting.

For using Comparable, Class needs to implement it whereas for using Comparator we don’t need to make any change in the class.

Comparable interface is in java.lang package whereas Comparator interface is present in java.utilpackage.

1. **Difference between inner class and nested class?**

Inner class is defined inside a class which is non-static and can access all variables and methods of outer class. Inner classes are associated with object of class. Since inner class cannot have static variables inside it.

Inner class is further divided into two types:

1. Local inner class which is defined with in a method
2. Anonymous inner class which is defined in a single statement with no name and constructor.

Inner class which is static is known as nested class and can access only static members of outer class.

1. **What will be output of following program ?**

Public class A {

static void function(){

SOP(“Y”);

}

Public static void main( String[] args) {

A obj = null;

Obj.function();

}

}

Output : Y.

1. **Can we create abstract class with only concrete methods?**

Yes. Can do so to stop instance creation of that class.

1. Can an Interface implement another Interface? No
2. Can an Interface extend another Interface? Yes, can extend more than one interface.
3. Can a class be defined inside an Interface? Yes
4. Can an Interface be defined inside a class? Yes
5. Write a Fibonacci series program.