1. **What is OOPS?**

Object Oriented Programming System is the programming technique to write programs based on the real world objects. The states and behaviors of an object are represented as the member variables and methods. In OOPS programming programs are organized around objects and data rather than actions and logic.

1. **What are the advantages of OOPS concepts?**

Major advantages of OOPS programming are;

* 1. **Simplicity**: OOPS programming objects model real world objects, so the complexity is reduced and the program structure is clear.
  2. **Modularity**: Each object forms a separate entity whose internal workings are decoupled from other parts of the system.
  3. **Modifiability**: It is easy to make minor changes in the data representation or the procedures in an OO program. Changes inside a class do not affect any other part of a program, since the only public interface that the external world has to a class is through the use of methods.
  4. **Extensibility**: Adding new features or responding to changing operating environments can be solved by introducing a few new objects and modifying some existing ones.
  5. **Maintainability**: Objects can be maintained separately, making locating and fixing problems easier.
  6. **Reusability**: Objects can be reused in different programs.

1. **What is the difference between Procedural programming and OOPS?**
   1. Procedural language is based on functions but object oriented language is based on real world objects.
   2. Procedural language gives importance on the sequence of function execution but object oriented language gives importance on states and behaviors of the objects.
   3. Procedural language exposes the data to the entire program but object oriented language encapsulates the data.
   4. Procedural language follows top down programming paradigm but object oriented language follows bottom up programming paradigm.
   5. Procedural language is complex in nature so it is difficult to modify, extend and maintain but object oriented language is less complex in nature so it is easier to modify, extend and maintain.
   6. Procedural language provides less scope of code reuse but object oriented language provides more scope of code reuse.
2. **What are the core concepts of OOPS?**

OOPS core concepts are;

* 1. Abstraction
  2. Encapsulation
  3. Polymorphism
  4. Inheritance
  5. Composition
  6. Association
  7. Aggregation

1. **What is Abstraction?**

Abstraction is an OOPS concept to construct the structure of the real world objects. During this construction only the general states and behaviors are taken and more specific states and behaviors are left aside for the implementers.

1. **What is Encapsulation?**

Encapsulation is an OOPS concept to create and define the permissions and restrictions of an object and its member variables and methods. A very simple example to explain the concept is to make the member variables of a class private and providing public getter and setter methods. Java provides four types of access level modifiers: public, protected, no modifier and private.

1. **What is the difference between Abstraction and Encapsulation?**
   1. “Program to interfaces, not implementations” is the principle for Abstraction and “Encapsulate what varies” is the OO principle for Encapsulation.
   2. Abstraction provides a general structure of a class and leaves the details for the implementers. Encapsulation is to create and define the permissions and restrictions of an object and its member variables and methods.
   3. Abstraction is implemented in Java using interface and abstract class while Encapsulation is implemented using four types of access level modifiers: public, protected, no modifier and private.
2. **What is Polymorphism?**

Polymorphism is the occurrence of something in various forms. Java supports various forms of polymorphism like polymorphic reference variables, polymorphic method, polymorphic return types and polymorphic argument types.

1. **What is Inheritance?**

A subclass can inherit the states and behaviors of it’s super class is known as inheritance.

1. **What is multiple inheritance?**

A child class inheriting states and behaviors from multiple parent classes is known as multiple inheritance.

1. **What is the diamond problem in inheritance?**

In case of multiple inheritance, suppose class A has two subclasses B and C, and a class D has two super classes B and C.If a method present in A is overridden by both B and C but not by D then from which class D will inherit that method B or C? This problem is known as diamond problem.

1. **Why Java does not support multiple inheritance?**

Java was designed to be a simple language and multiple inheritance introduces complexities like diamond problem. Inheriting states or behaviors from two different type of classes is a case which in reality very rare and it can be achieved easily through an object association.

1. **What is Static Binding and Dynamic Binding?**

Static or early binding is resolved at compile time. Method overloading is an example of static binding.

Dynamic or late or virtual binding is resolved at run time. Method overriding is an example of dynamic binding.

1. **What is the meaning of “IS-A” and “HAS-A” relationship?**

“IS-A” relationship implies inheritance. A sub class object is said to have “IS-A” relationship with the super class or interface. If class A extends B then A “IS-A” B. It is transitive, that is, if class A extends B and class B extends C then A “IS-A” C. The “instanceof” operator in java determines the “IS-A” relationship.

When a class A has a member reference variable of type B then A “HAS-A” B. It is also known as Aggregation.

1. **What is Association?**

Association is a relationship between two objects with multiplicity.

1. **What is Aggregation?**

Aggregation is also known as “HAS-A” relationship. When class Car has a member reference variable of type Wheel then the relationship between the classes Car and Wheel is known as Aggregation. Aggregation can be understood as “whole to its parts” relationship.

Car is the whole and Wheel is part. Wheel can exist without the Car. Aggregation is a weak association.

1. **What is Composition?**

Composition is a special form of Aggregation where the part cannot exist without the whole. Composition is a strong Association. Composition relationship is represented like aggregation with one difference that the diamond shape is filled.

1. **What is Dependency?**

When one class depends on another because it uses that at some point in time then this relationship is known as Dependency. One class depends on another if the independent class is a parameter variable or local variable of a method of the dependent class. A Dependency is drawn as a dotted line from the dependent class to the independent class with an open arrowhead pointing to the independent class.

1. **What is the difference between Association and Dependency?**

The main difference between Association and Dependency is in case of Association one class has an attribute or member variable of the other class type but in case of Dependency a method takes an argument of the other class type or a method has a local variable of the other class type.

1. **What is a Class?**

A class is the specification or template of an object.

1. **What is an Object?**

Object is instance of class.

### 2) How many types of memory areas are allocated by JVM?

Many types:

1. Class(Method) Area
2. Heap
3. Stack
4. Program Counter Register
5. Native Method Stack

[more details...](https://www.javatpoint.com/internal-details-of-jvm)

### 3) What is JIT compiler?

**Just-In-Time(JIT) compiler:**It is used to improve the performance. JIT compiles parts of the byte code that have similar functionality at the same time, and hence reduces the amount of time needed for compilation.Here the term “compiler” refers to a translator from the instruction set of a Java virtual machine (JVM) to the instruction set of a specific CPU.

### 4) What is platform?

A platform is basically the hardware or software environment in which a program runs. There are two types of platforms software-based and hardware-based. Java provides software-based platform.

### 5) What is the main difference between Java platform and other platforms?

The Java platform differs from most other platforms in the sense that it's a software-based platform that runs on top of other hardware-based platforms.It has two components:

1. Runtime Environment
2. API(Application Programming Interface)

### 6) What gives Java its 'write once and run anywhere' nature?

The bytecode. Java is compiled to be a byte code which is the intermediate language between source code and machine code. This byte code is not platform specific and hence can be fed to any platform.

### 7) What is classloader?

The classloader is a subsystem of JVM that is used to load classes and interfaces.There are many types of classloaders e.g. Bootstrap classloader, Extension classloader, System classloader, Plugin classloader etc.

### 8) Is Empty .java file name a valid source file name?

Yes, save your java file by .java only, compile it by **javac .java** and run by **java yourclassname** Let's take a simple example:

1. //save by .java only
2. **class** A{
3. **public** **static** **void** main(String args[]){
4. System.out.println("Hello java");
5. }
6. }
7. //compile by javac .java
8. //run by     java A

compile it by **javac .java**

run it by **java A**

### 9) Is delete,next,main,exit or null keyword in java?

No.

### 10) If I don't provide any arguments on the command line, then the String array of Main method will be empty or null?

It is empty. But not null.

### 11) What if I write static public void instead of public static void?

Program compiles and runs properly.

### 12) What is the default value of the local variables?

The local variables are not initialized to any default value, neither primitives nor object references.

### 13) What is difference between object oriented programming language and object based programming language?

Object based programming languages follow all the features of OOPs except Inheritance. Examples of object based programming languages are JavaScript, VBScript etc.

### 14) What will be the initial value of an object reference which is defined as an instance variable?

The object references are all initialized to null in Java.

### 15) What is constructor?

* Constructor is just like a method that is used to initialize the state of an object. It is invoked at the time of object creation.

[more details...](https://www.javatpoint.com/constructor)

### 16) What is the purpose of default constructor?

* The default constructor provides the default values to the objects. The java compiler creates a default constructor only if there is no constructor in the class.[more details...](https://www.javatpoint.com/constructor)

### 17) Does constructor return any value?

**Ans:**yes, that is current instance (You cannot use return type yet it returns a value).[more details...](https://www.javatpoint.com/constructor)

### 18)Is constructor inherited?

No, constructor is not inherited.

### 19) Can you make a constructor final?

No, constructor can't be final.

### 20) What is static variable?

* static variable is used to refer the common property of all objects (that is not unique for each object) e.g. company name of employees,college name of students etc.
* static variable gets memory only once in class area at the time of class loading.

### 21) What is static method?

* A static method belongs to the class rather than object of a class.
* A static method can be invoked without the need for creating an instance of a class.
* static method can access static data member and can change the value of it.

### 22) Why main method is static?

because object is not required to call static method if It were non-static method,jvm creats object first then call main() method that will lead to the problem of extra memory allocation.

### 23) What is static block?

* Is used to initialize the static data member.
* It is excuted before main method at the time of classloading.

### 24) Can we execute a program without main() method?

Ans) Yes, one of the way is static block.

### 25) What if the static modifier is removed from the signature of the main method?

Program compiles. But at runtime throws an error "NoSuchMethodError".

### 26) What is difference between static (class) method and instance method?

|  |  |
| --- | --- |
| **static or class method** | **instance method** |
| 1)A method i.e. declared as static is known as static method. | A method i.e. not declared as static is known as instance method. |
| 2)Object is not required to call static method. | Object is required to call instance methods. |
| 3)Non-static (instance) members cannot be accessed in static context (static method, static block and static nested class) directly. | static and non-static variables both can be accessed in instance methods. |
| 4)For example: public static int cube(int n){ return n\*n\*n;} | For example: public void msg(){...}. |

### 27) What is this in java?

It is a keyword that that refers to the current object.

### 28)What is Inheritance?

Inheritance is a mechanism in which one object acquires all the properties and behaviour of another object of another class. It represents IS-A relationship. It is used for Code Resusability and Method Overriding.

### 29) Which class is the superclass for every class.

Object class.

### 30) Why multiple inheritance is not supported in java?

* To reduce the complexity and simplify the language, multiple inheritance is not supported in java in case of class.

### 31) What is composition?

Holding the reference of the other class within some other class is known as composition.

### 32) What is difference between aggregation and composition?

Aggregation represents weak relationship whereas composition represents strong relationship. For example: bike has an indicator (aggregation) but bike has an engine (compostion).

### 33) Why Java does not support pointers?

Pointer is a variable that refers to the memory address. They are not used in java because they are unsafe(unsecured) and complex to understand.

### 34) What is super in java?

It is a keyword that refers to the immediate parent class object.[more details...](https://www.javatpoint.com/super-keyword)

### 35) Can you use this() and super() both in a constructor?

No. Because super() or this() must be the first statement.

### 36)What is object cloning?

The object cloning is used to create the exact copy of an object. [more details...](https://www.javatpoint.com/object-cloning)

### 37) What is method overloading?

If a class have multiple methods by same name but different parameters, it is known as Method Overloading. It increases the readability of the program.[more details...](https://www.javatpoint.com/method-overloading-in-java)

### 38) Why method overloading is not possible by changing the return type in java?

Becauseof ambiguity.[more details...](https://www.javatpoint.com/method-overloading-in-java)

### 39) Can we overload main() method?

Yes, You can have many main() methods in a class by overloading the main method.

### 40) What is method overriding:

If a subclass provides a specific implementation of a method that is already provided by its parent class, it is known as Method Overriding. It is used for runtime polymorphism and to provide the specific implementation of the method.[more details...](https://www.javatpoint.com/method-overriding-in-java)

### 41) Can we override static method?

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

### 42) Why we cannot override static method?

It is because the static method is the part of class and it is bound with class whereas instance method is bound with object and static gets memory in class area and instance gets memory in heap.

### 43) Can we override the overloaded method?

Yes.

### 44) Difference between method Overloading and 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 super class. |
| 2) method overlaoding is occurs within the class. | Method overriding occurs in two classes that have IS-A relationship. |
| 3) In this case, parameter must be different. | In this case, parameter must be same. |

### 45) Can you have virtual functions in Java?

Yes, all functions in Java are virtual by default.

### 46) What is covariant return type?

Now, since java5, it is possible to override any method by changing the return type if the return type of the subclass overriding method is subclass type. It is known as covariant return type. [more details...](https://www.javatpoint.com/covariant-return-type)

### 47) What is final variable?

If you make any variable as final, you cannot change the value of final variable(It will be constant

### 48) What is final method?

Final methods can't be overriden.[more details...](https://www.javatpoint.com/final-keyword)

### 49) What is final class?

Final class can't be inherited.

### 50) What is blank final variable?

A final variable, not initalized at the time of declaration, is known as blank final variable.

### 51) Can we intialize blank final variable?

Yes, only in constructor if it is non-static. If it is static blank final variable, it can be initialized only in the static block.

### 52) Can you declare the main method as final?

Yes, such as, public static final void main(String[] args){}.

### What is Runtime Polymorphism?

Runtime polymorphism or dynamic method dispatch is a process in which a call to an overridden method is resolved at runtime rather than at compile-time.

In this process, an overridden method is called through the reference variable of a super class. The determination of the method to be called is based on the object being referred to by the reference variable.

### 54) Can you achieve Runtime Polymorphism by data members?

No.

### 55) What is the difference between static binding and dynamic binding?

In case of static binding type of object is determined at compile time whereas in dynamic binding type of object is determined at runtime.

### 56) What is abstraction?

Abstraction is a process of hiding the implementation details and showing only functionality to the user.

Abstraction lets you focus on what the object does instead of how it does it.

### 57) What is the difference between abstraction and encapsulation?

Abstraction hides the implementation details whereas encapsulation wraps code and data into a single unit.

### 58) What is abstract class?

A class that is declared as abstract is known as abstract class. It needs to be extended and its method implemented. It cannot be instantiated.

### 59) Can there be any abstract method without abstract class?

No, if there is any abstract method in a class, that class must be abstract.

### 60) Can you use abstract and final both with a method?

No, because abstract method needs to be overridden whereas you can't override final method.

### 61) Is it possible to instantiate the abstract class?

No, abstract class can never be instantiated.

### 62) What is interface?

Interface is a blueprint of a class that have static constants and abstract methods.It can be used to achieve fully abstraction and multiple inheritance.

### 63) Can you declare an interface method static?

No, because methods of an interface is abstract by default, and static and abstract keywords can't be used together.

### 64) Can an Interface be final?

No, because its implementation is provided by another class.

### 65) What is marker interface?

An interface that have no data member and method is known as a marker interface.For example Serializable, Cloneable etc.

### 66) What is difference between abstract class and interface?

|  |  |
| --- | --- |
| **Abstract class** | **Interface** |
| 1)An abstract class can have method body (non-abstract methods). | Interface have only abstract methods. |
| 2)An abstract class can have instance variables. | An interface cannot have instance variables. |
| 3)An abstract class can have constructor. | Interface cannot have constructor. |
| 4)An abstract class can have static methods. | Interface cannot have static methods. |
| 5)You can extends one abstract class. | You can implement multiple interfaces. |

### 67) Can we define private and protected modifiers for variables in interfaces?

No, they are implicitly public.

### 68) When can an object reference be cast to an interface reference?

An object reference can be cast to an interface reference when the object implements the referenced interface.

### 69) What is package?

A package is a group of similar type of classes interfaces and sub-packages. It provides access protection and removes naming collision.

[more details...](https://www.javatpoint.com/package)

### 70) Do I need to import java.lang package any time? Why ?

No. It is by default loaded internally by the JVM.

### 71) Can I import same package/class twice? Will the JVM load the package twice at runtime?

One can import the same package or same class multiple times. Neither compiler nor JVM complains about it.But the JVM will internally load the class only once no matter how many times you import the same class.

### 72) What is static import ?

By static import, we can access the static members of a class directly, there is no to qualify it with the class name.

### 73) What is Exception Handling?

Exception Handling is a mechanism to handle runtime errors.It is mainly used to handle checked exceptions.

### 74) What is difference between Checked Exception and Unchecked Exception?

### 1)Checked Exception

The classes that extend Throwable class except RuntimeException and Error are known as checked exceptions e.g.IOException,SQLException etc. Checked exceptions are checked at compile-time.

### 2)Unchecked Exception

The classes that extend RuntimeException are known as unchecked exceptions e.g. ArithmeticException,NullPointerException etc. Unchecked exceptions are not checked at compile-time.

### 75) What is the base class for Error and Exception?

Throwable.

### 76) Is it necessary that each try block must be followed by a catch block?

It is not necessary that each try block must be followed by a catch block. It should be followed by either a catch block OR a finally block. And whatever exceptions are likely to be thrown should be declared in the throws clause of the method.

### 77) What is finally block?

* finally block is a block that is always executed.[more details...](https://www.javatpoint.com/finally-block-in-exception-handling)

### 78) Can finally block be used without catch?

* Yes, by try block. finally must be followed by either try or catch.[more details...](https://www.javatpoint.com/finally-block-in-exception-handling)

### 79) Is there any case when finally will not be executed?

finally block will not be executed if program exits(either by calling System.exit() or by causing a fatal error that causes the process to abort).[more details...](https://www.javatpoint.com/finally-block-in-exception-handling)

### 80) What is difference between throw and throws?

|  |  |
| --- | --- |
| **throw keyword** | **throws keyword** |
| 1)throw is used to explicitly throw an exception. | throws is used to declare an exception. |
| 2)checked exceptions can not be propagated with throw only. | checked exception can be propagated with throws. |
| 3)throw is followed by an instance. | throws is followed by class. |
| 4)throw is used within the method. | throws is used with the method signature. |
| 5)You cannot throw multiple exception | You can declare multiple exception e.g. public void method()throws IOException,SQLException. |

[more details...](https://www.javatpoint.com/throws-keyword-and-difference-between-throw-and-throws)

### 81) Can an exception be rethrown?

Yes.

### 82) Can subclass overriding method declare an exception if parent class method doesn't throw an exception ?

Yes but only unchecked exception not checked.

[more details...](https://www.javatpoint.com/exception-handling-with-method-overriding)

### 83) What is exception propagation ?

Forwarding the exception object to the invoking method is known as exception propagation.

### 84) What is the meaning of immutable in terms of String?

The simple meaning of immutable is unmodifiable or unchangeable. Once string object has been created, its value can't be changed.

[more details...](https://www.javatpoint.com/immutable-string)

### 85) Why string objects are immutable in java?

Because java uses the concept of string literal. Suppose there are 5 reference variables,all referes to one object "sachin".If one reference variable changes the value of the object, it will be affected to all the reference variables. That is why string objects are immutable in java.

[more details...](https://www.javatpoint.com/immutable-string)

### 86) How many ways we can create the string object?

There are two ways to create the string object, by string literal and by new keyword.

[more details...](https://www.javatpoint.com/string-handling-in-java)

### 87) How many objects will be created in the following code?

1. String s1="Welcome";
2. String s2="Welcome";
3. String s3="Welcome";

Only one object.

[more details...](https://www.javatpoint.com/string-handling-in-java)

### 88) Why java uses the concept of string literal?

To make Java more memory efficient (because no new objects are created if it exists already in string constant pool).

[more details...](https://www.javatpoint.com/string-handling-in-java)

### 89)How many objects will be created in the following code?

1. String s = **new** String("Welcome");

Two objects, one in string constant pool and other in non-pool(heap).

[more details...](https://www.javatpoint.com/string-handling-in-java)

### 90) What is the basic difference between string and stringbuffer object?

String is an immutable object. StringBuffer is a mutable object.

### 91) What is the difference between StringBuffer and StringBuilder ?

StringBuffer is synchronized whereas StringBuilder is not synchronized.

### 92) How can we create immutable class in java ?

We can create immutable class as the String class by defining final class and

### 93) What is the purpose of toString() method in java ?

The toString() method returns the string representation of any object. If you print any object, java compiler internally invokes the toString() method on the object. So overriding the toString() method, returns the desired output, it can be the state of an object etc. depends on your implementation.

### 94)What is nested class?

A class which is declared inside another class is known as nested class. There are 4 types of nested class member inner class, local inner class, annonymous inner class and static nested class.

### 95) Is there any difference between nested classes and inner classes?

Yes, inner classes are non-static nested classes i.e. inner classes are the part of nested classes.

### 96) Can we access the non-final local variable, inside the local inner class?

No, local variable must be constant if you want to access it in local inner class.

### 97) What is nested interface ?

Any interface i.e. declared inside the interface or class, is known as nested interface. It is static by default.

### 98) Can a class have an interface?

Yes, it is known as nested interface.

### 99) Can an Interface have a class?

Yes, they are static implicitely.

### What is Garbage Collection?

Garbage collection is a process of reclaiming the runtime unused objects.It is performed for memory management.

### 118) What is gc()?

gc() is a daemon thread.gc() method is defined in System class that is used to send request to JVM to perform garbage collection.

### 119) What is the purpose of finalize() method?

finalize() method is invoked just before the object is garbage collected.It is used to perform cleanup processing.

### 120) Can an unrefrenced objects be refrenced again?

Yes.

### 121)What kind of thread is the Garbage collector thread?

Daemon thread.

### 122)What is difference between final, finally and finalize?

|  |
| --- |
| **final:** final is a keyword, final can be variable, method or class.You, can't change the value of final variable, can't override final method, can't inherit final class. |
| **finally:** finally block is used in exception handling. finally block is always executed. |
| **finalize():**finalize() method is used in garbage collection.finalize() method is invoked just before the object is garbage collected.The finalize() method can be used to perform any cleanup processing. |

### 123)What is the purpose of the Runtime class?

The purpose of the Runtime class is to provide access to the Java runtime system.

### 124)How will you invoke any external process in Java?

By Runtime.getRuntime().exec(?) method.

### 125)What is the difference between the Reader/Writer class hierarchy and the InputStream/OutputStream class hierarchy?

The Reader/Writer class hierarchy is character-oriented, and the InputStream/OutputStream class hierarchy is byte-oriented.

### 126)What an I/O filter?

An I/O filter is an object that reads from one stream and writes to another, usually altering the data in some way as it is passed from one stream to another.

### 127) What is serialization?

Serialization is a process of writing the state of an object into a byte stream.It is mainly used to travel object's state on the network.

[more details...](https://www.javatpoint.com/serialization)

### 128) What is Deserialization?

Deserialization is the process of reconstructing the object from the serialized state.It is the reverse operation of serialization.

### 129) What is transient keyword?

If you define any data member as transient,it will not be serialized.[more details...](https://www.javatpoint.com/serialization)

### 130)What is Externalizable?

Externalizable interface is used to write the state of an object into a byte stream in compressed format.It is not a marker interface.

### 131)What is the difference between Serializalble and Externalizable interface?

Serializable is a marker interface but Externalizable is not a marker interface.When you use Serializable interface, your class is serialized automatically by default. But you can override writeObject() and readObject() two methods to control more complex object serailization process. When you use Externalizable interface, you have a complete control over your class's serialization process.

### 132)How do I convert a numeric IP address like 192.18.97.39 into a hostname like java.sun.com?

By InetAddress.getByName("192.18.97.39").getHostName() where 192.18.97.39 is the IP address.

### 133) What is reflection?

Reflection is the process of examining or modifying the runtime behaviour of a class at runtime.It is used in:

* IDE (Integreted Development Environment) e.g. Eclipse, MyEclipse, NetBeans.
* Debugger
* Test Tools etc.

### 134) Can you access the private method from outside the class?

Yes, by changing the runtime behaviour of a class if the class is not secured.

### 148)What are wrapper classes?

Wrapper classes are classes that allow primitive types to be accessed as objects.

### 149)What is a native method?

A native method is a method that is implemented in a language other than Java.

### 150)What is the purpose of the System class?

The purpose of the System class is to provide access to system resources.

### 151)What comes to mind when someone mentions a shallow copy in Java?

Object cloning.

### 152)What is singleton class?

Singleton class means that any given time only one instance of the class is present, in one JVM.

### 153)Which containers use a border layout as their default layout?

The Window, Frame and Dialog classes use a border layout as their default layout.

### 154)Which containers use a FlowLayout as their default layout?

The Panel and Applet classes use the FlowLayout as their default layout.

### 155)What are peerless components?

The peerless components are called light weight components.

### 156)is the difference between a Scrollbar and a ScrollPane?

A Scrollbar is a Component, but not a Container. A ScrollPane is a Container. A ScrollPane handles its own events and performs its own scrolling.

### 157)What is a lightweight component?

Lightweight components are the one which doesn?t go with the native call to obtain the graphical units. They share their parent component graphical units to render them. For example, Swing components.

### 158)What is a heavyweight component?

For every paint call, there will be a native call to get the graphical units.For Example, AWT.

### 159)What is an applet?

An applet is a small java program that runs inside the browser and generates dynamic contents.

### 160)Can you write a Java class that could be used both as an applet as well as an application?

Yes. Add a main() method to the applet.

### 161)What is Locale?

A Locale object represents a specific geographical, political, or cultural region.

### 162)How will you load a specific locale?

By ResourceBundle.getBundle(?) method.

### 163)What is a JavaBean?

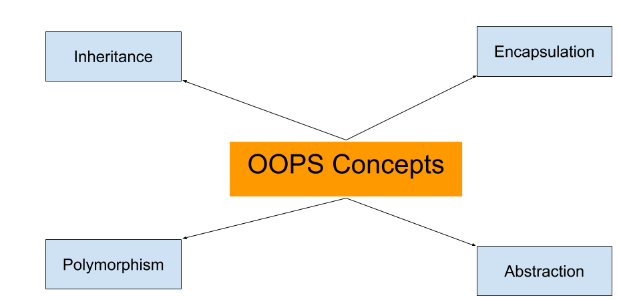
are reusable software components written in the Java programming language, designed to be manipulated visually by a software development environment, like JBuilder or VisualAge for Java.

### 164)Can RMI and Corba based applications interact?

Yes they can. RMI is available with IIOP as the transport protocol instead of JRMP.

**Q) What are different oops concept in java?**

Ans)



The different OOps concepts are :

* [Polymorphism](http://java-questions.com/oops-interview-questions.html#polymorphism-java)
* [Inheritance](http://java-questions.com/oops-interview-questions.html#inheritance-in-java)
* [Abstraction](http://java-questions.com/oops-interview-questions.html#abstraction)
* [Encapsulation](http://java-questions.com/oops-interview-questions.html#encapsulation)
* [Aggreagation](http://java-questions.com/oops-interview-questions.html#aggregation)
* [Composition](http://java-questions.com/oops-interview-questions.html#composition)
* [Association](http://java-questions.com/oops-interview-questions.html#association)

**Q1) What is polymorphism?**

Ans) The ability to identify a function to run is called Polymorphism. In java, c++ there are two types of polymorphism: compile time polymorphism (overloading) and runtime polymorphism (overriding).

**Method overriding:**Overriding occurs when a class method in a child class has the same name and signature as a method in the parent class. When you override methods, JVM determines the proper method to call at the program’s run time, not at the compile time.

**Overloading:**Overloading is determined at the compile time. It occurs when several methods have same names with:

* Different method signature and different number or type of parameters.
* Same method signature but the different number of parameters.
* Same method signature and same number of parameters but of different type

Example of Overloading

int add(int a,int b)

float add(float a,int b)

float add(int a ,float b)

void add(float a)

int add(int a)

void add(int a) //error conflict with the method int add(int a)

class BookDetails {

String title;

setBook(String title){}

}

class ScienceBook extends BookDetails {

setBook(String title){} //overriding

setBook(String title, String publisher,float price){} //overloading

}

**Q2) What is inheritance?**

Ans) Inheritance allows a Child class to inherit properties from its parent class. In Java this is achieved by using **extends** keyword. Only properties with access modifier public and protected can be accessed in child class.

public class Parent {

public String parentName;

public String familyName;

protected void printMyName() {

System.out.println(“ My name is “+ chidName+” “ +familyName);

}

}

public class Child extends Parent {

public String childName;

public int childAge;

//inheritance

protected void printMyName() {

System.out.println(“ My child name is “+ chidName+” “ +familyName);

}

}

In a example above the child has inherited its family name from the parent class just by inheriting the class. When a child object is created the method **printMyName()** present in the child class is called.

**Q3) What is multiple inheritance and does java support?**

Ans) If a child class inherits the property from multiple classes is known as multiple inheritance. Java does not allow to extend multiple classes. The problem with the multiple inheritance is that if multiple parent classes have methods with same name, then at runtime it becomes difficult for the compiler to decide which method to execute from the child class. To overcome this problem java allows to implement multiple Interfaces. The problem is commonly referred as [What is Diamond Problem.](http://java-questions.com/keyConcepts-interview-questions.html#diamond-problem)

**Q) What is the difference between polymorphism and inheritance?**

* Inheritance defines parent-child relationship between two classes, polymorphism takes advantage of that relationship to add dynamic behavior in your code.
* Inheritance encourages code reusability by allowing child class to inherit behavior from the parent class. On the other hand Polymorphism allows child to redefine already defined behaviour inside the parent class. Without Polymorphism it's not possible for a child to execute its own behaviour while represented by a Parent reference variable, but with Polymorphism it can be done.
* Java doesn't allow multiple inheritance of classes, but allows [multiple inheritance of Interface](http://java-questions.com/keyConcepts-interview-questions.html#diamond-problem), which is actually required to implement Polymorphism. For example, a class can be Runnable, Comparator and Serializable at the same time because all three are interfaces. This makes them pass around in code e.g. you can pass an instance of this class to a method which accepts Serializable, or to Collections.sort() which accepts a Comparator.
* Both Polymorphism and Inheritance allow Object oriented programs to evolve. For example, by using Inheritance you can define new user types in an Authentication System and by using Polymorphism you can take advantage of already written authentication code. Since, Inheritance guarantees minimum base class behaviour, a method depending upon super class or super interface can still accept an object of the base class and can authenticate it.

**Q4) What is an abstraction?**

Ans) Abstraction is a way of converting real world objects in terms of class. It's a concept of defining an idea in terms of classes or interface. For example creating a class Vehicle and injecting properties into it. E.g

public class Vehicle {

public String colour;

public String model;}

**Q5) What is Encapsulation?**

Ans) The encapsulation is achieved by combining the methods and attribute into a class. The class acts like a container encapsulating the properties. The users are exposed mainly public methods.The idea behind is to hide how things work and just exposing the requests a user can do.

**Q6) What is Association?**

Ans) 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 a 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.

**Q7) What is Aggregation?**

Ans) Aggregation is a specialized form of Association where all objects 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 cannot belong to multiple departments, but if we delete the department teacher object will not destroy. We can think about "has-a" relationship.

**Q8) What is Composition?**

Ans) Composition is specialized form of Aggregation and we can call this as a "death" relationship. It is a strong type of Aggregation. Child object does not have their lifecycle and if parent object deletes all child object will also be deleted. Let's take again an example of a 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.

**What is method overloading in OOP or Java?**([answer](http://java67.blogspot.sg/2012/08/what-is-method-overloading-in-java-example.html))  
It's one of the oldest OOPS concept questions, I have seen it 10 years ago and still sees it now. When we have multiple methods with the same name but different functionality then it's called method overloading. For example. System.out.println() is overloaded as we have 6 or 7 println() method each accepting a different type of parameter.  
  
**What is method overriding in OOP or Java?** ([answer](http://java67.blogspot.sg/2012/08/what-is-method-overriding-in-java-example-tutorial.html))  
It's one of the magic of object oriented programming where the method is chose based upon an object at runtime. In order for method overriding, we need Inheritance and Polymorphism, as we need a method with the same signature in both superclass and subclass. A call to such method is resolved at runtime depending upon the actual object and not the type o variable. See the answer for more detailed discussion.  
  
**What is method hiding in Java?**(answer)  
When you declare two static methods with same name and signature in both superclass and subclass then they hide each other i.e. a call to the method in the subclass will call the static method declared in that class and a call to the same method is superclass is resolved to the static method declared in the super class.  
  
**Is Java a pure object oriented language? if not why?** ([answer](http://java67.blogspot.com/2014/03/is-java-pure-object-oriented-programming-language.html))  
Java is not a pure object-oriented programming language e.g. there are many things you can do without objects e.g. static methods. Also, primitive variables are not objects in Java. See the answer for a more detailed explanation.  
  
**What are rules of method overloading and overriding in Java?**([answer](http://java67.blogspot.sg/2012/09/what-is-rules-of-overloading-and-overriding-in-java.html))  
One of the most important rules of method overloading in Java is that method signature should be different i.e. either the number of arguments or the type of arguments. Simply changing the return type of two methods will not result in overloading, instead, the compiler will throw an error. On the other hand, method overriding has more rules e.g. name and return type must be same, method signature should also be same, the overloaded method cannot throw a higher exception etc. See the answer for a full list of rules related to method overloading and overriding in Java.  
  
**The difference between method overloading and overriding?** ([answer](http://java67.blogspot.sg/2012/09/difference-between-overloading-vs-overriding-in-java.html))  
Several differences but the most important one is that method overloading is resolved at compile time and method overriding is resolved at runtime. The compiler only used the class information for method overloading, but it needs to know object to resolved overridden method calls. This diagram explains the difference quite well, though:

**Can we overload a static method in Java?** ([answer](http://java67.blogspot.sg/2012/08/can-we-overload-static-method-in-java.html))  
Yes, you can overload a static method in Java. You can declare as many static methods of the same name as you wish provided all of them have different method signatures. See the answer for more detailed explanation and code example.  
  
**Can we override static method in Java?**([answer](http://java67.blogspot.sg/2012/08/can-we-override-static-method-in-java.html))  
No, you cannot override a static method because it's not bounded to an object. Instead, static methods belong to a class and resolved at compile time using the type of reference variable. But, Yes, you can declare the same static method in a subclass, that will result in method hiding i.e. if you use reference variable of type subclass then new method will be called, but if you use reference variable of superclass than old method will be called.

**Can we prevent overriding a method without using the final modifier?** (answer)  
Yes, you can prevent the method overriding in Java without using the final modifier. In fact, there are several ways to accomplish it e.g. you can mark the method private or static, those cannot be overridden.  
  
**Can we override a private method in Java?**([answer](http://java67.blogspot.sg/2013/08/can-we-override-private-method-in-java-inner-class.html))  
No, you cannot. Since the private method is only accessible and visible inside the class they are declared, it's not possible to override them in subclasses. Though, you can override them inside the inner class as they are accessible there.  
  
**What is covariant method overriding in Java?**([answer](http://javarevisited.blogspot.com/2014/03/covariant-method-overriding-of-java-5.html))  
In covariant method overriding, the overriding method can return the subclass of the object returned by original or overridden method. This concept was introduced in Java 1.5 (Tiger) version and it's very helpful in case original method is returning general type like Object class, because, then by using covariant method overriding you can return more suitable object and prevent client side type casting. One of the practical use of this concept is in when you override the clone() method in Java.  
  
**Can we change the return type of method to subclass while overriding?**(answer)  
Yes, you can, but only from Java 5 onward. This feature is known as covariant method overriding and it was introduced in JDK 5 release. This is immensely helpful if original method return super-class e.g. clone() method return java.lang.Object. By using this, you can directly return the actual type, preventing client-side type casting of the result.

**Can we change the argument list of an overriding method?**([answer](http://javarevisited.blogspot.com/2011/08/what-is-polymorphism-in-java-example.html))  
No, you cannot. The argument list is part of the method signature and both overriding and overridden method must have the same signature.  
  
**Can we override a method which throws runtime exception without throws clause?** ([answer](http://javarevisited.blogspot.sg/2011/12/method-overloading-vs-method-overriding.html))  
Yes, there is no restriction on unchecked exception while overriding. On the other hand, in the case of checked exception, an overriding exception cannot throw a checked exception which comes higher in type hierarchy e.g. if original method is throwing IOException than overriding method cannot throw java.lang.Exception or java.lang.Throwable.  
  
**How do you call superclass version of an overriding method in sub class? (**answer**)**  
You can call a superclass version of an overriding method in the subclass by using super keyword. For example to call the toString() method from java.lang.Object class you can call super.toString().  
  
**Can we override a non-static method as static in Java?**(answer)  
Yes, you can override the non-static method in Java, no problem on them but it should not be private or final :)  
  
**Can we override the final method in Java?** ([answer](http://javarevisited.blogspot.com/2013/12/when-to-make-method-final-in-java.html))  
No, you cannot override a final method in Java, final keyword with the method is to prevent method overriding. You use final when you don't want subclass changing the logic of your method by overriding it due to security reason. This is [why String class is final in Java](http://java67.blogspot.com/2014/01/why-string-class-has-made-immutable-or-final-java.html). This concept is also used in template design pattern where template method is made final to prevent overriding.  
  
**Can we have a non-abstract method inside interface?** (answer)  
From Java 8 onward you can have a non-abstract method inside interface, prior to that it was not allowed as all method was implicitly public abstract. From JDK 8, you can add static and default method inside an interface.  
  
**What is the default method of Java 8?** ([answer](http://javarevisited.blogspot.com/2014/07/default-defender-or-extension-method-of-Java8-example-tutorial.html))  
Default method, also known as extension method are new types of the method which you can add on the interface now. These method has implementation and intended to be used by default. By using this method, JDK 8 managed to provide common functionality related to [lambda expression](http://javarevisited.blogspot.com/2014/02/10-example-of-lambda-expressions-in-java8.html) and [stream API](http://javarevisited.blogspot.com/2014/03/2-examples-of-streams-with-Java8-collections.html)without breaking all the clients which implement their interfaces. If you look Java 8 API documentation you will find several useful default method on key Java interface like Iterator, Map etc.  
  
**What is an abstract class in Java? (**[answer](http://java67.blogspot.sg/2014/06/why-abstract-class-is-important-in-java.html)**)**  
An abstract class is a class which is incomplete. You cannot create an instance of an abstract class in Java. They are provided to define default behavior and ensured that client of that class should adore to those contract which is defined inside the abstract class. In order to use it, you must extend and implement their abstract methods. BTW, in Java, a class can be abstract without specifying any abstract method.  
  
**What is an interface in Java? What is the real user of an interface?**([answer](http://java67.blogspot.sg/2014/02/what-is-actual-use-of-interface-in-java.html))  
Like an abstract class, the interface is also there to specify the contract of an API. It supports OOP abstraction concept as it defines only abstract behavior. It will tell that your program will give output but how is left to implementors. The real use of the interface to define types to leverage Polymorphism. See the answer for more detailed explanation and discussion.  
  
**The difference between Abstract class and interface?** ([answer](http://java67.blogspot.sg/2012/09/what-is-difference-between-interface-abstract-class-java.html))  
In Java, the key difference is that abstract class can contain non-abstract method but the interface cannot, but from Java 8 onward interface can also contain static and default methods which are non-abstract. See the answer for more detailed discussion as I have described a lot of points there.

**Can we make a class abstract without an abstract method?** ([answer](http://javarevisited.blogspot.com/2013/04/10-abstract-class-and-interface-interview-question-java-answers.html))  
Yes, just add abstract keyword on the class definition and your class will become abstract.  
  
**Can we make a class both final and abstract at the same time?** ([answer](http://javarevisited.blogspot.com/2011/12/final-variable-method-class-java.html))  
No, you cannot apply both final and abstract keyword at the class same time because they are exactly opposite of each other. A final class in Java cannot be extended and you cannot use an abstract class without extending and make it a concrete class. As per Java specification, the compiler will throw an error if you try to make a class abstract and final at the same time.  
  
**Can we overload or override the main method in Java?** ([answer](http://java67.blogspot.com/2015/06/can-you-overload-or-override-main-in-java.html))  
No, since main() is a static method, you can only overload it, you cannot override it because the static method is resolved at compile time without needing object information hence we cannot override the main method in Java.  
  
**What is the difference between Polymorphism, Overloading, and Overriding?** ([answer](http://java67.blogspot.sg/2012/10/difference-between-polymorphism-overloading-overriding-java.html))  
This is slight tricky OOP concept question because Polymorphism is the real concept behind on both Overloading and Overriding. Overloading is compiled time Polymorphism and Overriding are Runtime Polymorphism.  
  
**Can an interface extend more than one interface in Java?**  
Yes, an interface can extend more than one interface in Java, it's perfectly valid.  
  
**Can a class extend more than one class in Java?**  
No, a class can only extend another class because Java doesn't support multiple inheritances but yes, it can implement multiple interfaces.  
  
**What is the difference between abstraction and polymorphism in Java?** ([answer](http://java67.blogspot.sg/2015/05/difference-between-abstraction-and.html))  
Abstraction generalizes the concept and Polymorphism allow you to use different implementation without changing your code. This diagram explains the abstraction quite well, though:

## Object Oriented design principle and pattern Interview Questions

Now let's see some OOPS concept questions based on the SOLID design principles and GOF design patterns which take advantage of OOPS concept discussed here.

**What problem is solved by Strategy pattern in Java?** ([answer](http://java67.blogspot.com/2014/12/strategy-pattern-in-java-with-sample.html))

Strategy pattern allows you to introduce new algorithm or new strategy without changing the code which uses that algorithm. For example, the Collections.sort() method which sorts the list of the object uses the Strategy pattern to compare object. Since every object uses different comparison strategy you can compare various object differently without changing sort method.

**Which OOP concept Decorator design Pattern is based upon?**([answer](http://java67.blogspot.com/2013/07/decorator-design-pattern-in-java-real-life-example-tutorial.html))

Decorator pattern takes advantage of Composition to provide new features without modifying the original class. A very good to-the-point question for the telephonic round. This is quite clear from UML diagram of Decorator pattern, as you can see the Component is associated with Decorator.

**When to use Singleton design pattern in Java?** ([answer](http://java67.blogspot.com/2012/08/what-is-singleton-pattern-in-java.html))  
When you need just one instance of a class and wants that to be globally available then you can use [Singleton pattern](http://javarevisited.blogspot.com/2011/03/10-interview-questions-on-singleton.html). It's not free of cost though because it increases coupling between classes and makes them hard to test. This is one of the oldest design pattern questions from Java interviews. Please see the answer for more detailed discussion.

**What is the difference between State and Strategy Pattern?**([answer](http://javarevisited.blogspot.com/2014/04/difference-between-state-and-strategy-design-pattern-java.html))  
Though the structure or class diagram of State and Strategy pattern is same, their intent is completely different. State pattern is used to do something specific depending upon state while [Strategy](http://java67.blogspot.com/2014/12/strategy-pattern-in-java-with-sample.html) allows you to switch between algorithms without changing the code which uses it.

**What is the difference between Association, Aggregation, and Composition in OOP?** ([answer](http://javarevisited.blogspot.com/2014/02/ifference-between-association-vs-composition-vs-aggregation.html))  
When an object is related to another object it called association. It has two forms, aggregation, and composition. the former is the loose form of association where the related object can survive individually while later is a stronger form of association where a related object cannot survive individually. For example, the city is an aggregation of people but is the composition of body parts.

**What is the difference between Decorator, Proxy and Adapter pattern in Java?**([answer](http://javarevisited.blogspot.com/2015/01/adapter-vs-decorator-vs-facade-vs-proxy-pattern-java.html))  
Again they look similar because their structure or class diagram is very similar but their intent is quite different. Decorator adds additional functionality without touching the class, Proxy provides access control and Adapter is used to make two incompatible interfaces work together.

**What is the 5 objects oriented design principle from SOLID?**([answer](http://javarevisited.blogspot.com/2012/03/10-object-oriented-design-principles.html))  
SOLID is the term given by Uncle Bob in his classic book, the [Clean Code](http://www.amazon.com/Clean-Code-Handbook-Software-Craftsmanship/dp/0132350882?tag=javamysqlanta-20), one of the must-read books for programmers. In SOLID each character stands for one design principle:  
S for Single Responsibility Principle  
O for Open closed design principle  
L for Liskov substitution principle  
I for Interface segregation principle  
D for Dependency inversion principle

**What is the difference between Composition and Inheritance in OOP?**([answer](http://javarevisited.blogspot.sg/2015/06/difference-between-inheritance-and-Composition-in-Java-OOP.html))

This is another great OOPS concept question because it tests what matters, both of them are very important from a class design perspective. Though both Composition and Inheritance allows you to reuse code, formerly is more flexible than later. Composition allows the class to get an additional feature at runtime, but Inheritance is static. You can not change the feature at runtime by substitution new implementation. See the answer for more detailed discussion.

### What is Object Oriented Programming? What are the advantages of OOPs?

Object Oriented Programming is a technique which combines data and operations/behavior into a single unit called object.  Object-oriented programming helps modelling complicated systems of real world into manageable software solutions. In OOPS programming programs are organized around objects rather than actions and logic.

The following are advantages of OOPs.  
–**Encapsulation**  
–**Inheritance**  
–**Polymorphism  
– Abstraction**

### What is Abstraction?

In **object-oriented** programming, **abstraction** is the technique through, a programmer hides all but the relevant data about an object in order to reduce complexity and increase efficiency. Basically by this programmer hides the internal details and shows only relevant information to the rest of the world.

### How to achieve Abstraction?

Abstraction can be achieved using abstract class and interface. If you have partial implementation then go for abstract class. Before Java 8, it was not possible to provide non-abstract method(method with definition) within the interface.  
From Java 8 you can define default and static methods with the interface.

### What is Encapsulation?

Encapsulation is one of the four fundamental OOP concepts. The other three are inheritance, polymorphism, and abstraction. It is a mechanism of wrapping the data (variables) and methods together as a single unit.

### How to achieve Encapsulation?

Encapsulation can be achieved by using access modifiers **private**,**public**,**protected**. You can restrict the access as per your need. Like we can make data member private to be accessible through class methods only. No one can access private data members outside the class.

### What is Polymorphism?

It is derived from 2 greek words **poly** and **morphs**. The word “poly” means many and “morphs” means forms. So polymorphism means many forms.

Polymorphism is one of the major feature that allows us to perform a single operation in different ways. For example, we can send text or MMS message through mobile phone with single method ***sendMessage()***.

### What is Inheritance?

Inheritance is one of the major feature of OOPs which facilitates the code reusability. Through inheritance subclass inherits the data and methods from its parent class. Java supports only single inheritance means a class can extend only single class. Java does not support multiple inheritance.

### What are access modifiers allowed at class level?

***final***, ***abstract*** and ***public*** modifiers are allowed at class level.

### Why Java does not support Multiple Inheritance?

Java does not support multiple inheritance due to ***Diamond Problem***. In case of multiple inheritance, suppose class X has two subclasses Y and Z, and a class W has two super classes Y and Z. If a method present in X is overridden by both Y and Z but not by W then from which class W will inherit that method Y or Z? This problem is known as diamond problem.

### What is Static Binding and Dynamic Binding?

Static or early binding is resolved at compile time. Method overloading is an example of static binding. Dynamic or late or virtual binding is resolved at run time. Method overriding is an example of dynamic binding.

### What is Method hiding?

When you declare two static methods with same name and signature in both super class and subclass then they hide each other i.e. a call to the method in the subclass will call the static method declared in that class and a call to the same method is super class is resolved to the static method declared in the super class.

### What are “IS-A” and “HAS-A” relationships?

When a class B extends class A then it is called IS-A relationship where B extends the properties and methods of its parent class. If class keeps the reference of some other class as its member then it is called HAS-A relationship. Like Class B has class A reference as member.

### What is Method Overloading and Overriding?

Method overloading is the process by which we can define more than one method with different signature within the class. It is done at compile time. Declaring a method in subclass which is already present in the parent class is known as method overriding. Overridden method must have same method signature as in parent class.

### What are the rules for Method Overloading?

To overload methods you must change the signature of methods. Signature of method contains types of arguments and order of arguments. Following are the rules to overload a method.  
-Change the method signature  
-Method can’t be overloaded by changing the return type of method.

### What are the rules of Method Overriding?

A overridden method should have same signature. The return type of overridden method can be covariant added in Java 5 which facilitates to change the **return type** of the **overriding method**in the subclass. However the return type in subclass method must be a sub-type of super class method return type**.**

### Can method be overloaded by changing the return type?

-No, you should atleast change the types of arguments or order of arguments.

### Can overridden method be overloaded?

Yes. Overridden method can be overloaded.

### Can we override static method in Java?

No, you cannot override a static method because it’s not bounded to an object. Static methods belong to a class and resolved at compile time using the type of reference variable. But, you can declare the same static method in a subclass, that will result in method hiding.

### Can we override a static method as instance method?

No, instance method cannot override the static method.

### Can we override a private method in Java?

No. Since the private method is only accessible and visible inside the class only, They cannot be overridden in subclasses. You can override them inside the inner class as they are accessible there.

### How to prevent method overriding?

The **final** keyword is used to prevent method overriding. We can make method final and its can’t be overridden by child class.

### How to stop class inheritance?

We can make class final to stop inheritance means no subclass can be created for final class.

### Can we overload final method?

Yes. We can overload final method but can’t override them.

### What is default constructor?

Constructor without any arguments is called default constructor. Compiler will add default constructor if there is no constructor inside a class.

### What is Constructor overloading?

Within a class we can have default and parameterized constructors. If you don’t define any constructor, compiler automatically create a default constructor for that class. But while you define parameterized constructor compiler will not add default constructor. So you should add default constructor also while defining parameterized constructors.

### Why java 8 introduces default method in an interface?

It was added primarily to let you add methods to existing interfaces that are already in use without breaking everyone’s code, but also to share method implementations across classes implementing the same interface.  
We can create default method using default keyword. [**Read more**](http://www.startwithjava.com/java-8-interface-default-and-static-methods/).

### What are the sequence of execution of instance block,static block and constructor?

First static block get executed while loading class. Each instance block get placed at the top of constructor method and constructor method get executed.

|  |  |
| --- | --- |
| 01  02  03  04  05  06  07  08  09  10  11  12  13  14  15  16  17  18  19  20  21  22  23 | package com.startwithjava;  class X{  static{  System.out.println("X static block");  }  {  System.out.println("Instance block 1");  }  public X(int x){    System.out.println("X constructor()");  }    {   System.out.println("Instance block 2");  }    }    public class ConstructorOverloading {  public static void main(String[] args) {   X x= new X(10);   }  } |

Output

|  |  |
| --- | --- |
| 1  2  3  4 | X static block  Instance block 1  Instance block 2  X constructor() |

### What is Functional Interface?

Functional Interface is added in Java 8 to support lambda expression. Functional interface can have only one abstract method. Java 8 provide **@FunctionalInterface** annotation for which give compile error while you define more than one abstract method in functional interface. [**Read More**](http://www.startwithjava.com/functional-interfaces-and-lambda-expressions/)

### What is Output of following?

|  |  |
| --- | --- |
| 01  02  03  04  05  06  07  08  09  10  11  12  13  14  15  16  17  18  19  20  21 | package com.startwithjava;  class A{  public int x=10;  public void m1(){  System.out.println("A m1()");  }  }  class B extends A{  public int x=20;  public void m1(){  System.out.println("B m1()");  }  }  public class OverloadingExample {    public static void main(String[] args) {  A b = new B();  System.out.println(b.x);  b.m1();  }  } |

#### Output

|  |  |
| --- | --- |
| 1  2 | 10  B m1() |

### What will be output of following?

|  |  |
| --- | --- |
| 01  02  03  04  05  06  07  08  09  10  11  12  13  14  15  16  17  18  19  20  21  22 | package com.startwithjava;  class A{  public int x=10;  public void m1(){  System.out.println("A m1()");  }  }  class B extends A{  public int x=20;  public void m1(String name){  System.out.println("Name="+name);  }  }  public class OverloadingExample {    public static void main(String[] args) {  A b = new B();  System.out.println(b.x);  b.m1("Abc");  }    } |

#### Output

|  |  |
| --- | --- |
| 1  2 | As "b" is reference of class A that's why method m1(String) is not visible.  As parent class could not access the child class methods. |

### What will be output of following?

|  |  |
| --- | --- |
| 01  02  03  04  05  06  07  08  09  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37 | package com.startwithjava;    class X{  static{  System.out.println("X static block");  }  {  System.out.println("Instance block 1");  }  public X(){  System.out.println("X constructor()");  }  {  System.out.println("Instance block 2");  }  }  class Y extends X{  static{  System.out.println("Y static block");  }  public Y(){  System.out.println("Y constructor()");  }  {  System.out.println("Instance block 3");  }  {  System.out.println("Instance block 4");  }  }  public class ConstructorOverloading {    public static void main(String[] args) {  X y= new Y();  }    } |

1. First static block get executed for both parent and sub class. Then instance blocks are placed under constructor method. So parent class constructor invoked first and then sub class constructor invoked.

#### Output

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8 | X static block  Y static block  Instance block 1  Instance block 2  X constructor()  Instance block 3  Instance block 4  Y constructor() |

### What will be output for the following?

|  |  |
| --- | --- |
| 01  02  03  04  05  06  07  08  09  10  11  12  13  14  15  16  17 | package com.startwithjava;    public class MethodOverloading {  public static void m1(int x,double y){  System.out.println("int,int");  }  public static void m1(double x,double y){  System.out.println("double,double");  }  public static void m1(double x,int y){  System.out.println("double,int");  }  public static void main(String[] args) {  m1(10,10);  }    } |

#### Output

1. The compiler will complain for “The method m1(int, double) is ambiguous for the type MethodOverloading”. As we see there are two ambiguous methods **m1(int,double)** and **m1(double,double)**.

### What are output of following?

|  |  |
| --- | --- |
| 01  02  03  04  05  06  07  08  09  10  11  12  13  14  15  16  17 | package com.startwithjava;    public class MethodOverloading {  public static void m1(String s){  System.out.println("string");  }  public static void m1(StringBuffer sb){  System.out.println("stringbuffer");  }  public static void m1(Object object){  System.out.println("object");  }  public static void main(String[] args) {  m1(null);  }    } |

1. **Output**
2. Compile time error for ambiguous methods, **m1(String)** and **m1(StringBuffer)**. Where there is overloaded methods with child and parent as arguments then in this case compiler choose the most specific method(child class argument).
3. As here are three methods **m1(Object)**,**m1(String)** and **m1(StringBuffer)**, compiler first looks for for most specified method but there are two sub classes **String** and **StringBuffer** for Object class. No inheritance there between String and StringBuffer and null can hold by any reference type that why there is ambiguity issue.

### What is output for following?

|  |  |
| --- | --- |
| 01  02  03  04  05  06  07  08  09  10  11  12  13  14  15  16  17  18  19  20 | interface A1{       default void test(){          System.out.println("A test()");        }  }  interface B1 {     default void test(){         System.out.println("B test()");      }     default void test(String msg){       System.out.println("B test(msg)");      }  }  class C1 implements A1,B1{  public static void main(String args[]) {   C1 c1 = new C1();   c1.test();   c1.test("Testing");   }  } |

1. **Output:**
2. As we know that Java 8 introduces default methods in interface. As there are two default methods in interface A1 & B1 to C1.  
   So in this case compiler will raise an error **“Duplicate default methods named test with the parameters () and () are inherited from the types B1 and A1”.**
3. To resolve this either we need to define test() method in C1 class or need to remove that method from anyone interface A1 or B1.

### What will be output for following?

|  |  |
| --- | --- |
| 01  02  03  04  05  06  07  08  09  10  11  12  13  14  15  16  17  18  19  20  21 | interface A1{    default void test(){      System.out.println("A test()");    }    }  interface B1 extends A1{      default void test(){          System.out.println("B test()");       }       default void test(String msg){          System.out.println("B test(msg)");       }  }  class C1 implements B1{  public static void main(String args[]) {   C1 c1 = new C1();   c1.test();   c1.test("Testing");   }  } |

1. **Output:**The program will execute successfully and prints below messages.

|  |  |
| --- | --- |
| 1  2 | B test()  B test(msg) |

### Why java 8 introduces static method in an interface?

Static method in interface cannot be implemented in concrete class rather it can be used with interface itself.  
If you override them in concrete class then it hides that static method.

Note:- If you annotate overriden method in concrete class using **@Override** then compiler time error for no method is in SuperType. As static method from interface is not visible for their implementation class.

|  |  |
| --- | --- |
| 1  2  3  45 | interface A{   static void test(){    System.out.println("hello");   }  } |

**1. What is OOPS?**

OOPS is abbreviated as Object Oriented Programming system in which programs are considered as a collection of objects. Each object is nothing but an instance of a class.

**2. Write basic concepts of OOPS?**

Following are the concepts of OOPS and are as follows:

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

**3. What is a class?**

A class is simply a representation of a type of object. It is the blueprint/ plan/ template that describes the details of an object.

**4. What is an object?**

An object is an instance of a class. It has its own state, behavior, and identity.

**5. What is Encapsulation?**

Encapsulation is an attribute of an object, and it contains all data which is hidden. That hidden data can be restricted to the members of that class.

Levels are Public, Protected, Private, Internal and Protected Internal.

**6. What is Polymorphism?**

Polymorphism is nothing but assigning behavior or value in a subclass to something that was already declared in the main class. Simply, polymorphism takes more than one form.

**7. What is Inheritance?**

Inheritance is a concept where one class shares the structure and behavior defined in another class. If inheritance applied on one class is called Single Inheritance, and if it depends on multiple classes, then it is called multiple Inheritance.

**8. What are manipulators?**

Manipulators are the functions which can be used in conjunction with the insertion (<<) and extraction (>>) operators on an object. Examples are endl and setw.

**9. Define a constructor?**

A constructor is a method used to initialize the state of an object, and it gets invoked at the time of object creation. Rules forconstructor are:

* Constructor Name should be same as class name.
* A constructor must have no return type.

**10. Define Destructor?**

A destructor is a method which is automatically called when the object is made of scope or destroyed. Destructor name is also same as class name but with the tilde symbol before the name.

**11. What is an Inline function?**

An inline function is a technique used by the compilers and instructs to insert complete body of the function wherever that function is used in the program source code.

**12. What is a virtual function?**

A virtual function is a member function of a class, and its functionality can be overridden in its derived class. This function can be implemented by using a keyword called virtual, and it can be given during function declaration.

A virtual function can A token in C++, and it can be achieved in C Language by using function pointers or pointers to function.

**13. What is a friend function?**

A friend function is a friend of a class that is allowed to access to Public, private or protected data in that same class. If the function is defined outside the class cannot access such information.

Friend can be declared anywhere in the class declaration, and it cannot be affected by access control keywords like private, public or protected.

**14. What is function overloading?**

Function overloading an as a normal function, but it can perform different tasks. It allows the creation of several methods with the same name which differ from each other by the type of input and output of the function.

Example

void add(int& a, int& b);

void add(double& a, double& b);

void add(struct bob& a, struct bob& b);

**15. What is operator overloading?**

Operator overloading is a function where different operators are applied and depends on the arguments. Operator,-,\* can be used to pass through the function, and it has their own precedence to execute

**16. What is an abstract class?**

An abstract class is a class which cannot be instantiated. Creation of an object is not possible with an abstract class, but it can be inherited. An abstract class can contain only Abstract method. Java allows only abstract method in abstract class while for other languages allow non-abstract method as well.

**17. What is a ternary operator?**

The ternary operator is said to be an operator which takes three arguments. Arguments and results are of different data types, and it depends on the function. The ternary operator is also called a conditional operator.

**18. What is the use of finalize method?**

Finalize method helps to perform cleanup operations on the resources which are not currently used. Finalize method is protected, and it is accessible only through this class or by a derived class.

**19. What are different types of arguments?**

A parameter is a variable used during the declaration of the function or subroutine and arguments are passed to the an, and it should match with the parameter defined. There are two types of Arguments.

* Call by Value – Value passed will get modified only inside the function, and it returns the same value whatever it is passed it into the function.
* Call by Reference – Value passed will get modified in both inside and outside the functions and it returns the same or different value.

**20. What is the super keyword?**

Super keyword is used to invoke the overridden method which overrides one of its superclass methods. This keyword allows to access overridden methods and also to access hidden members of the superclass.

It also forwards a call from a constructor to a constructor in the superclass.

**21. What is method overriding?**

Method overriding is a feature that allows a subclass to provide the implementation of a method that overrides in the main class. This will overrides the implementation in the superclass by providing the same method name, same parameter and same return type.

**22. What is an interface?**

An interface is a collection of an abstract method. If the class implements an inheritance, and then thereby inherits all the abstract methods of an interface.

**23.   What is exception handling?**

An exception is an event that occurs during the execution of a program. Exceptions can be of any type – Runtime exception, Error exceptions. Those exceptions are adequately handled through exception handling mechanism like try, catch and throw keywords.

**24. What are tokens?**

The token is recognized by a compiler, and it cannot be broken down into component elements. Keywords, identifiers, constants, string literals and operators are examples of tokens.

Even punctuation characters are also considered as tokens – Brackets, Commas, Braces and Parentheses.

**25. Difference between overloading and overriding?**

Overloading is static binding whereas Overriding is dynamic binding. Overloading is nothing but the same method with different arguments, and it may or may not return the same value in the same class itself.

Overriding is the same method names with same arguments and return types associated with the class and its child class.

**26. Difference between class and an object?**

An object is an instance of a class. Objects hold multiple information, but classes don’t have any information. Definition of properties and functions can be done in class and can be used by the object.

A class can have sub-classes, and an object doesn’t have sub-objects.

**27. What is an abstraction?**

Abstraction is a good feature of OOPS, and it shows only the necessary details to the client of an object. Means, it shows only required details for an object, not the inner constructors, of an object. Example – When you want to switch On television, it not necessary to show all the functions of TV. Whatever is required to switch on TV will be showed by using abstract class.

**28. What are access modifiers?**

Access modifiers determine the scope of the method or variables that can be accessed from other various objects or classes. There are 5 types of access modifiers, and they are as follows:

* Private.
* Protected.
* Public.
* Friend.
* Protected Friend.

**29. What are sealed modifiers?**

Sealed modifiers are the access modifiers where it cannot be inherited by the methods. Sealed modifiers can also be applied to properties, events, and methods. This modifier cannot be applied to static members.

**30. How can we call the base method without creating an instance?**

Yes, it is possible to call the base method without creating an instance. And that method should be “Static method”.

Doing inheritance from that class.-Use Base Keyword from a derived class.

**31. What is the difference between new and override?**

The new modifier instructs the compiler to use the new implementation instead of the base class function. Whereas, Override modifier helps to override the base class function.

**32. What are the various types of constructors?**

There are three various types of constructors, and they are as follows:

–  Default Constructor – With no parameters.

–  Parametric Constructor – With Parameters. Create a new instance of a class and also passing arguments simultaneously.

–  Copy Constructor – Which creates a new object as a copy of an existing object.

**33. What is early and late binding?**

Early binding refers to the assignment of values to variables during design time whereas late binding refers to the assignment of values to variables during run time.

**34. What is ‘this’ pointer?**

THIS pointer refers to the current object of a class. THIS keyword is used as a pointer which differentiates between the current object with the global object. Basically, it refers to the current object.

**35. What is the difference between structure and a class?**

Structure default access type is public , but class access type is private. A structure is used for grouping data whereas class can be used for grouping data and methods. Structures are exclusively used for data, and it doesn’t require strict validation , but classes are used to encapsulates and inherit data which requires strict validation.

**36. What is the default access modifier in a class?**

The default access modifier of a class is Private by default.

**37. What is a pure virtual function?**

A pure virtual function is a function which can be overridden in the derived class but cannot be defined. A virtual function can be declared as Pure by using the operator =0.

Example -.

|  |  |
| --- | --- |
| 1  2  3 | Virtual void function1() // Virtual, Not pure    Virtual void function2() = 0 //Pure virtual |

**38. What are all the operators that cannot be overloaded?**

Following are the operators that cannot be overloaded -.

1. Scope Resolution (:: )
2. Member Selection (.)
3. Member selection through a pointer to function (.\*)

**39. What is dynamic or run time polymorphism?**

Dynamic or Run time polymorphism is also known as method overriding in which call to an overridden function is resolved during run time, not at the compile time. It means having two or more methods with the same name, same signature but with different implementation.

**40. Do we require a parameter for constructors?**

No, we do not require a parameter for constructors.

**41. What is a copy constructor?**

This is a special constructor for creating a new object as a copy of an existing object. There will always be only one copy constructor that can be either defined by the user or the system.

**42. What does the keyword virtual represented in the method definition?**

It means, we can override the method.

**43. Whether static method can use nonstatic members?**

False.

**44. What is a base class, sub class, and super class?**

The base class is the most generalized class, and it is said to be a root class.

A Sub class is a class that inherits from one or more base classes.

The superclass is the parent class from which another class inherits.

**45. What is static and dynamic binding?**

Binding is nothing but the association of a name with the class. Static binding is a binding in which name can be associated with the class during compilation time, and it is also called as early Binding.

Dynamic binding is a binding in which name can be associated with the class during execution time, and it is also called as Late Binding.

**46. How many instances can be created for an abstract class?**

Zero instances will be created for an abstract class.

**47. Which keyword can be used for overloading?**

Operator keyword is used for overloading.

**48. What is the default access specifier in a class definition?**

Private access specifier is used in a class definition.

**49. Which OOPS concept is used as reuse mechanism?**

Inheritance is the OOPS concept that can be used as reuse mechanism.

**50. Which OOPS concept exposes only necessary information to the calling functions?**

Encapsulation