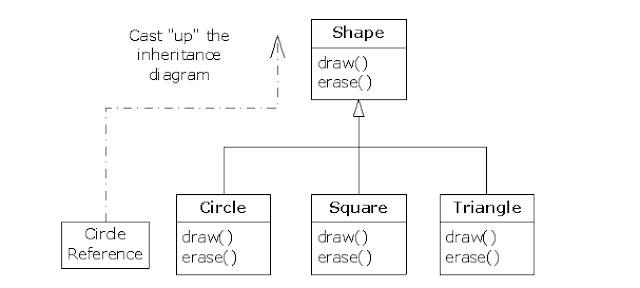
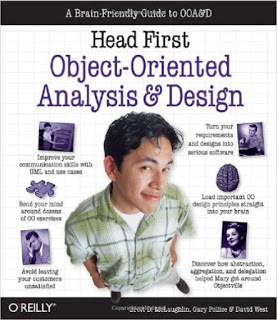
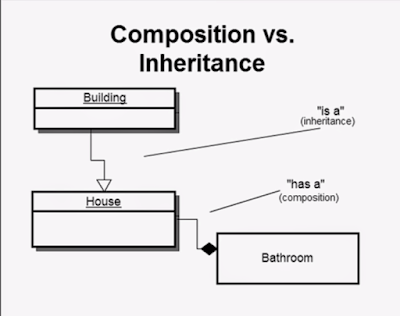
**Question 1: What is Inheritance in Java?**([detailed answer](http://java67.blogspot.com/2012/08/what-is-inheritance-in-java-oops-programming-example.html))  
Answer: Inheritance is an Object oriented feature which allows a class to inherit behavior and data from other class. For example, a class Car can extend basic feature of Vehicle class by using Inheritance. One of the most intuitive examples of Inheritance in the real world is Father-Son relationship, where Son inherit Father's property. If you don't know, Inheritance is the quick way to become rich :)  
  
  
**Question 2: What are different types of Inheritance supported by Java?**(detailed answer)  
Answer: Java supports single Inheritance, multi-level inheritance and at some extent multiple inheritances because Java allows a class to only extend another class, but an interface in Java can extend multiple inheritances.

[](https://pluralsight.pxf.io/c/1193463/424552/7490?u=https://www.pluralsight.com/courses/principles-oo-design)

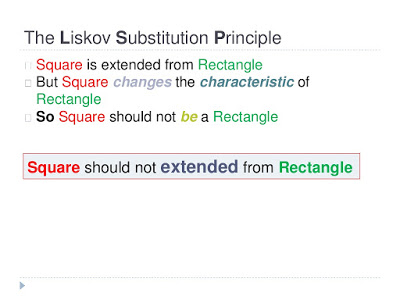
**Question 3: Why multiple Inheritance is not supported by Java?**([detailed answer](http://javarevisited.blogspot.com/2011/07/why-multiple-inheritances-are-not.html))  
Answer: Java is introduced after C++ and Java designer didn't want to take some C++ feature which is confusing and not essential. They think multiple inheritances is one of them which doesn't justify complexity and confusion it introduces. You can also check why multiple inheritances are not supported in Java for more reasons and discussion around this.  
  
  
**Question 4: Why Inheritance is used by Java Programmers?**([detailed answer](http://java67.blogspot.com/2015/12/top-30-oops-concept-interview-questions-answers-java.html))  
Answer: Inheritance is used for code reuse and leveraging Polymorphism by creating a type hierarchy. It's better to use Inheritance for type declaration but for code reuse composition is a better option because it's more flexible. See this article for learning more about why Composition is better than Inheritance.  
  
  
**Question 5: How to use Inheritance in Java?**([detailed answer](http://javarevisited.blogspot.com/2012/10/what-is-inheritance-in-java-and-oops-programming.html))  
Answer: You can use Inheritance in Java by extending classes and implementing interfaces. Java provides two keywords extends and implements to achieve inheritance.  A class which is derived from another class is known as a subclass and an interface which is derived from another interface is called subinterface. A class which implements an interface is known as implementation.  
  
  
 **Question 6: What is the syntax of Inheritance?**(detailed answer)  
Answer: You can use either extends of implements keyword to implement Inheritance in Java.  A class extends another class using extends keyword, an interface can extend another interface using extend keyword, and a class can implement an interface using implements keyword in Java.  
  
  
**Question 7: What is the difference between Inheritance and Encapsulation?**(detailed answer)  
Answer: Inheritance is an object oriented concept which creates a parent-child relationship. It is one of the ways to reuse the code written for parent class but it also forms the basis of Polymorphism. On the other hand, Encapsulation is an object oriented concept which is used to hide the internal details of a class e.g. HashMap encapsulate how to store elements and how to calculate hash values.  
  
  
**Question 8: What is the difference between Inheritance and Abstraction?**(detailed answer)  
Answer: Abstraction is an object oriented concept which is used to simply things by abstracting details. It helps in the designing system. On the other hand, Inheritance allows code reuse. You can reuse the functionality you have already coded by using Inheritance. See [Head First Object Oriented Analysis and Design](http://www.amazon.com/dp/0596008678/?tag=javamysqlanta-20) for more details.

[](http://www.amazon.com/dp/0596008678/?tag=javamysqlanta-20)

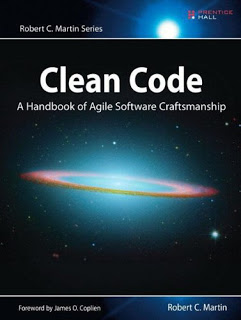
**Question 9: What is the difference between Polymorphism and Inheritance?** ([detailed answer](http://java67.blogspot.com/2014/04/difference-between-polymorphism-and-Inheritance-java-oops.html))  
Answer: Both Polymorphism and Inheritance goes hand on hand, they help each other to achieve their goal. Polymorphism allows flexibility, you can choose which code to run at runtime by overriding.  See the detailed answer for more details.  
  
 **Question 10: What is the difference between Composition and Inheritance in OOP?**([detailed answer](http://javarevisited.blogspot.com/2015/06/difference-between-inheritance-and-Composition-in-Java-OOP.html))  
Answer: One of the good question to check the candidate's object-oriented programming skills. There are several differences between Composition and Inheritance in Java, some of them are following:  
  
1. The Composition is more flexible because you can change the implementation at runtime by calling setXXX() method, but Inheritance cannot be changed i.e. you cannot ask a class to implement another class at runtime.  
  
2. Composition builds HAS-A relationship while Inheritance builds IS-A relationship e.g. A Room HAS A Fan, but Mango IS-A Fruit.  
  
3. The parent-child relationship is best represented using Inheritance but If you just want to use the services of another class use Composition. For more differences see [5 reasons to favor composition over Inheritance](http://javarevisited.blogspot.com/2013/06/why-favor-composition-over-inheritance-java-oops-design.html).

[](https://click.linksynergy.com/fs-bin/click?id=JVFxdTr9V80&subid=0&offerid=323058.1&type=10&tmpid=14538&RD_PARM1=https://www.udemy.com/java-object-oriented-programming-for-absolute-beginners/)

**11. Can we override static method in Java?**([detailed answer](http://java67.blogspot.com/2012/08/can-we-override-static-method-in-java.html))  
No, you cannot override a static method in Java because it's resolved at compile time. In order for overriding to work, a method should be virtual and resolved at runtime because objects are only available at runtime. This is one of the tricky Java questions, where interviewer tries to confuse you. A programmer is never sure about whether they can override or overload a static method in Java.  
  
  
**12. Can we overload a static method in Java?**([detailed answer](http://java67.blogspot.com/2012/08/can-we-overload-static-method-in-java.html))  
Yes, you can overload a static method in Java. Overloading has nothing to do with runtime but the signature of each method must be different. In Java, to change the method signature, you must change either number of arguments, type of arguments or order of arguments.  
  
 **13. Can we override a private method in Java?**([detailed answer](http://java67.blogspot.com/2012/08/can-we-override-private-method-in-java.html))  
No,  you cannot override a private method in Java because the private method is not inherited by the subclass in Java, which is essential for overriding. In fact, a private method is not visible to anyone outside the class and, more importantly, a call to the private method is resolved at compile time by using Type information as opposed to runtime by using the actual object.  
  
 **Question 14: What is method hiding in Java?**([detailed answer](http://java67.blogspot.com/2015/08/top-10-method-overloading-overriding-interview-questions-answers-java.html))  
Answer: Since the static method cannot be overridden in Java, but if you declare the same static method in subclass then that would hide the method from the superclass. It means, if you call that method from subclass then the one in the subclass will be invoked but if you call the same method from superclass then the one in superclass will be invoked. This is known as method hiding in Java.  
  
  
**Question 15: Can a class implement more than one interface in Java?**([detailed answer](http://java67.blogspot.com/2014/02/what-is-actual-use-of-interface-in-java.html))  
Yes, A class can implement more than one interface in Java e.g. A class can be both Comparable and Serializable at the same time. This is why the interface should be the best use for defining Type as described in Effective Java. This feature allows one class to play a polymorphic role in the program.  
 **Question 16: Can a class extends more than one class in Java?**(detailed answer)  
Answer: No, a class can only extend just one more class in Java.  Though Every class also, by default extend the java.lang.Object class in Java.  
  
  
**Question 17: Can an interface extends more than one interface in Java?**([answer](http://java67.blogspot.com/2013/07/15-advanced-core-java-interview-questions-answers-senior-experienced-5-6-years-programmers-developers.html))  
Answer: Yes, unlike classes, an interface can extend more than one interface in Java. There are several example of this behavior in JDK itself e.g. java.util.List interface extends both Collection and Iterable interface to tell that it is a Collection as well as it allows iteration via Iterator.  
  
  
**18: What will happen if a class extends two interfaces and they both have a method with same name and signature?**(detailed answer)  
In this case, a conflict will arise because the compiler will not able to link a method call due to ambiguity. You will get a compile time error in Java.  
  
 **Question 19: Can we pass an object of a subclass to a method expecting an object of the super class?**([answer](http://java67.blogspot.com/2013/07/15-advanced-core-java-interview-questions-answers-senior-experienced-5-6-years-programmers-developers.html))  
Answer: Yes, you can pass that because subclass and superclass are related to each other by Inheritance which provides IS-A property.  I mean Banana is a Fruit so you can pass banana if somebody expect fruit. Now there are scenario, where you can't do e.g. when subclass violates the Liskov Substitution principle i.e. you cannot pass a plastic banana to someone expecting fruit :-), The eat() function will throw exception.

[](https://click.linksynergy.com/fs-bin/click?id=JVFxdTr9V80&subid=0&offerid=323058.1&type=10&tmpid=14538&RD_PARM1=https://www.udemy.com/java-object-oriented-programming-for-absolute-beginners/)

**Question 20: What is the Liskov substitution principle?**(detailed answer)  
Answer: The Liskov substitution principle is one of the five object-oriented design principles, collectively know as [SOLID principles](http://javarevisited.blogspot.com/2012/03/10-object-oriented-design-principles.html). This design principle is L of SOLID acronym. The Liskov substitution principle states that in an object oriented program if a function or method is expecting an object of base class then  it should work fine with a derived class object as well. If it cannot function properly with derived class object then the derived class is violating the Liskov Substitution principle.  
  
For example, if a method is expecting a List you can also pass [ArrayList or LinkedList](http://java67.blogspot.com/2012/12/difference-between-arraylist-vs-LinkedList-java.html) and it should work just fine because ArrayList and LinkedList both follow Liskov Substitution Principle, but the [java.sql.Date](http://java67.blogspot.com/2014/02/how-to-convert-javautildate-to-javasqldate-example.html) which is a subclass of java.util.Date in Java violates Liskov Substitution Principle because you cannot pass an object of java.sql.Date class to a method which is expecting an object of java.util.Date, Why? because all time-related method will throw java.lang.UnsupportedOperationException.  
  
Here is another example of violating The Liskov Substitution Principle, Square is a special type of Rectangle whose adjacent sides are equal but making Square extending Rectangle violates LSP principle. For more details on SOLID design principles, read [Clean Code](http://www.amazon.com/Clean-Code-Handbook-Software-Craftsmanship/dp/0132350882?tag=javamysqlanta-20) by Rober C. Martin, the inventor of SOLID principles.

[](http://www.amazon.com/Clean-Code-Handbook-Software-Craftsmanship/dp/0132350882?tag=javamysqlanta-20)

**Question 21: How to call a method of a subclass, if you are holding an object of the subclass in a reference variable of type superclass?**([answer](http://javarevisited.blogspot.com/2012/12/what-is-type-casting-in-java-class-interface-example.html))  
Answer: You can call a method of the subclass by first casting the object hold by reference variable of  superclass into the subclass. Once you hold the object in subclass reference type, you can call methods from the subclass. See how type casting works in Java for more details.

**What is inheritance in Java?**  
**Ans:** Acquiring the features from existing entities is known as “inheritance”. The existing entity is called a super or parent class and the new entity that is acquiring the features of an existing entity is called sub-class or child class. The process of inheriting the properties and behaviors from one object to another object is known as java. Creating a new class from an existing class using an IS-A relationship is known as “inheritance”.

**How do you implement inheritance practically in Java?**  
**Ans:** There are two keywords in java to implement is-a relationship

* Extends
* Implements.

**What is the purpose of inheritance?**  
**Ans:** Inheritance offers code reusability and thereby Rapid Application Development (RAD) is possible.

**What are the generalized and specialized classes in Java?**  
**Ans:** The top-level or superclasses are known as “generalized class”. It contains common data and common behavior.  
The low-level or sub-level classes are known as “specialized classes”. It contains more specific data  
Ex:

* **class** Person
* **{**
* int name;
* int age;
* **}**
* Class Student **extends** Person
* **{**
* int rollNo;
* int marks;
* **}**
* **class** Employee **extends** Person
* **{**
* int empId;
* float empSal;
* **}**

In this class hierarchy Person is generalized class and Student and Employee are the specialized classes.

**What are the types of inheritance?**  
**Ans:** The following are the types of inheritances:

* Single inheritance: If a class is having only one parent class that is known as “single inheritance”.
* Multiple inheritances

The following are two special cases of single inheritance supported by Java: Hierarchical Inheritance and Multi-level inheritance.

* **Hierarchical inheritance:** If a class has more than one sub-class that form of single inheritance especially known as “hierarchical inheritance”.
* **Multi-level inheritance:**
* Multiple Inheritance: The mechanism of inheriting the features of more than one base class into a single class is known as multiple inheritances. Java does not support multiple inheritances but the multiple inheritances can be achieved by using the interfaces by implementing more than one interface in a class.

**Inheritance Interview Questions For Experienced**

**What is the purpose of the ‘super’ keyword in Java?**  
**Ans:** The following are the major uses of super keyword they are:

* If your method overrides one of its super class’s methods, you can invoke the overridden method through the use of the keyword super.
* Using a super keyword we can explicitly call the immediate superclass constructor from the subclass constructor.
* From sub-class, to access instance variables of superclass where a subclass is also having a variable with the same name.

**What are the differences between ‘this’ and ‘super’ keyword?**  
**Ans:** Using ‘this’ from within one class constructor, a call can be made to another constructor of the same class. Whereas, using ‘super’ keyword superclass constructor is called from sub-class constructor.

**What are the rules to be followed while overriding a method?**

**Ans:** The following are the rules to be followed while overriding a method:

* The overriding method has the same name, number, and type of parameters, and return type as the method it overrides.
* For the child method accessibility should not be reduced than that of the parent class method, equal is ok or higher is ok.
* In the child method exception specification, extra checked exception classes should not be listed then that of parent class method, equal is ok or less is ok.

**What is the base class of all classes?**  
**Ans:** The java.lang.An object is the base class of all classes.

**Does Java support multiple inheritances?**  
**Ans:** Java doesn’t support multiple inheritances.

**How to define a constant variable in Java?**  
**Ans:** The variable should be declared as static and final. So only one copy of the variable exists for all instances of the class and the value can’t be changed also. static final int PI = 2.14; is an example for constant.

**What is the purpose of declaring a variable as ‘final’?**  
**Ans:** A final variable’s value can’t be changed. The ‘final’ variables should be initialized before using them.

**What is the impact of declaring a method as final?**  
**Ans:** A method declared as final can’t be overridden. A sub-class can’t have the same method signature with a different implementation.

**I don’t want my class to be inherited by any other class. What should I do?**  
**Ans:** You should declare your class as final. But you can’t define your class as final if it is an abstract class. A class declared as final can’t be extended by any other class.

**Can you give a few examples of final classes defined in Java API?**  
**Ans:** java.lang.String, java.lang.Math is the ‘final’ classes.

**How is the final different from finally and finalize()?**  
**Ans:** The ‘final’ is a modifier that can be applied to a class or a method or a variable. ‘final’ class can’t be inherited, the final method can’t be overridden and final variable can’t be changed. The ‘finally’ is an exception handling code section which gets executed whether an exception is raised or not by the try block code segment. The ‘finalize()’ is a method of the Object class that will be executed by the JVM just before garbage collecting objects to give a final chance for resource releasing activity.

**Does a class inherit the constructors of its superclass?**  
**Ans:** A class does not inherit constructors from any of its superclasses.

**What is Overriding?**  
**Ans:** When a class defines a method using the same name, return type, and arguments as a method in its superclass, the method in the class overrides the method in the superclass. When the method is invoked for an object of the class, it is the new definition of the method that is called and not the method definition from the superclass. Methods may be overridden to be more public, not more private.

**How is this() and super() used with constructors?**  
**Ans:** The ‘this()’ method is used to invoke a constructor of the same class. Whereas the ’super()’ is used to invoke a superclass constructor.

**What modifiers are allowed for methods in an Interface?**  
**Ans:** Only public and abstract modifiers are allowed for methods in interfaces.

**Latest Inheritance Java Interview Questions**

**What are some alternatives to inheritance?**  
**Ans:** Delegation is an alternative to inheritance.  Delegation means that you include an instance of another class as an instance variable and forward messages to the instance. It is often safer than inheritance because it forces you to think about each message you forward, because the instance is of a known class, rather than a new class, and because it doesn’t force you to accept all the methods of the superclass: you can provide only the methods that really make sense. On the other hand, it makes you write more code, and it is harder to re-use (because it is not a subclass).

**Does a class inherit the constructors of its superclass?**  
**Ans:** A class does not inherit constructors from any of its superclasses.

**What restrictions are placed on method overloading?**  
**Ans:** Two methods may not have the same name and argument list but different return types.

**What is method overloading & method overriding?**  
**Ans:** Method overloading: When a method in a class having the same method name with different arguments is said to be method overloading. Method overriding: When a method in a class having the same method name with the same arguments is said to be method overriding.

**What is the difference between overloading & overriding?  
Ans:**

* In overloading, there is a relationship between methods available in the same class whereas in overriding, there is a relationship between a superclass method and a subclass method.
* Overloading does not block inheritance from the superclass whereas overriding blocks inheritance from the superclass.
* In overloading, separate methods share the same name whereas in overriding, the subclass method replaces the superclass.
* Overloading must have different method signatures whereas overriding must have the same signature.

**What is meant by Inheritance and what are its advantages?**  
**Ans:** Inheritance is the process of inheriting all the features from a class. The advantages of inheritance are the reusability of code and accessibility of variables and methods of the superclass by subclasses.

**What is the difference between the superclass and the subclass?**  
**Ans:** A superclass is a class that is inherited whereas subclass is a class that does the inheriting.

**What modifiers may be used with top-level class?**  
**Ans:** The public, abstract and final can be used for top-level class.

If you found any mistake in the above interview questions or you want to share the missed interview questions then you can comment on the comment section.

1. **Question 1. What Is Inheritance In Java?**

**Answer :**

Inheritance is an Object oriented feature which allows a class to inherit behavior and data from other class. For example, a class Car can extend basic feature of Vehicle class by using Inheritance. One of the most intuitive examples of Inheritance in the real world is Father-Son relationship, where Son inherit Father's property. If you don't know, Inheritance is the quick way to become rich.

1. **Question 2. Why Multiple Inheritance Is Not Supported By Java?**

**Answer :**

Java is introduced after C++ and Java designer didn't want to take some C++ feature which is confusing and not essential. They think multiple inheritances is one of them which doesn't justify complexity and confusion it introduces. You can also check why multiple inheritances are not supported in Java for more reasons and discussion around this.

[Adv Java Interview Questions](https://www.wisdomjobs.com/e-university/adv-java-interview-questions.html)

1. **Question 3. Why Inheritance Is Used By Java Programmers?**

**Answer :**

Inheritance is used for code reuse and leveraging Polymorphism by creating a type hierarchy. It's better to use Inheritance for type declaration but for code reuse composition is a better option because it's more flexible. See this article for learning more about why Composition is better than Inheritance.

1. **Question 4. How To Use Inheritance In Java?**

**Answer :**

You can use Inheritance in Java by extending classes and implementing interfaces. Java provides two keywords extends and implements to achieve inheritance.  A class which is derived from another class is known as a subclass and an interface which is derived from another interface is called subinterface. A class which implements an interface is known as implementation.

[Adv Java Tutorial](https://www.wisdomjobs.com/e-university/adv-java-tutorial-227.html)

1. **Question 5. What Is The Syntax Of Inheritance?**

**Answer :**

You can use either extends of implements keyword to implement Inheritance in Java.  A class extends another class using extends keyword, an interface can extend another interface using extend keyword, and a class can implement an interface using implements keyword in Java.

[J2EE Interview Questions](https://www.wisdomjobs.com/e-university/j2ee-interview-questions.html)

1. **Question 6. What Is The Difference Between Inheritance And Encapsulation?**

**Answer :**

Inheritance is an object oriented concept which creates a parent-child relationship. It is one of the ways to reuse the code written for parent class but it also forms the basis of Polymorphism. On the other hand, Encapsulation is an object oriented concept which is used to hide the internal details of a class e.g. HashMap encapsulate how to store elements and how to calculate hash values.

1. **Question 7. What Is The Difference Between Polymorphism And Inheritance?**

**Answer :**

Both Polymorphism and Inheritance goes hand on hand, they help each other to achieve their goal. Polymorphism allows flexibility, you can choose which code to run at runtime by overriding.  See the detailed answer for more details.

[J2EE Tutorial](https://www.wisdomjobs.com/e-university/j2ee-tutorial-230.html) [Core Java Interview Questions](https://www.wisdomjobs.com/e-university/core-java-interview-questions.html)

1. **Question 8. Can We Override Static Method In Java?**

**Answer :**

No, you cannot override a static method in Java because it's resolved at compile time. In order for overriding to work, a method should be virtual and resolved at runtime because objects are only available at runtime. This is one of the tricky Java questions, where interviewer tries to confuse you. A programmer is never sure about whether they can override or overload a static method in Java.

1. **Question 9. Can We Overload A Static Method In Java?**

**Answer :**

Yes, you cannot overload a static method in Java. Overloading has nothing to do with runtime but the signature of each method must be different. In Java, to change the method signature, you must change either number of arguments, type of arguments or order of arguments.

[JSP Interview Questions](https://www.wisdomjobs.com/e-university/jsp-interview-questions.html)

1. **Question 10. Can We Override A Private Method In Java?**

**Answer :**

No,  you cannot override a private method in Java because the private method is not inherited by the subclass in Java, which is essential for overriding. In fact, a private method is not visible to anyone outside the class and, more importantly, a call to the private method is resolved at compile time by using Type information as opposed to runtime by using the actual object.

[Core Java Tutorial](https://www.wisdomjobs.com/e-university/core-java-tutorial-231.html)

1. **Question 11. What Is Method Hiding In Java?**

**Answer :**

Since the static method cannot be overridden in Java, but if you declare the same static method in subclass then that would hide the method from the superclass. It means, if you call that method from subclass then the one in the subclass will be invoked but if you call the same method from superclass then the one in superclass will be invoked. This is known as method hiding in Java.

[Java-Springs Interview Questions](https://www.wisdomjobs.com/e-university/java-springs-interview-questions.html)

1. **Question 12. Can A Class Implement More Than One Interface In Java?**

**Answer :**

Yes, A class can implement more than one interface in Java e.g. A class can be both Comparable and Serializable at the same time. This is why the interface should be the best use for defining Type as described in Effective Java. This feature allows one class to play a polymorphic role in the program.

[Adv Java Interview Questions](https://www.wisdomjobs.com/e-university/adv-java-practice-tests-227-327247)

1. **Question 13. Can A Class Extends More Than One Class In Java?**

**Answer :**

No, a class can only extend just one more class in Java. Through Every class also, by default extend the java.lang.Object class in Java.

[JSP Tutorial](https://www.wisdomjobs.com/e-university/jsp-tutorial-279.html)

1. **Question 14. Can An Interface Extends More Than One Interface In Java?**

**Answer :**

Yes, unlike classes, an interface can extend more than one interface in Java. There are several example of this behavior in JDK itself e.g. java.util.List interface extends both Collection and Iterable interface to tell that it is a Collection as well as it allows iteration via Iterator.

1. **Question 15. What Will Happen If A Class Extends Two Interfaces And They Both Have A Method With Same Name And Signature?**

**Answer :**

In this case, a conflict will arise because the compiler will not able to link a method call due to ambiguity. You will get a compile time error in Java.

[JMS(Java Message Service) Interview Questions](https://www.wisdomjobs.com/e-university/jms-java-message-service-interview-questions.html)

1. **Question 16. What Is The Liskov Substitution Principle?**

**Answer :**

* + The Liskov substitution principle is one of the five object-oriented design principles, collectively know as SOLID principles. This design principle is L of SOLID acronym. The Liskov substitution principle states that in an object oriented program if a function or method is expecting an object of base class then  it should work fine with a derived class object as well. If it cannot function properly with derived class object then the derived class is violating the Liskov Substitution principle.
  + For example, if a method is expecting a List you can also pass ArrayList or LinkedList and it should work just fine because ArrayList and LinkedList both follow Liskov Substitution Principle, but the java.sql.Date which is a subclass of java.util.Date in Java violates Liskov Substitution Principle because you cannot pass an object of java.sql.Date class to a method which is expecting an object of java.util.Date, Why? because all time-related method will throw java.lang.UnsupportedOperationException.
  + Here is another example of violating The Liskov Substitution Principle, Square is a special type of Rectangle whose adjacent sides are equal but making Square extending Rectangle violates LSP principle. For more details on SOLID design principles, read Clean Code by Rober C. Martin, the inventor of SOLID principles.

[Java-Springs Tutorial](https://www.wisdomjobs.com/e-university/java-springs-tutorial-287.html)

1. **Question 17. How To Call A Method Of A Subclass, If You Are Holding An Object Of The Subclass In A Reference Variable Of Type Superclass?**

**Answer :**

* + You can call a method of the subclass by first casting the object hold by reference variable of  superclass into the subclass. Once you hold the object in subclass reference type, you can call methods from the subclass. See how type casting works in Java for more details.
  + That's all about some good interview questions based OOP concept,  Inheritance. You should also know that how private and final variables affect Inheritance. How can you extend a class which is holding a private variable, and probably the difference between private and protected modifier in Java? They are really important to understand and use Inheritance in Java.
  + Also, if you think we have missed any important question related to Inheritance o If you come across any good questions based upon Inheritance or other object oriented concept then please share with us.

Question: Types of Inheritance in Java?

There are 5 types of Inheritance.

* **Single Inheritance**: One class is extended by only one class.
* **Multilevel Inheritance:** One class is extended by a class and that class in turn is extended by another class thus forming a chain of inheritance.
* **Hierarchical Inheritance**: One class is extended by many classes.
* **Hybrid Inheritance**: It is a combination of above types of inheritance.
* **Multiple Inheritance**: One class extends more than one classes. (Java does not support multiple inheritance.)

Question: Can a class extend more than one classes or does java support multiple inheritance? If not, why?

Java does not support multiple inheritance. This feature is avoided intentionally to avoid ambiguity, complexity and confusion.

For example, If Class C extends Class A and Class B which have a method with same name, then Class C will have two methods with same name.

This causes ambiguity and confusion for which method to use. To avoid this, java does not supports multiple inheritance.

class A

{

void methodOne()

{

System.out.println("From methodOfClassA");

}

}

class B

{

void methodOne()

{

System.out.println("From methodOfClassB");

}

}

class C extends A,B (If it is supported)

{

//two same methods will be inherited to Class C.

//This causes ambiguity and confusion.

}

Question: Can a Class Extend Itself in Java?

Answer: A class can not extend itself.

Question: What is Method Overriding And Method Hiding in Java?

**Method Overriding:**

An instance method in a subclass with the same signature (name, number and the type of its parameters) and return type as an instance method in the superclass *overrides* the superclass's method.

The overriding method has the same name, number and type of parameters, and return type as the method that it overrides.

An overriding method can also return a subtype of the type returned by the overridden method. This subtype is called a *covariant return type*.

When overriding a method, you might want to use the @Override annotation that instructs the compiler that you intend to override a method in the superclass.

If, for some reason, the compiler detects that the method does not exist in one of the superclasses, then it will generate an error.

Method Hiding:

If a subclass defines a static method with the same signature as a static method in the superclass, then the method in the subclass ***hides*** the one in the superclass.

The distinction between hiding a static method and overriding an instance method has important implications:

* The version of the overridden instance method that gets invoked is the one in the subclass.
* The version of the hidden static method that gets invoked depends on whether it is invoked from the superclass or the subclass.

Consider an example that contains two classes. The first is Animal, which contains one instance method and one static method:

public class Animal {

public static void testClassMethod() {

System.out.println("The static method in Animal");

}

public void testInstanceMethod() {

System.out.println("The instance method in Animal");

}

}

The second class, a subclass of Animal, is called Cat:

public class Cat extends Animal {

public static void testClassMethod() {

System.out.println("The static method in Cat");

}

public void testInstanceMethod() {

System.out.println("The instance method in Cat");

}

public static void main(String[] args) {

Cat myCat = new Cat();

Animal myAnimal = myCat;

Animal.testClassMethod();

myAnimal.testInstanceMethod();

}

}

The Cat class overrides the instance method in Animal and hides the static method in Animal. The main method in this class creates an instance of Cat and invokes testClassMethod() on the class and testInstanceMethod() on the instance.

The output from this program is as follows:

The static method in Animal  
The instance method in Cat

As promised, the version of the hidden static method that gets invoked is the one in the superclass, and the version of the overridden instance method that gets invoked is the one in the subclass.

Summary

The following table summarizes what happens when you define a method with the same signature as a method in a superclass.

|  |  |  |
| --- | --- | --- |
| Defining a Method with the Same Signature as a Superclass's Method | | |
|  | **Superclass Instance Method** | **Superclass Static Method** |
| **Subclass Instance Method** | Overrides | Generates a compile-time error |
| **Subclass Static Method** | Generates a compile-time error | Hides |

**Note:** In a subclass, you can overload the methods inherited from the superclass.

Such overloaded methods neither hide nor override the superclass instance methods—they are new methods, unique to the subclass.

Question: What is Method Overriding And Method Hiding in Java?

Super class field will be hidden in the sub class. You can access hidden super class field in sub class using super keyword.

Question: A class member declared protected becomes member of subclass of which type?

A class member declared protected becomes private member of subclass.

Question: How do you restrict a member of a class from inheriting to it’s sub classes?

By declaring that member as a private. Because, private members are not inherited to sub classes.

Question: Are constructors and initializers also inherited to sub classes in Java?

No, Constructors and initializers(Static initializers and instance initializers) are not inherited to sub classes.

But, they are executed while instantiating a sub class.

Question: Are static members inherited to sub classes in Java?

Yes, Static members are also inherited to sub classes.

Question:  What is Super Keyword In Java?

The super keyword in java is a reference variable that is used to refer immediate parent class object.

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

Super keyword has three purposes

* super is used to refer immediate parent class instance variable.
* super() is used to invoke immediate parent class constructor.
* super is used to invoke immediate parent class method.

Question: What are some characteristics of constructors in Super and child classes in Java?

Most important points about constructors are

* Constructors are not inherited.
* If you do not make a constructor, the default empty constructor is automatically created.
* If any constructor does not explicitly call a super or this constructor as its first statement, a call to super() is automatically added.

In Java, constructor of base class with no argument gets automatically called in derived class constructor. For example, output of following program is:

Base Class Constructor Called

Derived Class Constructor Called

// filename: Main.java

class Base {

Base() {

System.out.println("Base Class Constructor Called ");

}

}

class Derived extends Base {

Derived() {

System.out.println("Derived Class Constructor Called ");

}

}

public class Main {

public static void main(String[] args) {

Derived d = new Derived();

}

}

But, if we want to call parameterized contructor of base class, then we can call it using super().

The point to note is base class constructor call must be the first line in derived class constructor.

For example, in the following program, super(\_x) is first line derived class constructor.

// filename: Main.java

class Base {

int x;

Base(int \_x) {

x = \_x;

}

}

class Derived extends Base {

int y;

Derived(int \_x, int \_y) {

super(\_x);

y = \_y;

}

void Display() {

System.out.println("x = "+x+", y = "+y);

}

}

public class Main {

public static void main(String[] args) {

Derived d = new Derived(10, 20);

d.Display();

}

}

Output:

x = 10, y = 20

Question: What is Method Overloading In Java?

Overloading is a process of declaring two methods with same name but different method signature

E.g. System.out which is object of PrintStream class has several println() method to print different data types e.g. byte, short, int, char, float and double.

All of them are called overloaded method. Overloaded method calls are resolved during compile time in Java and they must have different method signatures.

Question: What are rules of overloading a method in Java?

The only rule of method overloading is that method signature of all overloaded method must be different.

Method signature is changed by changing either number of method arguments, or type of method arguments e.g. System.out.println() method is overloaded to accept different primitive types like int, short, byte, float etc.

They all accept just one argument but their type is different.

You can also change method signature by changing order of method argument but that often leads to ambiguous code so better to be avoided.

Question: What is Method Overriding In Java?

Method overriding is another way to define method with same name but different code and it must be in sub class.

Overriding is based upon run-time Polymorphism as method calls are resolved at run-time depending upon actual object.

For example if a variable of type Parent holds an object of Child class then method invoked will be from child class and not parent class, provides its overridden.

In order to override a method, you must follow rules of method overriding which means declaring method with same signature in sub class.

Question: What is Method Hiding In Java?

Static method cannot be overriding in Java because their method calls are resolved at compile time but it didn't prevent you from declaring method with same name in sub class.

In this case we say that method in sub class has hided static method from parent class.

If you have a case where variable of Parent class is pointing to object of Child class then also static method from Parent class is called because overloading is resolved at compile time.

Question: Can you prevent overriding a method without using final modifier?

Yes, there are some funky ways to prevent method overriding in Java. Though final modifier is only for that purpose you can use private keyword to prevent method overriding.

How? If you remember correctly, in order to override a method, the class must be extensible. If you make the constructor of parent class private then its not possible to extend that class because its constructor will not be accessible in sub class.

Which is automatically invoked by sub class constructor, hence its not possible to override any method from that class.

This technique is used in Singleton design pattern, where constructor is purposefully made private and a static getInstance() method is provided to access singleton instance.

Question: Can We Override a Private Method in Java?

No, you cannot override private method in Java. Since private methods are not visible outside the class, they are not available in sub-class hence they cannot be overridden.

Question: What is co-variant Method Overriding?

One of the rule of method overriding is that return type of overriding method must be same as overridden method but this restriction is relaxed little bit from Java 1.5 and now overridden method can return sub class of return type of original method.

This relaxation is known as co-variant method overriding and it allows you to remove casting at client end.

One of the best example of this comes when you override clone() method. Original Object.clone() method returns Object which needs to cast, but with co-variant method overriding you can directly return relevant type

E.g. Date class returns object of java.util.Date instead of java.lang.Object.

Question: Can we change argument list of overridden method?

No, you cannot change the argument list of overridden method in Java. An overriding method must have same signature as original method.

Only return type can be changed that to only to sub type of return type of original method.

Question: Can we change return type of method in subclass while overriding?

No, you cannot change the return type of method during overriding. It would be violation of rules of overriding.

Though from Java 5 onward you can replace the return type with sub type e.g. if original method has return type as java.lang.Object then you can change return type of overridden method as any type e.g. clone() method.

This is also known as co-variant method overriding in Java.

Question: Can we override a method which throws run-time exception without throws clause?

Yes, you can. There is no restriction on throwing RuntimeException from overriding method.

So if your original method throws NullPointerException than its not necessary to throw NPE from overriding method as well.

Question: How to call super class version of an overriding method in sub class?

You can call it using super keyword. For example if you have a method draw() in both parent and child class.

Then from child class you can invoke parent class method draw() as super.draw(). It's very similar to calling super class constructor from sub class.

Question: What are rules of method overriding in Java?

Some rules of method overriding are following:

**Overriding** method cannot throw higher exception than overridden one, but that's only true for checked exception.

**Overriding** method cannot restrict access of overridden method e.g. if original method is public then overriding method must be public.

But it can expand access e.g. if original method is protected than overriding method can be protected or public.

Question: Can we override a non-static method as static in Java?

No, its not possible to define a non-static method of same name as static method in parent class, its compile time error in Java.

Question: Can we override constructor in Java?

No, you cannot override constructor in Java because they are not inherited.

Remember, we are talking about overriding here not overloading, you can overload construct but you cannot override them.

Overriding always happens at child class and since constructor are not inherited and their name is always same as the class name its not possible to override them in Java.

Question: Can we override final method in Java?

No, you cannot override final method in Java. Trying to override final method in subclass will result in compile time error.

Actually making a method final is signal to all developer that this method is not for inheritance and it should be use in its present form.

You generally make a method final due to security reasons.

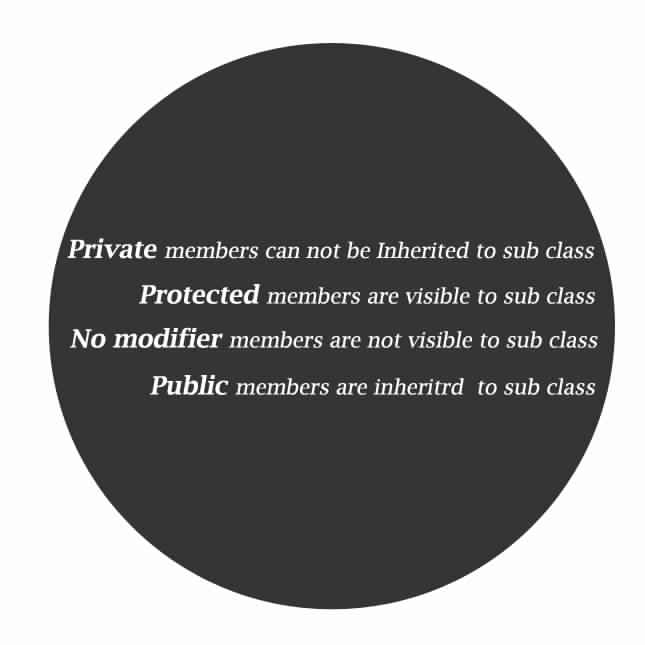
Question: Can you overload or override main method in Java?

Since main() is a static method in Java, it follows rules associated with static method, which means you can overload main method but you cannot override it.

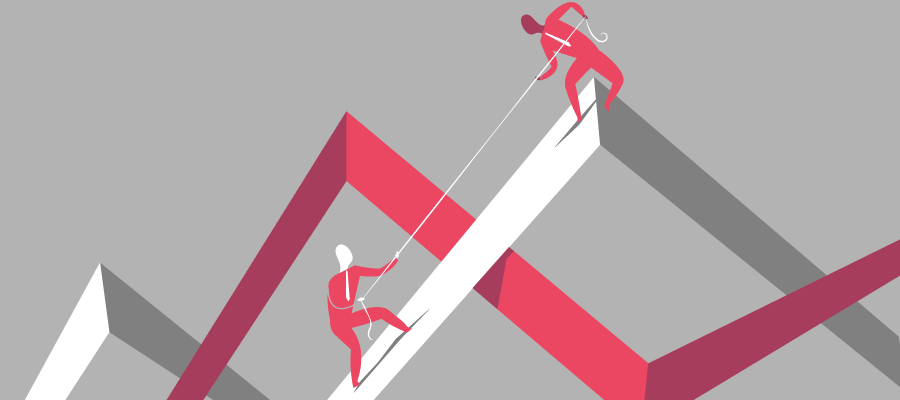
By the way, even if you overload a main method, JVM will always call the standard public static void main(String args[]) method to start your program.

If you want to call your overloaded method you need to do it explicitly in your code.

Question: Access Modifiers  And Inheritance?



Comparison With Other OOP Concepts



Question: What is the difference between Composition and Inheritance?

One of the good question to check the candidate's object-oriented programming skills. There are several differences between Composition and Inheritance in Java, some of them are following:

1. The Composition is more flexible because you can change the implementation at runtime by calling setXXX() method, but Inheritance cannot be changed i.e. you cannot ask a class to implement another class at runtime.
2. Composition builds HAS-A relationship while Inheritance builds IS-A relationship e.g. A Room HAS A Fan, but Mango IS-A Fruit.
3. The parent-child relationship is best represented using Inheritance but If you just want to use the services of another class use Composition. For more differences see 5 reasons to favor composition over Inheritance.

Question: What is the difference between Polymorphism and Inheritance?

Both Polymorphism and Inheritance goes hand on hand, they help each other to achieve their goal.

Polymorphism allows flexibility, you can choose which code to run at runtime by overriding.

Question:  What is the difference between Inheritance and Abstraction in Java?

 Abstraction is an object oriented concept which is used to simply things by abstracting details.

It helps in the designing system. On the other hand, Inheritance allows code reuse. You can reuse the functionality you have already coded by using Inheritance

Question: What is the difference between Inheritance and Encapsulation?

Inheritance is an object oriented concept which creates a parent-child relationship. It is one of the ways to reuse the code written for parent class but it also forms the basis of Polymorphism.

On the other hand, Encapsulation is an object oriented concept which is used to hide the internal details of a class e.g. HashMap encapsulate how to store elements and how to calculate hash values.

Question: Difference between method overloading and overriding?

Fundamental difference between overloading and overriding is that former took place during compile time while later took place during run-time. Due to this reason.

Its only possible to overload virtual methods in Java. You cannot overload methods which are resolved during compile time e.g. private, static and final method cannot be overridden in Java.

Also, rules of method overloading and overriding are different, for example in order to overload a method its method signature must be different but for overriding method it must be same.

Question: Super() vs This() in Java

**Super()**

* Super keyword is used to call constructor in the super class.
* Super always refers to the parent of the current class
* Super allows you to access public/protected method/attributes of parent class. You cannot see the parent's private method/attributes.
* Super allows access to constructors from within the class' constructors only.

**this()**

* this refers to a reference of the current class.
* this allows access methods/attributes of the current class (including its own private methods/attributes).
* this is used to access the methods and fields of the current object. For this reason, it has no meaning in static methods, for example. this keyword use to call constructor in the same class (other overloaded constructor)

Code Review And Inheritance Programs



Question: Following code is showing compile time error. Can you identify the error?

class X

{

//Class X Members

}

class Y

{

//Class Y Members

}

class Z extends X, Y

{

//Class Z Members

}

Answer:

In Java, a class can not extend more than one class. Class Z is extending two classes – Class X and Class Y. It is a compile time error in java.

Question: What will be the output of following program?

class A

{

int i = 10;

}

class B extends A

{

int i = 20;

}

public class MainClass

{

public static void main(String[] args)

{

A a = new B();

System.out.println(a.i);

}

}

Answer:

10

Question: What will be the output of the below program?

class A

{

{

System.out.println(1);

}

}

class B extends A

{

{

System.out.println(2);

}

}

class C extends B

{

{

System.out.println(3);

}

}

public class MainClass

{

public static void main(String[] args)

{

C c = new C();

}

}

Answer:

1 2 3

Question: How to identify super and invocation hierarchy ?

class A

{

String s = "Class A";

}

class B extends A

{

String s = "Class B";

{

System.out.println(super.s);

}

}

class C extends B

{

String s = "Class C";

{

System.out.println(super.s);

}

}

public class MainClass

{

public static void main(String[] args)

{

C c = new C();

System.out.println(c.s);

}

}

Answer:

Class A

Class B

Class C

Question: What is the invocation flow in following example ?

class A

{

static

{

System.out.println("THIRD");

}

}

class B extends A

{

static

{

System.out.println("SECOND");

}

}

class C extends B

{

static

{

System.out.println("FIRST");

}

}

public class MainClass

{

public static void main(String[] args)

{

C c = new C();

}

}

Answer:

THIRD SECOND FIRST

Question: Where super() or this() should be located?

public class A

{

public A()

{

System.out.println(1);

super();

System.out.println(2);

Answer: Constructor calling statements ( super() or this() ), if written, must be the first statements in the constructor.

Question: Can you identify the member visibilty error in this code ?

class X

{

private int m = 48;

}

class Y extends X

{

void methodOfY()

{

System.out.println(m);

}

}

Answer:

Because, private field ‘m’ is not visible to class Y.

Keys to interview success

You must understand that knowing the OOP concepts is the key to [successful interview](http://www.codespaghetti.com/interview-success).

You will be asked questions about inheritance, [abstraction](http://www.codespaghetti.com/abstract-interview-questions/), and [interfaces](http://www.codespaghetti.com/interfaces-interview-questions/). No technical interview is complete without questions on these topics. And the best way to prepare for them is to understand the core concepts.

They are also important in interview assignments or are often asked in the telephonic interviews.

So make sure you have a sound understanding of the concepts presented above.



1. **What is Inheritance?**

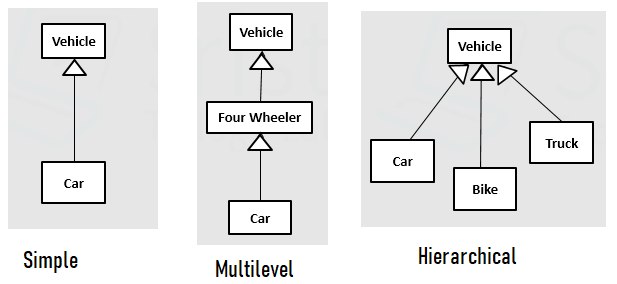
Inheritance is one important concept of OOPs. In Inheritance, one class can inherit the properties(instance variables) and behavior(methods) of another class. Few important terminologies that we use often are given below

**Super Class:**This is the class from which the properties(instance variables) and behavior(methods) are inherited. Also called as base class or parent class  
  
**Sub Class:**This is the class which inherits the properties and behavior from another class. Also called as derived class or child class. The sub class can access all the methods and variables except the private members. It also can have its own instance variables and methods.  
  
**extends:**Use the keyword extends to extend a class. The syntax is as follows  
**class Manager extends Employee**

1. **Why do we need inheritance ?**

Inheritance supports **code reusability**.  
  
Assume we have three classes Vehicle,Car,Bike with similar properties as brand,model,price and method as void getMileage(). Instead of creating the instance variables in both the classes, we can add the variables in just one class which is generic – in this case Vehicle class. Then make the Car and the Bike class inherit the properties and methods of Vehicle class.  
Remember while doing so, there should be an **IS-A** relationship between the classes.That is Car is a Vehicle, Bike is a Vehicle.  
  
Let me talk about a *wrong scenario*. Assume we have two classes as Employee and Student with similar properties as name,age,mobileno, city and method as void printDetails(). Is it right to apply inheritance here?. Try applying**IS-A** rule. Student is an Employee or Employee is a Student. Oh, that sounds weird. So don’t apply inheritance  
  
Remember the rule – Inheritance means **IS-A** relationship and is used for **code reusability**

1. **What type of inheritance does java support?**

Java supports simple, multilevel and hierarchical inheritance.  
  
  
  
  
**Simple:**This is one level of inheritance. One class inherits from another class. In this case, class Car inherits from class Vehicle – **class Car extends Vehicle**  
  
**Multilevel:**This type of inheritance goes down for multiple levels –  
class Car inherits from FourWheeler, class FourWheeler inherits from class Vehicle  
class Vehicle is the sub class of Object class. So, automatically class Car can access the properties of Vehicle class.  
  
**Hierarchical:**Multiple classes can inherit from a single class or A class can have multiple subclasses. In this case, Bike,Car,Truck are sub classes of Vehicle.

1. **What is the super class for all the classes?**

**Object** class from **java.lang package** is the super class for all the classes.

[Become an expert Java Developer](https://academy.shristitechlabs.com/p/java-programming-become-a-java-developer)

1. **Explain inheritance with an example.**

Here is an example for inheritance. I have two classes – **Employee**and **Manager**. Employee is the super class with name,empId as instance variables, void printDetails() as method.  
  
**Employee class**



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16 | public class Employee {  String name;  int empId;    public Employee(String name, int empId) {  super();  this.name = name;  this.empId = empId;  }  void printDetails(){  System.out.println("Name "+name);  System.out.println("EmpId "+empId);  }  } |

Manager is the subclass with salary as instance variable. It uses the properties name,empId of the Employee class. Manager has its own method – double calcBonus(double amount).  
  
**Manager class**



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13 | public class Manager extends Employee {  int salary;    public Manager(String name, int empId, int salary) {  super(name, empId);  this.salary=salary;  }  double calcBonus(double amount){  return salary\*amount;  }  } |

Let us create a main class Demo with main() method. Create an object of Employee and you can call printDetails() method. Create object of Manager. Using manager object you can call the methods of Manager and also of the super class Employee.

**Demo class**



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16 | public class Demo {    public static void main(String[] args) {    System.out.println("welcome");  Employee employee = new Employee("Ram",100);  employee.printDetails();    Manager manager = new Manager("Raja",200,4000);  manager.printDetails();  System.out.println("Bonus: "+manager.calcBonus(2));  }    } |

1. **What is the use of super keyword?**

**‘super’** keyword should be always in the first line of any constructor. If it is not added by you, it will added automatically by the compiler. The purpose of super is to call and initialize the super class instance variables first. Always the super class instance variables are initialized first and then the subclass variables are initialized.  
Remember the parameters within **super()** should match atleast one constructor of the super class.

1. **Can a class extend more than one class?**

No. Java does not support multiple inheritance. Please refer question-3. However, whatever classes you create will become a sub class of Object class

1. **Can I access the sub class methods using a super class object?**

No. You cannot access the subclass methods/properties using the super class object.Only the sub class object can access the properties of the super class.(one real example – The son/daughter can access the fathers’ mobile. But the father cannot access the son/daughters’ mobile)

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

Inheritance means **IS-A** relationship. Association means **HAS-A**relationship. For example **Car is a Vehicle** is Inheritance. **Car has an Engine property** becomes Association.Let me show this as java code



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11 | public class Vehicle{    }  public class Car extends Vehicle{ //Inheritance    private Engine engine;// This is Association. HAS-A relationship  }  public class Engine{    } |

1. **What is the difference between Inheritance and Polymorphism?**

Inheritance and Polymorphism complement each other. Runtime Polymorphism, which also called as Overriding happens with inheritance support. That is overriding happens only in a super class, subclass scenario.

If you have some more questions relevant to Inheritance, then do share with us. Also learn about the [top questions and answers in overriding](https://www.shristitechlabs.com/java/interviewquestions/top-10-interview-questions-in-overriding-runtime-polymorphism/).

**1) What do you mean by inheritance.?**

Inheritance is one of the key features of object oriented programming. Through inheritance, a class (Sub Class) can inherit properties of another class (Super Class). Sub class can have it’s own properties along with the inherited properties from it’s super class.

**2) What are the types of inheritance.?**

There are 5 types of inheritance.

1). Single Inheritance : One class is extended by only one class.

2). Multilevel Inheritance : One class is extended by a class and that class in turn is extended by another class thus forming a chain of inheritance.

3). Hierarchical Inheritance : One class is extended by many classes.

4).Hybrid Inheritance : It is a combination of above types of inheritance.

5). Multiple Inheritance : One class extends more than one classes. (Java does not support multiple inheritance.)

**3) Can a class extend more than one classes or does java support multiple inheritance? If not, why?**

No, a class in java can not extend more than one classes or java does not support multiple inheritance. To avoid ambiguity, complexity and confusion, java does not supports multiple inheritance. For example, If Class C extends Class A and Class B which have a method with same name, then Class C will have two methods with same name. This causes ambiguity and confusion for which method to use. To avoid this, java does not supports multiple inheritance.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22 | class A  {      void methodOne()      {          System.out.println("From methodOfClassA");      }  }    class B  {      void methodOne()      {          System.out.println("From methodOfClassB");      }  }    class C extends A,B (If it is supported)  {      //two same methods will be inherited to Class C.        //This causes ambiguity and confusion.  } |

**4) How do you implement multiple inheritance in java?**

Through interfaces, we can implement multiple inheritance in java. As classes in java can not extend more than one classes, but a class can implement more than one interfaces.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14 | interface A  {    }    interface B  {    }    class C implements A, B  {      //Class implementing two interfaces.  } |

**5) You know that all classes in java are inherited from java.lang.Object class. Are interfaces also inherited from Object class.?**

No, only classes in java are inherited from Object class. Interfaces in java are not inherited from Object class. But, classes which implement interfaces are inherited from Object class.

**6) How do you restrict a member of a class from inheriting to it’s sub classes.?**

By declaring that member as a private. Because, private members are not inherited to sub classes.

**7) Can a class extend itself.?**

No, A class can not extend itself.

**8) Are constructors and initializers also inherited to sub classes.?**

No, Constructors and initializers(Static initializers and instance initializers) are not inherited to sub classes. But, they are executed while instantiating a sub class.

**9) What happens if both, super class and sub class, have a field with same name.?**

Super class field will be hidden in the sub class. You can access hidden super class field in sub class using super keyword.

**10) Are static members inherited to sub classes?**

Yes, Static members are also inherited to sub classes.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24 | class A  {      static int i = 10;        static void method()      {          System.out.println("Static Method");      }  }    class B extends A  {    }    public class StaticInitializers  {      public static void main(String[] args)      {          B.method();       //Calling inherited static method            System.out.println(B.i);    //printing inherited static field.      } |

**Q1. What is Inheritance in Java?**

**Answer:** Inheritance in Java or OOPS (Object oriented programming) is a feature which allows coding reusability.  In other words, Inheritance self-implies inheriting or we can say acquiring something from others. Along with Abstraction, Encapsulation and Polymorphism, Inheritance forms the backbone of Object oriented programming and Java.  In Java, we use the term inheritance when one object acquires some property from other objects. In Java, inheritance is defined in terms of superclass and subclass. it is normally used when some object wants to use existing feature of some class and also want to provide some special feature, so we can say inheritance has given the advantage of reusability.

**Q2. What are different types of Inheritance supported by Java?**

**Answer:**Java supports different types of inheritance which is supported by Java.

* Single Inheritance.
* Multiple Inheritance (Through Interface)
* Multilevel Inheritance.
* Hierarchical Inheritance.
* Hybrid Inheritance (Through Interface)

 Java allows a class to only extend another class, but an interface in Java can extend multiple inheritances.

**Q3. Why Inheritance is used by Java?**

**Answer:**In Java, inheritance is used when a class wants to use/inherit the features of another existing class. The class that wants to use the feature of another class, is called subclass, whereas the class whose features are to be used is referred to as superclass. Inheritance is used to use the existing features of a class.

**Q4. What is syntax of inheritance?**

**Answer:**

public class subclass extends superclass{

//all methods and variables declare here

}

**Q5. How Inheritance can be implemented in java**

**Answers**

• Inheritance can be implemented in JAVA using below two keywords:

1.extends

2.implements

• extends is used for developing inheritance between two classes and two interfaces. • implements keyword is used to developed inheritance between interface and class

**Q6. What is multilevel inheritance?**

**Answers:**

Getting the properties from one class object to another class object level wise with different priorities.

**Q7. What is Multiple inheritance? why Java Doesn’t Support multiple Inheritance?**

**Answers:**

* The concept of Getting the properties from multiple class objects to sub class object with same priorities is known as multiple inheritance.
* In multiple inheritance there is every chance of multiple properties of multiple objects with the same name available to the sub class object with same priorities leads for the ambiguity. also known as diamond problem. one class extending two super classes.
* Because of multiple inheritance there is chance of the root object getting created more than once.
* Always the root object i.e object of object class hast to be created only once.
* Because of above mentioned reasons multiple inheritance would not be supported by java.
* Thus in java a class can not extend more than one class simultaneously. At most a class can extend only one class.

**Q8. How do you restrict a member of a class from inheriting to it’s sub classes.?**

**Answers**

By declaring that member as a private. Because, private members are not inherited to sub classes.

**Q9. How do you implement multiple inheritance in java?**

**Answer**

Using interfaces java can support multiple inheritance concept in java. in java can not extend more than one classes, but a class can implement more than one interfaces.

**Program**:

interface A{

}

interface B{

}

class C implements A,B{

**Q10. What is the difference between Inheritance and composition in Java?**

**Answer:**

Java supports both composition and inheritance but both are different in many ways. Composition objects have a reference to the composition classes by having a loosely bounded relationship and it has a – has a relationship between classes. Using this we can use single classes can be composed within multiple classes and used in dependency injection and composition is a relationship between objects whereas Inheritance is a relationship between the classes and it has is a relationship between the classes and using inheritance derived class object carries the base class definition hence it is a tightly bounded relationship. It is mostly used in run time polymorphism and in this a single class can only inherit only one class.

**Java Inheritance Multiple Choice Questions**

**Q11. Which inheritance in java programming is not supported?**

1. Multiple inheritance using classes
2. Multiple inheritance using interfaces
3. Multilevel inheritance
4. Single inheritance

**Answer: A**

NOTE: Java does not support multiple inheritance of classes but it supports multiple inheritance for interfaces. Means, a class cannot inherit more than one class but it can inherit and implement multiple interfaces.

**Q12. What is subclass in java?**

1. A subclass is a class that extends another class
2. A subclass is a class declared inside a class
3. Both above.
4. None of the above

**Answer:** A

A subclass is a class that extends another class. In other words, subclass inherits the functionality of the class it extends. See below example.

class BaseClass {

public void foo() {

System.out.println(“Base class”);

}

}

//This is a subclass that inherit

//public methods of base class

class SubClass extends BaseClass {

}

public class Program {

public static void main(String[] args) {

SubClass s = new SubClass();

         s.foo();

}

}

**Output:**

Base class

**Q13. If class B is subclassed from class A then which is the correct syntax?**

1. **class B:A{}**
2. **class B extends A{}**
3. **class B extends class A{}**
4. **class B implements A{}**
5. **View Answer**

**Answer: B**

Below is the example

class A{  
  
}

//Class B is subclassed from class A  
class B extends A{  
}

**Q14. Which of this keyword must be used to inherit a class?**

a) super

b) this

c) extent

d) extends

**View Answer**

Answer: d

Explanation: None.

**Q15. A class member declared protected becomes a member of subclass of which type?**

a) public member

b) private member

c) protected member

d) static member

**View Answer**

**Answer: b**

**Explanation:**A class member declared protected becomes a private member of subclass.

**Q16. Which two classes use the Shape class correctly?**

**A**.

public class Circle implements Shape {

private int radius;

}

**B**.

public abstract class Circle extends Shape

{

private int radius;

}

**C**.

public class Circle extends Shape

{

private int radius;

public void draw();

}

﻿

**D**.

public abstract class Circle implements Shape

{

private int radius;

public void draw();

}

**E**.

public class Circle extends Shape

{

private int radius;

public void draw()

{

/\* code here \*/

}

}

**F**.

public abstract class Circle implements Shape {

private int radius;

public void draw()

{

/\* code here \*/

}

}

**Options:**  
a) B,E

b) A,C

c) C,E

d) T,H

**View Answer**

**Answer: a**

**Explanation:**If one is extending any class, then they should use extends keyword not implements.

**Q17. What will be the output of the following Java program?**

class A {

int i;

}

class B extends A

{

int j;

void display()

{

super.i = j + 1;

System.out.println(j + " " + i);

}

}

class inheritance {

public static void main(String args[]) {

B obj = new B();

obj.i=1;

obj.j=2;

obj.display();

}

}

﻿

**Options**

**a)** 2 2

**b)** 3 3

**c)**2 3

**d)** 3 2

**View Answer**

**Answer: c**

**Q18. What will be the output of the below program?**

class A{

{

System.out.println(1);

}

}

class B extends A

{

{

System.out.println(2);

}

}

class C extends B{

{

System.out.println(3);

}

}

public class MainClass{

public static void main(String[] args) {

C c = new C();

}

}

**Answer:**

1  
2  
3

**Q19. What will be the output of the following program?**

class A{

String s = "Class A";

}

class B extends A{

String s = "Class B";

{

System.out.println(super.s);

}

}

class C extends B{

String s = "Class C";

{

System.out.println(super.s);

}

}

public class MainClass{

public static void main(String[] args) {

C c = new C();

System.out.println(c.s);

}

}

**Answer:**  
Class A  
Class B  
Class C

**Q20. What will be the output of the following program?**

class X

{

static void staticMethod()

{

System.out.println("Class X");

}

}

class Y extends X

{

static void staticMethod()

{

System.out.println("Class Y");

}

}

public class MainClass

{

public static void main(String[] args)

{

Y.staticMethod();

}

}

﻿

**Answers:** Class Y

**The Best Java Inheritance Interview Questions And Answers**

**Q21. Can we override static method in Java?  
  
Answer:**  
No, you cannot override a static method in Java because it’s resolved at compile time. In order for overriding to work, a method should be virtual and resolved at runtime because objects are only available at runtime. This is one of the tricky Java questions, where interviewer tries to confuse you. A programmer is never sure about whether they can override or overload a static method in Java.

**Q22. Can we overload a static method in Java?**

**Answer:**

Yes, you can overload a static method in Java. Overloading has nothing to do with runtime but the signature of each method must be different. In Java, to change the method signature, you must change either number of arguments, type of arguments or order of arguments.

**Q23. Can an interface extends more than one interface in Java?**

**Answer:**Yes, unlike classes, an interface can extend more than one interface in Java. There are several example of this behavior in JDK itself e.g. java.util.List interface extends both Collection and Iterable interface to tell that it is a Collection as well as it allows iteration via Iterator.

**Q24.  Can a class extend itself.?**

**Answer:**

No, A class can not extend itself.

**Q25. Are constructors and initializers also inherited to sub classes.?**

**Answer:**

No, Constructors and initializers(Static initializers and instance initializers) are not inherited to sub classes. But, they are executed while instantiating a sub class.

**Q26. What happens if both, super class and sub class, have a field with same name.?**

**Answer:**

Super class field will be hidden in the sub class. You can access hidden super class field in sub class using super keyword.

**Q27. What is Method Hiding?**

**Answer:  
Method Hiding:**

If a subclass defines a static method with the same signature as a static method in the superclass, then the method in the subclass hides the one in the superclass.

The distinction between hiding a static method and overriding an instance method has important implications:

* The version of the overridden instance method that gets invoked is the one in the subclass.
* The version of the hidden static method that gets invoked depends on whether it is invoked from the superclass or the subclass.

**Q28. Are constructors and initializers also inherited to sub classes in Java?**

**Answers:**No, Constructors and initializers(Static initializers and instance initializers) are not inherited to sub classes. But, they are executed while instantiating a sub class.

**Q29. What is Method Overloading In Java?**

**Answer:** Overloading is a process of declaring two methods with same name but different method signature

E.g. System.out which is object of PrintStream class has several println() method to print different data types e.g. byte, short, int, char, float and double.

All of them are called overloaded method. Overloaded method calls are resolved during compile time in Java and they must have different method signatures.

**Q30. What are rules of overloading a method in Java?**

**Answer:**

The only rule of method overloading is that method signature of all overloaded method must be different.

Method signature is changed by changing either number of method arguments, or type of method arguments e.g. System.out.println() method is overloaded to accept different primitive types like int, short, byte, float etc.

They all accept just one argument but their type is different. You can also change method signature by changing order of method argument but that often leads to ambiguous code so better to be avoided.