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

Ans: The technique of creating a new class by using an existing class functionality is called inheritance in Java. In other words, inheritance is a process where a child class acquires all the properties and behaviors of the parent class.

**2. Why do we need to use inheritance?**  
Or, What is the purpose of using inheritance?

Ans: Inheritance is one of the main pillars of OOPs concept. Some objects share certain properties and behaviors. By using inheritance, a child class acquires all properties and behaviors of parent class.

There are the following reasons to use inheritance in java.

* We can reuse the code from the base class.
* Using inheritance, we can increase features of class or method by overriding.
* Inheritance is used to use the existing features of class.
* It is used to achieve runtime polymorphism i.e method overriding.

**3. What is Is-A relationship in Java?**

Ans: Is-A relationship represents Inheritance. It is implemented using the “extends” keyword. It is used for code reusability.

**4. What is super class and subclass?**

Ans: A class from where a subclass inherits features is called superclass. It is also called base class or parent class.

A class that inherits all the members (fields, method, and nested classes) from other class is called subclass. It is also called a derived class, child class, or extended class.

**5. How is Inheritance implemented/achieved in Java?**

Ans: Inheritance can be implemented or achieved by using two keywords:

1. **extends:** extends is a keyword that is used for developing the inheritance between two classes and two interfaces.
2. **implements:** implements keyword is used for developing the inheritance between a class and interface.

**6. Write the syntax for creating the subclass of a class?**

Ans: A subclass can be created by using the “extends” keyword. The syntax for declaring a subclass of class is as follows:

class subclassName extends superclassName

{

// Variables of subclass

// Methods of subclass

}

where class and extends are two keywords.

**7. Which class in Java is superclass of every other class?**

Ans: In Java, Object class is the superclass of every other class.

**8. How will you prove that the features of Superclass are inherited in Subclass?**

Ans: Refer to this tutorial: [Inheritance in Java with Realtime Example](https://www.scientecheasy.com/2020/07/inheritance-in-java.html/)

**9. Can a class extend itself?**

Ans: No, a class cannot extend itself.

**10. Can we assign superclass to subclass?**

Ans: No.

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

Ans: No, one class can extend only a single class.

**12. Are constructor and instance initialization block inherited to subclass?**

Ans: No, constructor and instance initialization block of the superclass cannot be inherited to its subclass but they are executed while creating an object of the subclass.

**13. Are static members inherited to subclass in Java?**

Ans: Static block cannot be inherited to its subclass.

A static method of superclass is inherited to the subclass as a static member and non-static method is inherited as a non-static member only.

**14. Can we extend (inherit) final class?**

Ans: No, a class declared with final keyword cannot be inherited.

**15. Can a final method be overridden?**

Ans: No, a final method cannot be overridden.

**16. Can we inherit private members of base class to its subclass?**

Ans: No.

**17. What is order of calling constructors in case of inheritance?**

**Ans: In case of inheritance, constructors are called from the top to down hierarchy.**

**18. Which keyword do you use to define a subclass?**  
Or, which keyword is used to inherit a class?

Ans: extends keyword.

**19. What are the advantages of inheritance in Java?**

Ans: The advantages of inheritance in java are as follows:

* We can minimize the length of duplicate code in an application by putting the common code in the superclass and sharing it amongst several subclasses.
* Due to reducing the length of code, the redundancy of the application is also reduced.
* Inheritance can also make application code more flexible to change.

**20. What are the types of inheritance in Java?**

Ans: The various types of inheritance are as follows:

a. Single inheritance  
b. Multi-level inheritance  
c. Hierarchical inheritance  
d. Multiple inheritance  
e. Hybrid inheritance

**21. What are the various forms of inheritance available in Java?**

Ans: The various forms of inheritance to use are single inheritance, hierarchical inheritance, and multilevel inheritance.

**22. What is single inheritance and multi-level inheritance?**

Ans: When one class is extended by only one class, it is called single level inheritance. In single-level inheritance, we have just one base class and one derived class.

A class which is extended by a class and that class is extended by another class forming chain inheritance is called multilevel inheritance.

**23. What is Multiple inheritance in Java?**

Ans: A class that has many superclasses is known as multiple inheritance. In other words, when a class extends multiple classes, it is known as multiple inheritance.

**24. Why multiple inheritance is not supported in java through class?**  
Ans:

Multiple inheritance means that one class extends two superclasses or base classes but in Java, one class cannot extend more than one class simultaneously. At most, one class can extend only one class.

Therefore, to reduce ambiguity, complexity, and confusion, Java does not support multiple inheritance through classes.

For more detail, go to this tutorial: [Types of Inheritance in Java](https://www.scientecheasy.com/2020/07/types-of-inheritance-in-java.html/)

**25. How does Multiple inheritance implement in Java?**

Ans: Multiple inheritance can be implemented in Java by using interfaces. A class cannot extend more than one class but a class can implement more than one interface.

**26. What is Hybrid inheritance in java? How will you achieve it?**

Ans: A hybrid inheritance in java is a combination of single and multiple inheritance. It can be achieved through interfaces.

**27. How will you restrict a member of a class from inheriting its subclass?**

Ans: We can restrict members of a class by declaring them private because the private members of superclass are not available to the subclass directly. They are only available in their own class.

**28. Can we access subclass members if we create an object of superclass?**

Ans: No, we can access only superclass members but not the subclass members.

**29. Can we access both superclass and subclass members if we create an object of subclass?**

Ans: Yes, we can access both superclass and subclass members.

**30. What happens if both superclass and subclass have a field with the same name?**

Ans: Only subclass members are accessible if an object of subclass is instantiated.

**31. Will the code successfully compiled? If yes, what is the output?**

public class A {

int x = 20;

}

public class B extends A {

int x = 30;

}

public class Test {

public static void main(String[] args)

{

B b = new B();

System.out.println(b.x);

A a = new A();

System.out.println(a.x);

A a2 = new B();

System.out.println(a2.x);

}

}

Ans: Yes, code will be successfully compiled. The output is 30, 20, 20.

Practice the best collection of [**inheritance example program in Java**](https://www.scientecheasy.com/2019/01/java-inheritance-example-program.html/) with output and explanation for Java interviews.

**32. Which of the following is correct way of inheriting class A by class B?**

a) class B + class A { }  
b) class B inherits class A { }  
c) class B extends A { }  
d) class B extends class A { }

Ans: c

**33. Is interface inherited from the Object class?**

Ans: No.

**34. What is the output of executing code of class Test?**

public class A {

void m1() {

System.out.println("m1 in class A");

}

}

public class B extends A {

void m1() {

System.out.println("m1 in class B");

}

}

public class Test {

public static void main(String[] args)

{

B b = new B();

b.m1();

A a = new A();

a.m1();

A a2 = new B();

a2.m1();

}

}

Ans: The result is m1 in class B, m1 in class A, m1 in class B.

**34. Will this code run successfully? If yes, what will be the output?**

public class A

{

private int x = 50;

void m1() {

System.out.println(x);

}

}

public class B extends A {

}

public class Test {

public static void main(String[] args)

{

A a = new B();

a.m1();

}

}

Ans: Yes, code will be successfully compiled. The output will be 50.

**35. Will this code compile successfully? If yes, what is the output? If no, identify the errors.**

package pack1;

public class A

{

private int x = 50;

protected int y = 100;

int z = 200;

}

package pack2;

import pack1.A;

public class B extends A {

}

import pack2.B;

public class Test {

public static void main(String[] args)

{

B b = new B();

System.out.println(b.x);

System.out.println(b.y);

System.out.println(b.z);

}

}

Ans: No, the code will not compile successfully because of two compile-time errors.

* First error is in line System.out.println(b.x); because private members cannot be accessed in the subclass.
* Second error is in line System.out.println(b.y); because default members of superclass can be accessed in the subclass within the same package only.

**36. How will you prohibit inheritance in Java?**

Ans: If you declare a class as final, it cannot be extended. Thus, we can prohibit the inheritance of that class in Java.

**37. In the following code, we have created 3 Classes A, B, and C. Class C extends Class B and Class B extends Class A.**

**Each class has a method m1(), is there any way to call A’s m1() method from Class C?**

public class A

{

void m1(){

System.out.println("m1 in class A");

}

}

public class B extends A

{

void m1() {

System.out.println("m1 in class B");

}

}

public class C extends B

{

void m1() {

System.out.println("m1 in class C");

}

}

public class Test {

public static void main(String[] args)

{

C c = new C();

c.m1();

}

}

Ans: In the above code, every class has m1() method with the same signature thus Class B is overriding A’s m1() method and Class C is overriding Class B’s m1() method.

Now, it is possible for class B and class C to call their super class’s m1() method by using super.m1() call.

But here, the question is asking to invoke A’s m1() method from Class C, which is not possible because it violates the OOPs concept in Java and also not super.super in java.

Since Java does not allow multiple inheritance, that means C can see only one superclass which will have just one m1() method implementation. C can never see A’s m1() method.

We also call this scenario as the Diamond Problem of Multiple Inheritance. The only way to call A’s m1() method from Class C is if Class B and class C call super.m1() method in its m1() implementation.

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

public class A

{

void m1() {

System.out.println("m1 in class A");

}

}

public class B extends A

{

void m1() {

super.m1();

System.out.println("m1 in class B");

}

}

public class C extends B

{

void m1() {

System.out.println("m1 in class C");

}

}

public class Test {

public static void main(String[] args)

{

C c = new C();

c.m1();

}

}

Ans: Output: m1 in class C

**39. Find out the output of the following program.**

public class A

{

void m1() {

System.out.println("m1 in class A");

}

}

public class B extends A

{

void m1() {

super.m1();

System.out.println("m1 in class B");

}

}

public class C extends B

{

void m1() {

System.out.println("m1 in class C");

super.m1();

}

}

public class Test {

public static void main(String[] args)

{

C c = new C();

c.m1();

}

}

Ans: Output: m1 in class C, m1 in class A, m1 in class B

**40. Identify the error in the following code.**

final class A

{

void m1() {

System.out.println("m1 in class A");

}

}

public class B extends A

{

public static void main(String[] args)

{

B b = new B();

}

}

Ans: A class declared with final keyword cannot be extended.

**41. Will the code compile successfully? If not, identify the error in the following code.**

class A

{

int x = 50;

private A() {

System.out.println(x);

}

}

public class B extends A

{

public static void main(String[] args)

{

B b =new B();

}

}

Ans: No, the code will not compile successfully because the constructor is private, the class cannot be inherited.

**42. What problem will arise when the following code is compiled?**

class A {

public A(int a) {

}

}

public class B extends A {

public B() {

}

}

public class Test {

public static void main(String[] args)

{

B b = new B();

}

}

Ans: Java compiler will generate compile-time error: Implicit super constructor A() is undefined. Must explicitly invoke another constructor

**43. How many ways to implement relationships among classes in Java?**

Ans: There are four ways to make relationships among classes. They are as follows:

a. Association  
b. Aggregation  
c. Composition  
d. Inheritance

**44. What is Association in OOPs concept?**

Ans: Association in Java is one of the core concepts of OOPs. It establishes the relation between two classes that are independent of one another.

In association, all objects have their own life cycle and there is no ownership between objects. Association can be unidirectional or bidirectional.

**45. In the OOPs concept, what is meant by Aggregation?**

Ans: Aggregation in java represents has-a relationship. It is a special form of association that represents an ownership relation between two objects.

In aggregation, two aggregated objects have their own life cycle but one of the objects is the owner of has-a relationship.

**46. What is meant by Composition in OOPs?**

Ans: Composition in java is one of the core concepts of OOPs and a more restrictive case of aggregation. It represents has-a relationship that contains an object and cannot exist on its own.

In simple words, if a class object holds an object of another class, it is called composition. It establishes strong relationship between two objects than aggregation.

**47. Which is more tightly bound? Inheritance or Composition?**

Ans: Composition is more tightly bound than inheritance.

**48. What is the difference between Inheritance and Composition?**

Ans: Inheritance represents Is-A relationship between subclass and its superclass whereas, composition represents Has-A relationship between classes.

**49. When will you prefer composition over inheritance?**

Ans: Inheritance can be used only when there is a perfect IS-A relationship between the superclass and subclass definitions. But in case of any confusion, prefer composition over inheritance.

**50. How aggregation and composition both are different concepts?**

Ans: In the OOPs concept, aggregation and composition both are types of association relations. A composition establishes a strong relationship between classes.

If the composite object is destroyed, all its parts are destroyed. For example, a car has a steering wheel. If car object is destroyed, there is no meaning to be the existence of steering wheel.

Aggregation establishes a weaker relationship between classes than composition. For example, a library has students. If a library is destroyed, students still exist. Hence, library and student are related by aggregation.