- **what is inheritance?**

• inheritance is one of the oops concepts in java. inheritance is concept of getting properties of one class object to another class object.

• Inheritance represents the IS-A relationship, also known as parent-child relationship.

- **Why we need to use Inheritance?**

1.For Code Re usability.
2.For Method Overriding.

- **Which of these keyword must be used to inherit a class?**

a) super
b) this
c) extent
d) extends

Answer: d Explanation: None.

- **what are the types of inheritance?**

1.Multiple inheritance(java doesn't support multiple inheritance). 2.Multilevel inheritance.

- **what is syntax of inheritance?**

```
public class subclass extends superclass{
//all methods and variables declare here
}
```

- **What is the output of this program?**

```
class A {
 int i;
 void display() {
 System.out.println(i);
  }
 }

 class B extends A {
 int j;
 void display() {
 System.out.println(j);
  }
 }

 class inheritance_demo {
 public static void main(String args[])
 {
```

```
B obj = new B();
obj.i=1;
obj.j=2;
obj.display();
}
    }
```

a) 0
b) 1
c) 2
d) Compilation Error

Answer: c Explanation: class A & class B both contain display() method, class B inherits class A, when display() method is called by object of class B, display() method of class B is executed rather than that of Class A.

output:

```
$ javac inheritance_demo.java
$ java inheritance_demo 2
```

- **How Inheritance can be implemented in java?**

• 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.

- **what is multilevel inheritance?**

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

- **what is Multiple inheritance? why Java Doesn't Support multiple Inheritance.**

• 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.

• lways 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.

- **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.

- **Which of these keywords is used to refer to member of base class from a sub class?**

a) upper
b) super
c) this
d) None of the mentioned

Answer: b
Explanation: whenever a subclass needs to refer to its immediate superclass, it can do so by use of the keyword super.

- **How do you implement multiple inheritance in java?**

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{
}
```

- **What is the output of this 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();
 }
    }
```

a) 2 2 b) 3 3 c) 2 3 d) 3 2

Answer: c
Explanation: None

output:

```
$ javac inheritance.java
$ java inheritance
2 3
```

- **Can a class extend itself?**

No,A class can't extend itself.

- **What happens if super class and sub class having same field name?**

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

- **Does Java support Multiple Inheritance ?**

No, Java doesn't support multiple inheritance. Interfaces doesn't facilitate inheritance and hence implementation of multiple interfaces doesn't make multiple inheritance.

- **Are constructors inherited? Can a subclass call the parent's class constructor? When?**

You cannot inherit a constructor. That is, you cannot create a instance of a subclass using a constructor of one of it's superclasses. One of the main reasons is because you probably don't want to override the superclasses constructor, which would be possible if they were inherited. By giving the developer the ability to override a superclasses constructor you would erode the encapsulation abilities of the language.

- **What is the output of this program?**

```
class A {
 public int i;
```

```
    private int j;
    }
    class B extends A {
    void display() {
    super.j = super.i + 1;
    System.out.println(super.i + " " + super.j);
    }
    }
    class inheritance {
    public static void main(String args[])
    {
    B obj = new B();
    obj.i=1;
    obj.j=2;
    obj.display();
    }
}
```

a) 2 2 b) 3 3 c) Runtime Error d) Compilation Error

d)Compilation Error Explanation: class contains a private member variable j, this cannot be inherited by subclass B and does not have access to it. output: $ javac inheritance.java Exception in thread "main" java.lang.Error: Unresolved compilation problem: The field A.j is not visible

- **Why java doesn't support multiple Inheritance ?**

```
class A {
void test() {
System.out.println("test() method");
}
}

class B {
void test() {
System.out.println("test() method");
}
}

Suppose if Java allows multiple inheritance like this,
class C extends A, B {
    }
```

A and B test() methods are inheriting to C class. So which test() method C class will take? As A & B class test() methods are different , So here we would Facing Ambiguity.

- **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.

- **What is the output of this program?**

```
class A {
 public int i;
 public int j;
 A() {
 i = 1;
 j = 2;
  }
 }
 class B extends A {
 int a;
 B() {
 super();
  }
 }
 class super_use {
 public static void main(String args[])
 {
 B obj = new B();
 System.out.println(obj.i + " " + obj.j)
  }
     }
```

a) 1 2 b) 2 1 c) Runtime Error d) Compilation Error

- **Can we reduce the visibility of the inherited or overridden method ?**

No.

- **A class member declared protected becomes member of subclass of which type?**

a) public member
b) private member

c) protected member
d) static member

Answer: b Explanation: A class member declared protected becomes private member of subclass.

- **Which of the following is tightly bound ? Inheritance or Composition ?**

Inheritence.

- **Difference Between this() and super() ?**

1.this is a reference to the current object in which this keyword is used whereas super is a reference used to access members specific to the parent Class. 2.this is primarily used for accessing member variables if local variables have same name, for constructor chaining and for passing itself to some method whereas super is primarily used to initialize base class members within derived class constructor.

- **What is the output of this program?**

```
class A {
 public int i;
 protected int j;
  }
 class B extends A {
 int j;
 void display() {
 super.j = 3;
 System.out.println(i + " " + j);
  }
 }
 class Output {
 public static void main(String args[])
 {
 B obj = new B();
 obj.i=1;
 obj.j=2;
 obj.display();
  }
     }
```

a) 1 2
b) 2 1
c) 1 3
d) 3 1

Answer: a Explanation: Both class A & B have member with same name that is j, member of class B will be called by default if no specifier is used.

I contains 1 & j contains 2, printing 1 2.

output:

```
$ javac Output.java
$ java Output
1 2
```

- **What will happen if class implement two interface having common method?**

That would not be a problem as both are specifying the contract that implement class has to follow. If class C implement interface A & interface B then Class C thing I need to implement print() because of interface A then again Class think I need to implement print() again because of interface B, it sees that there is already a method called test() implemented so it's satisfied.

- **Does a class inherit the constructor of its super class?**

No

- **What are points to consider in terms of access modifier when we are overriding any method?**

1. Overriding method can not be more restrictive than the overridden method.

reason : in case of polymorphism , at object creation jvm look for actual runtime object. jvm does not look for reference type and while calling methods it look for overridden method.

If by means subclass were allowed to change the access modifier on the overriding method, then suddenly at runtime—when the JVM invokes the true object's version of the method rather than the reference type's version then it will be problematic

2. In case of subclass and superclass define in different package, we can override only those method which have public or protected access.

3. We can not override any private method because private methods can not be inherited and if method can not be inherited then method can not be overridden.

- **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.)

- **what is covariant return type?**

co-variant return type states that return type of overriding method can be subtype of the return type declared in method of superclass. it has been introduced since jdk 1.5

- **How compiler handles the exceptions in overriding ?**

1)The overriding methods can throw any runtime Exception , here in the case of runtime exception overriding method (subclass method) should not worry about exception being thrown by superclass method.

2)If superclass method does not throw any exception then while overriding, the subclass method can not throw any new checked exception but it can throw any runtime exception

3) Different exceptions in java follow some hierarchy tree(inheritance). In this case , if superclass method throws any checked exception , then while overriding the method in subclass we can not throw any new checked exception or any checked exception which are higher in hierarchy than the exception thrown in superclass method

- **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.

```
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.
}
```

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

No, A class can not extend itself.

- **Which of these 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 {}

- **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.

- **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.

```
interface A
{

}

interface B
{

}

class C implements A, B
{
//Class implementing two interfaces.
}
```

- **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.

- **Are static members inherited to sub classes?**

Yes, Static members are also inherited to sub classes.

```
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.
}
}
```

- **Which of the following statements are incorrect? a) public members of class can be accessed by any code in the program. b) private members of class can only be accessed by other members of the class. c) private members of class can be inherited by a sub class, and become protected members in sub class. d) protected members of a class can be inherited by a sub class, and become private members of the sub class.**

Answer: c Explanation: private members of a class cannot be inherited by a sub class.