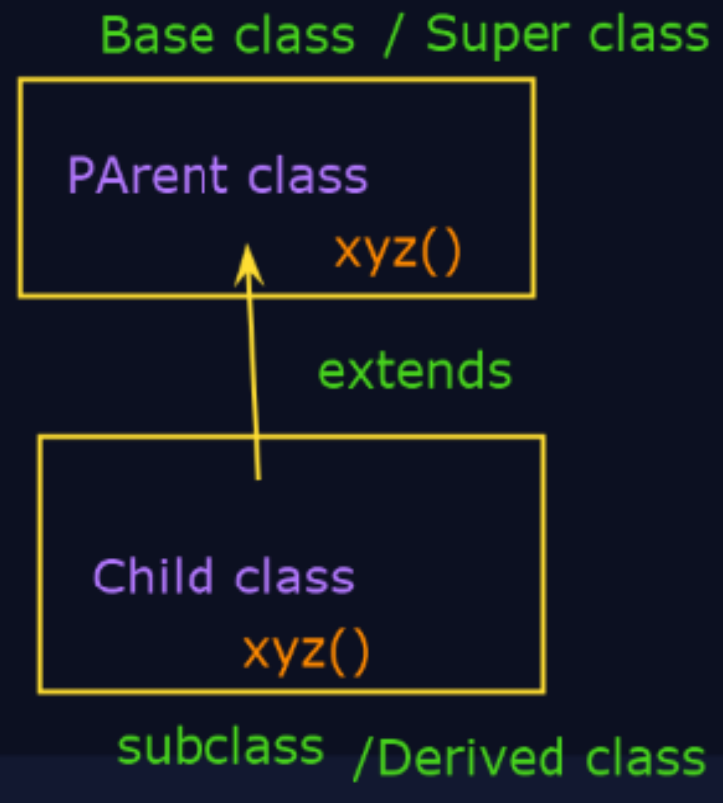
Object Oriented Programming with Java (OOPJ)

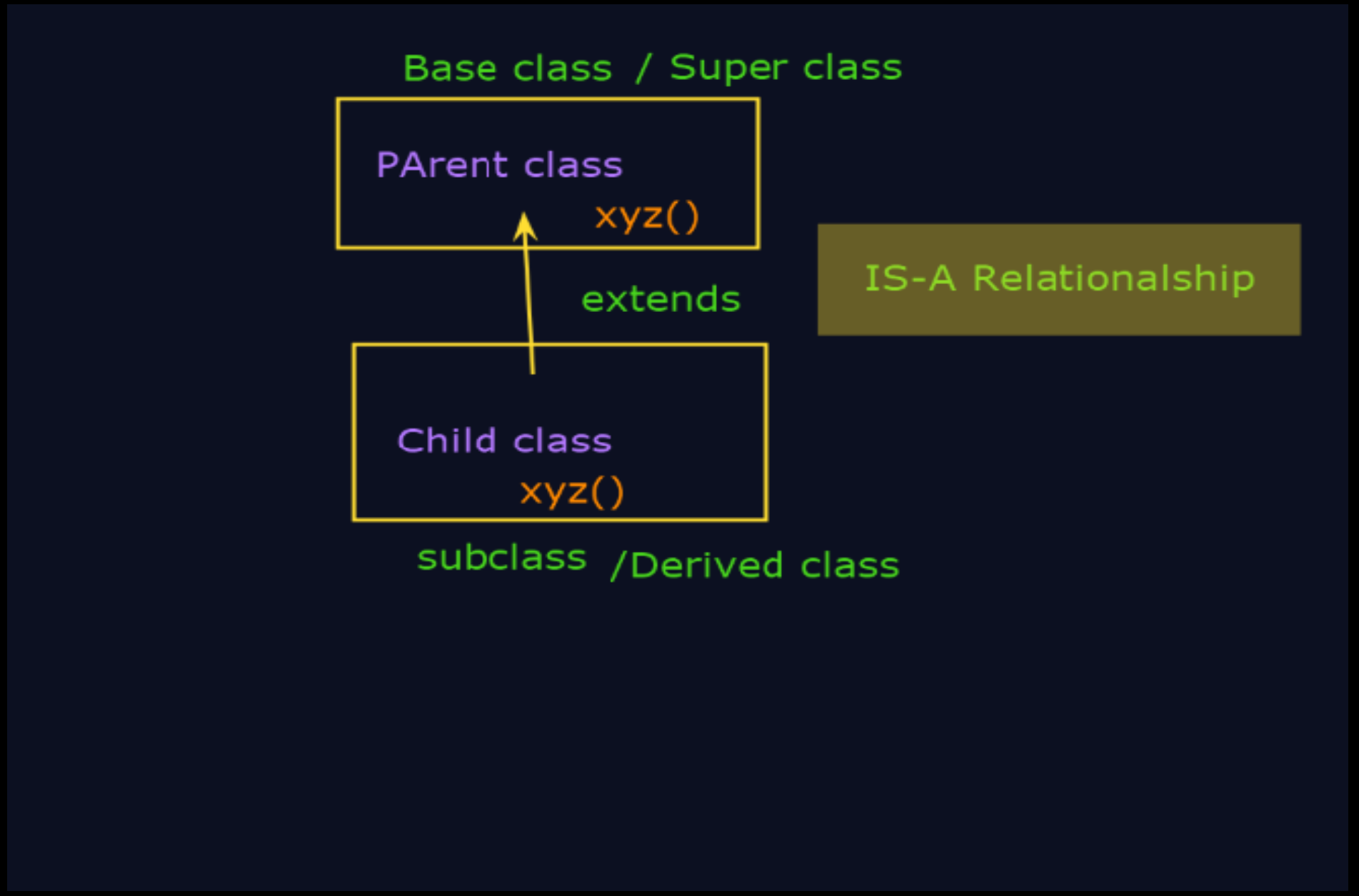
Session 8: OOPS Pillar

Kiran Waghmare

```
Java command:
set JAVA_HOME=C:\Program Files\Java\jdk-21
set PATH=%JAVA_HOME%\bin;%PATH%
------------------------------
OOPJ Module
-----------------
Meeting Id: 81940226085
Key: 12345
------------------------------
Date: 03/09/2025
Day 8 : OOPJ
------------------------------
Topics-
    -Abstraction
    -Encapsulation
    -Inheritance
    -Polymorphism
```

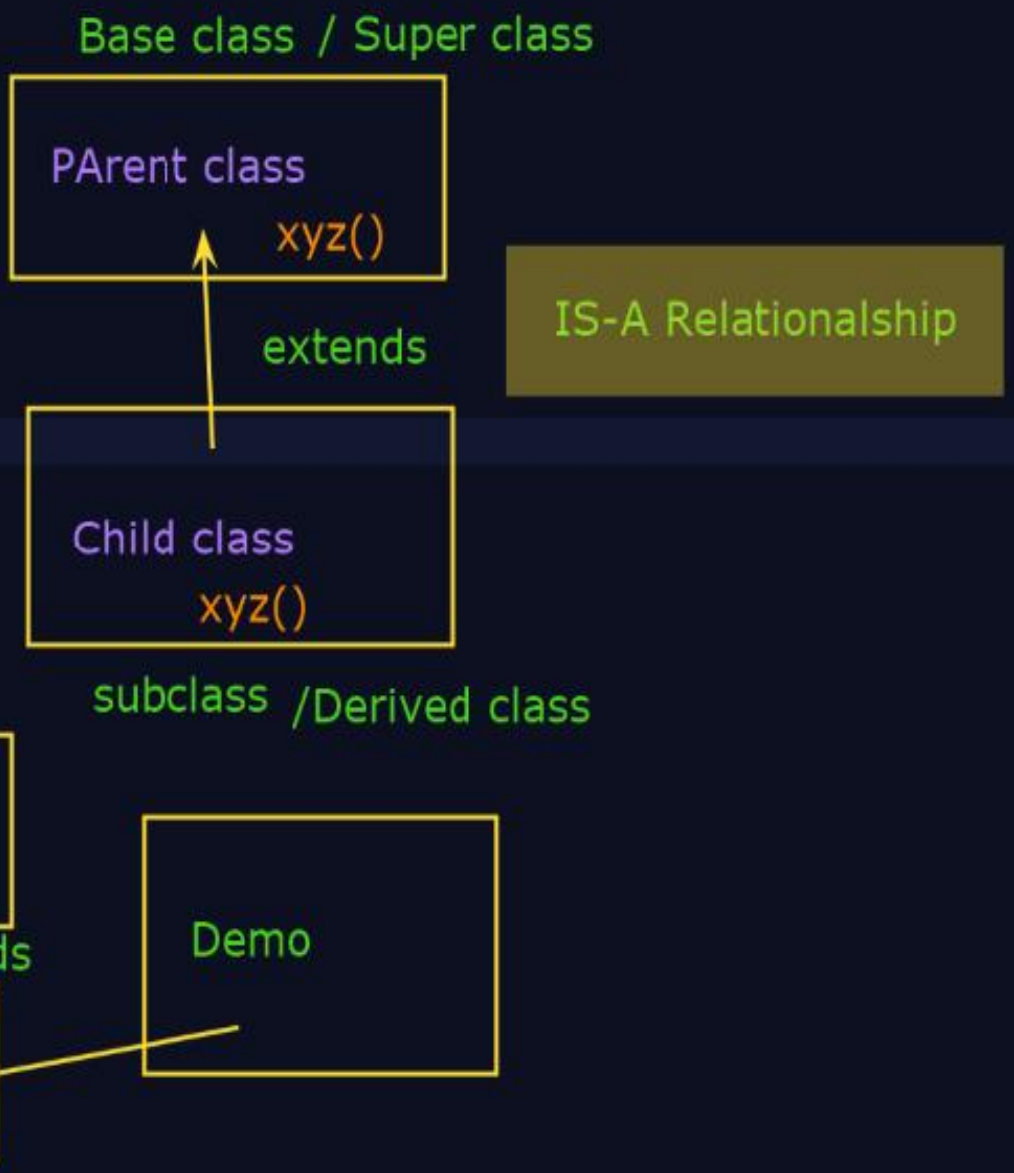




```
class: A blueprint for creating objects.
Super class (Parent class) : The class whose properties are inherited
Sub class (child class) : The class that inherits properties from another class
Reusability: Using existing methods and filelds of a super class in a subclass.
```

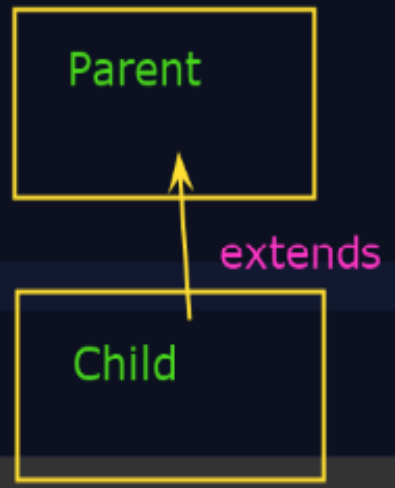

Base class / Super class
PArent class
xyz()
extends
IS-A Relationalship
Child class
xyz()
subclass /Derived class

Syntax:

```
class Parent{
}


class Child extends Parent{
}


class Demo extends Child{
}
```



```
    int b=20;

    void PrintData(){
        System.out.println("Parent class : P");
        System.out.println("a="+a);
        System.out.println("b="+b);
    }

}

class Child extends Parent{
    int x =30;
    void PrintChildData(){
        System.out.println("Childclass : C1");
        System.out.println("x="+x);
    }
}

public class SingleInheritanceDemo{

    public static void main(String args[]){

        Child c = new Child();
        c.PrintChildData();
        c.PrintData();

    }
}
```




```
C:\WINDOWS\syster
Childclass : C1
Parent class : P

C:\Test>javac SingleInheritanceDemo.java

C:\Test>java SingleInheritanceDemo
Childclass : C1
x=30
Parent class : P
a=10
b=20

C:\Test>
```

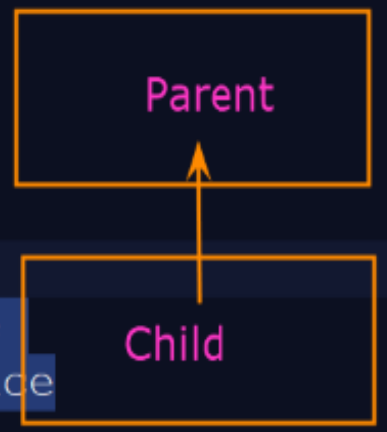
SingleInheritanceDemo

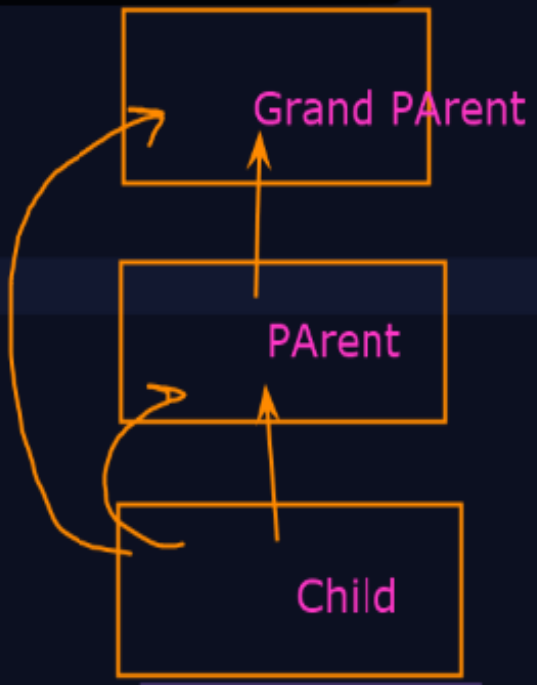
```
class Demo extends Child{
}
```

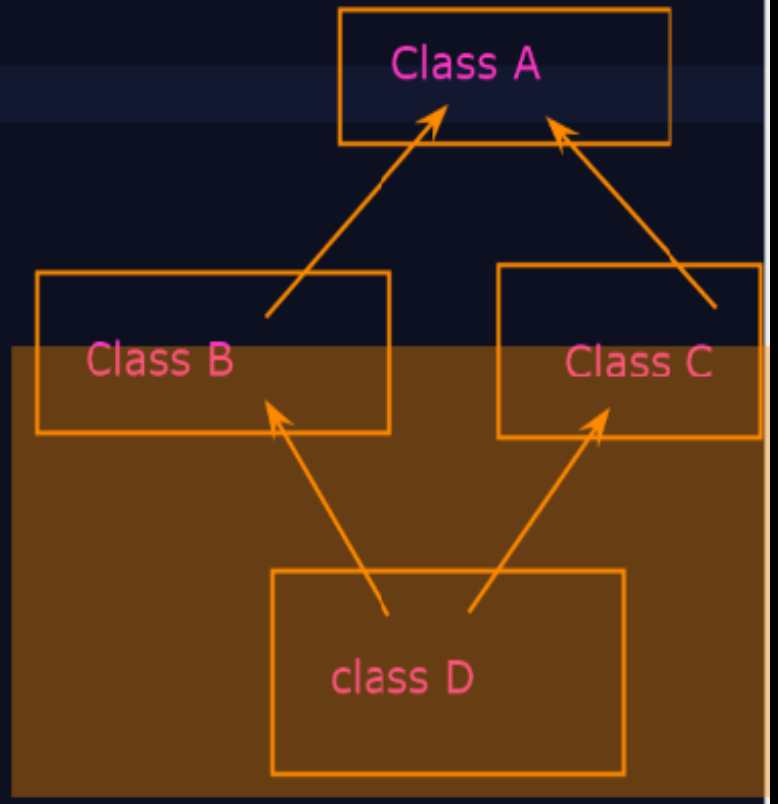
Types of Inheritance:

---

1. Single Inheritance
2. Multilevel Inheritance
3. Hierarchical Inheritance





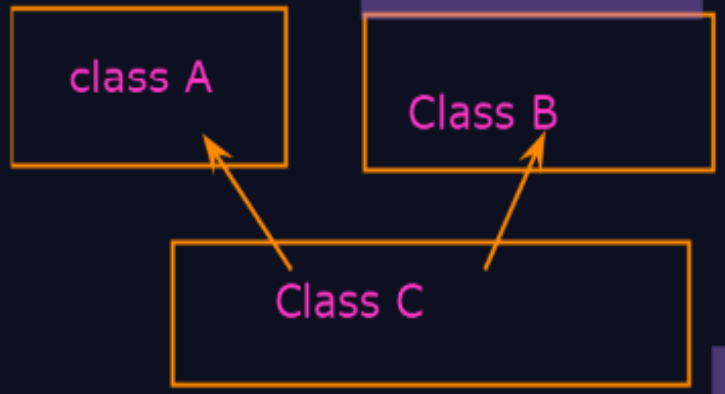


Hybrid Inheritance

Single Inheritance

Hierarchical Inheritance

Multilevel Inheritance



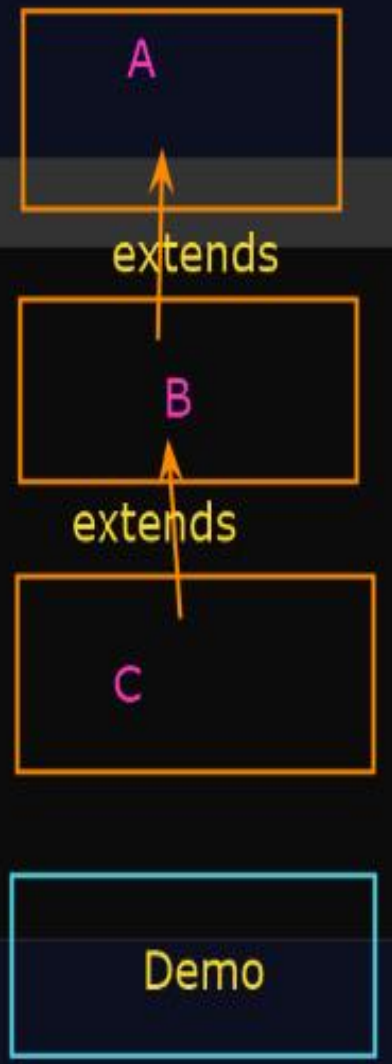
Multiple Inheritance

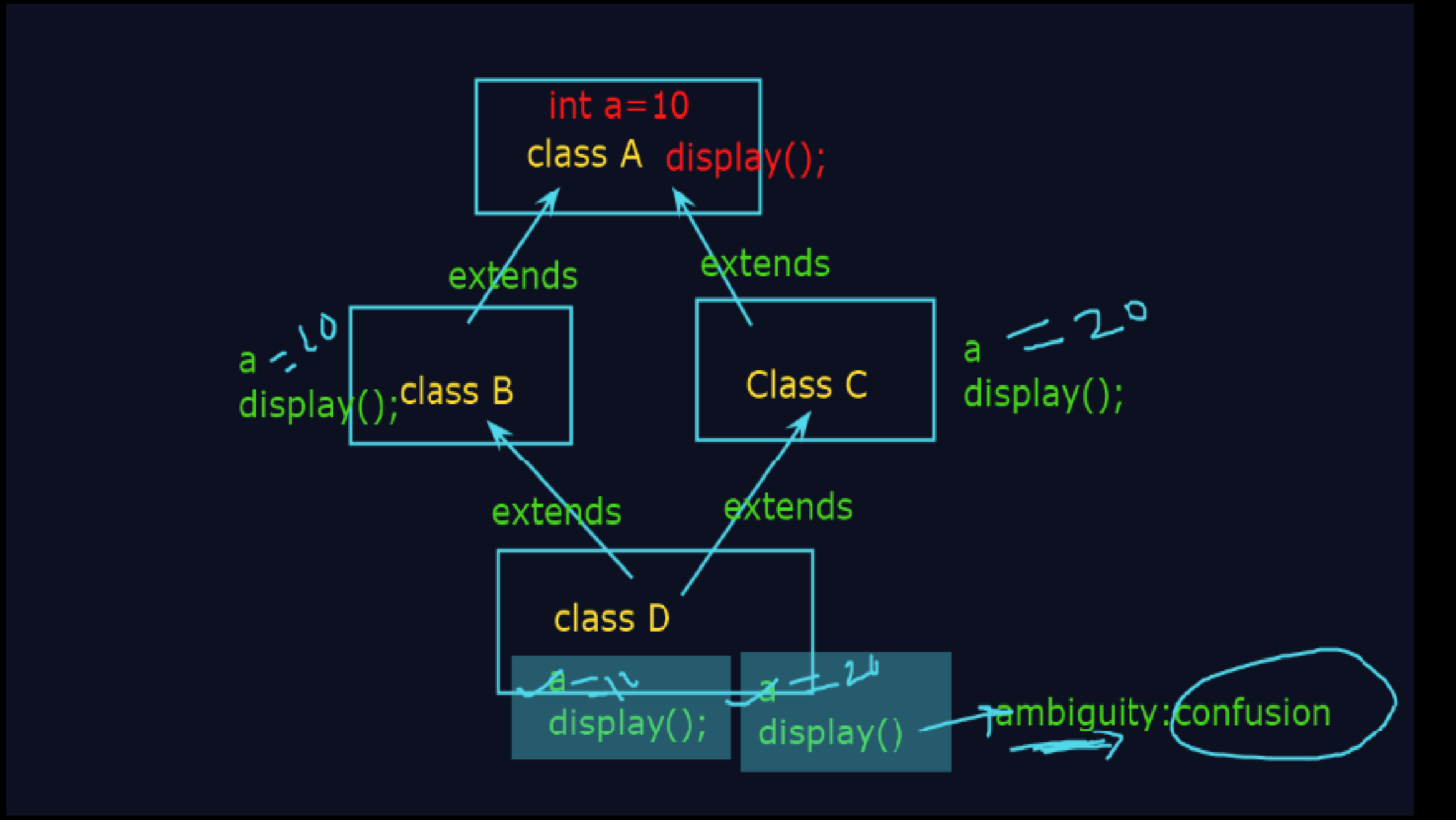
Interfaces

```java
public class MultipleInheritanceDemo1{

    public static void main(String args[]){
        A a1 = new A();
        a1.showA();

        B b1 = new B();
        b1.showB();// method of B cla
        b1.showA();//inherited method

        C c1 = new C();
        c1.showC();//method of C clas
        c1.showB();//inherited method
        c1.showA();

    }
}
```

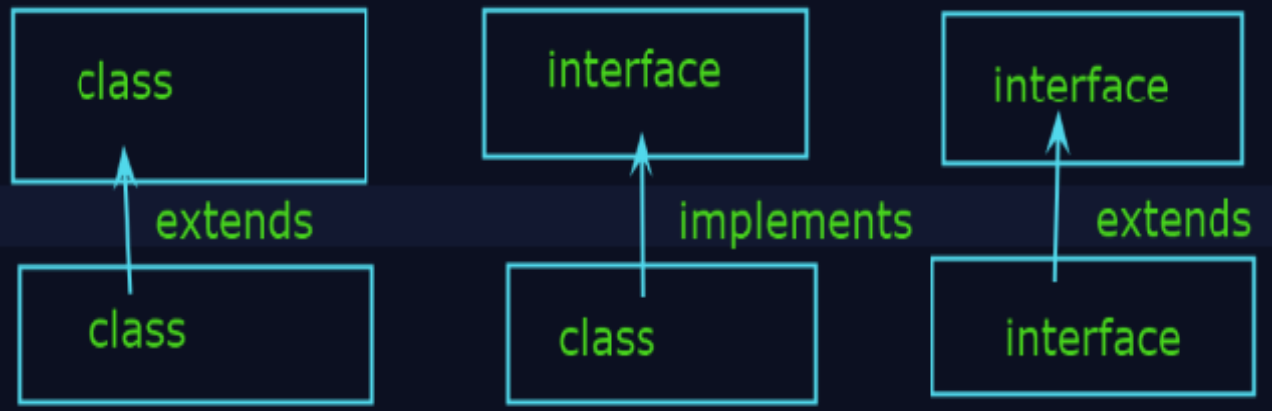
You are screen sharing Stop share

```
C:\WINDOWS\syster
Class A method: showA()

C:\Test>javac MultipleInheritanceDemo1.java

C:\Test>java MultipleInheritanceDemo1
Class A method: showA()
Class B method: showB()
Class A method: showA()
Class C method: showC()
Class B method: showB()
Class A method: showA()

C:\Test>
```

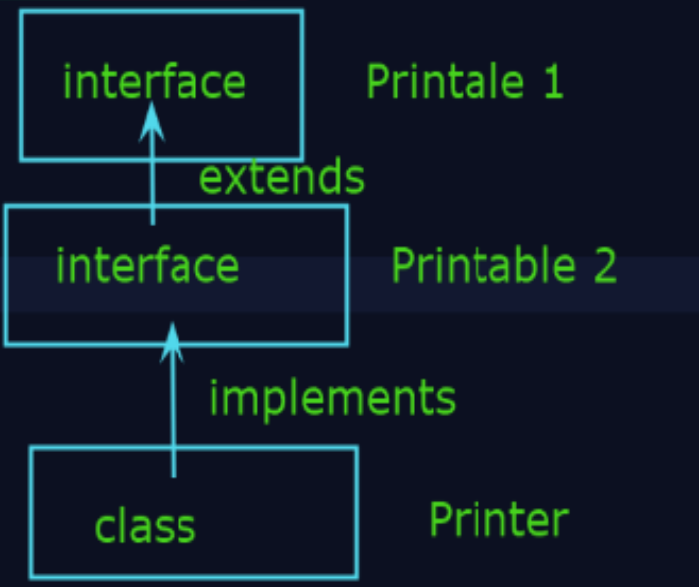



int a=10
class A display();
extends
extends
a =10
display();
class B
Class C
a =20
display();
extends
extends
class D
a =10
display();
a =20
display()
ambiguity:confusion

```
}

interface Book{

    //constant variable
    int x=100;
    //abstract method
    void read();
    //default method: Java 8+
    void show(){
    SOP()
    }
    //static method : Java 9+
    static void display(){
        SOP();
    }
    //Private methods : Java 9+
    private void print(){
    }

}
```



```java
interface Printable1{
    void print();//abstract method

}

interface Printable2 extends Printable1{

    void show();//abstract method
}

class Printer implements Printable2 {
    public void print(){
        System.out.println("Printing data....");

    }
    public void show(){
        System.out.println("Display content....");

    }
}

public class InterfaceDemo{
```

InterfaceDemo

2. Inheritance is manadatory
-method overriding can occur when subclass inherits parent class

3. Access modifiers
private
default
protected
public
Ex: parent class= public : child class : public
Ex: parent class= protected : child class :protected , public
Ex: parent class= default : child class : default, protected , public

Important Note:

-private: cannot inherit
-satic: can be redefined but not overridden
-final: cannot be override