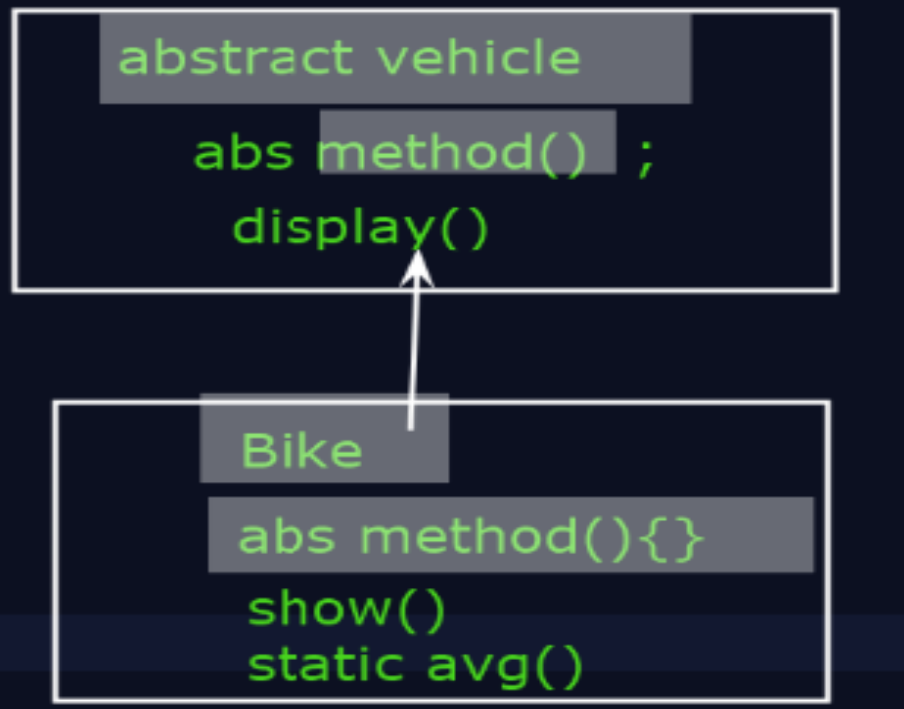
Object Oriented Programming with Java (OOPJ)

Session 5: Arrays

Kiran Waghmare

```
    abstract void eat();
}

Reference :
1.
Employee e = new Employee();

2.
Employee e = null;

3.
Employee e = new childclass();
```



127_Umesh Shewale_KH raised hand View ×

```java
class Employee{
    float salary = 40000;//Parent class

}

class Programmer extends Employee{
    int bonus = 10000;//child class

}
class SingleInheritanceDemo{
    public static void main(String args[]){

        Employee e = new Programmer();
        System.out.println("Sal = "+e.salary);
        System.out.println("Bonus = "+e.bonus);

        Programmer e1= new Programmer();
        System.out.println("Sal = "+e1.salary);
        System.out.println("Bonus = "+e1.bonus);
    }
}
//Programmer class inherits the salary(inherit) and bonus field.
//Employee class  salary
```

```
}
```

Types of Inheritance:

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

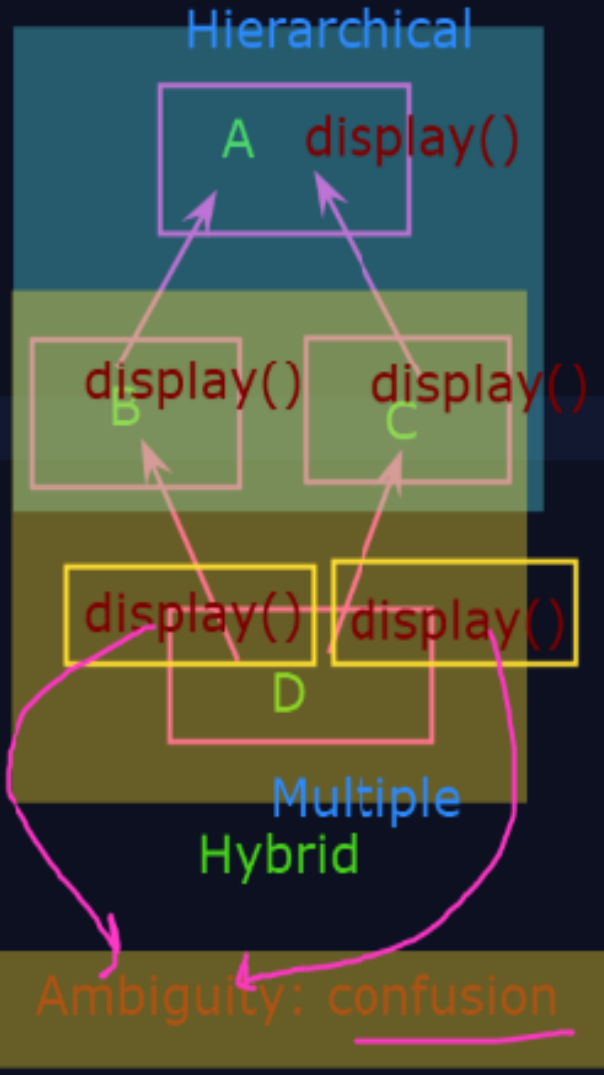
---

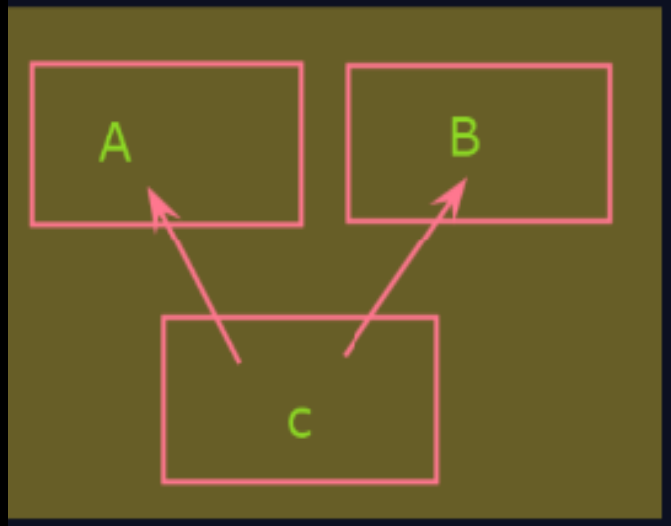
4. Multiple Inheritance
5. Hybrid Inheritance

Single

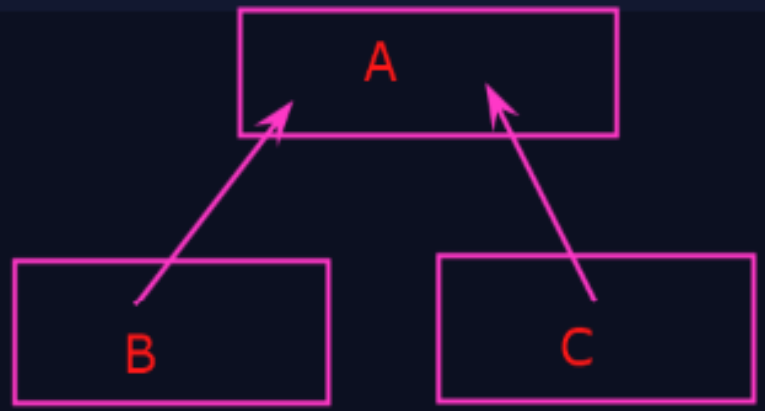
Multilevel

Diamond problem:
Inheriatnce



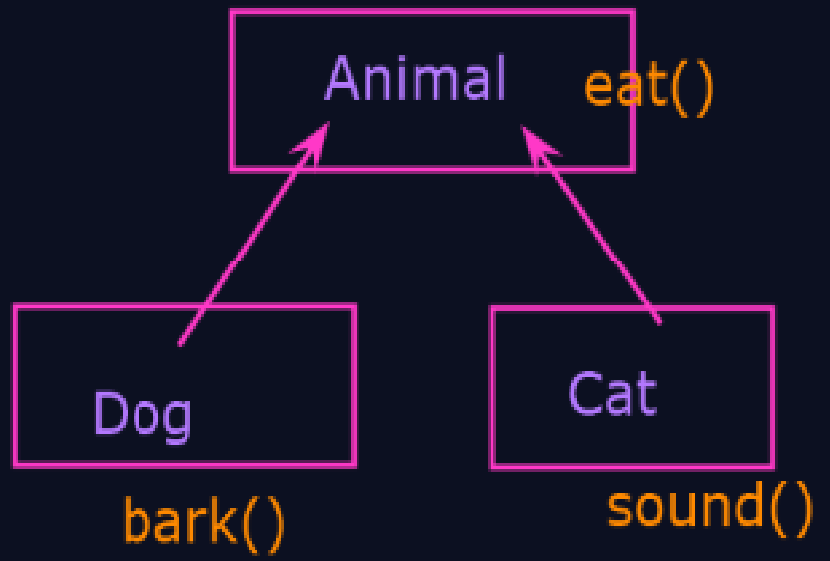


Multiple : Solution(interface)



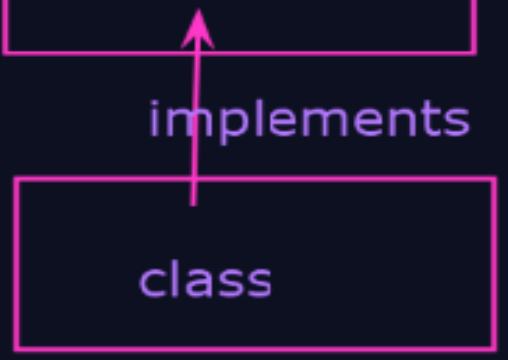
Hierarchical

```
    //Additional fields and methods: Child
}


Types of Inheritance:
----------------------
1. Single Inheritance
2. Multilevel Inheritance
3. Hierarchical Inheritance
---------------------------
4. Multiple Inheritance
5. Hybrid Inheritance

Multiple Inheritance : Not supported in Java:
-----------------------------------------------
class A
```



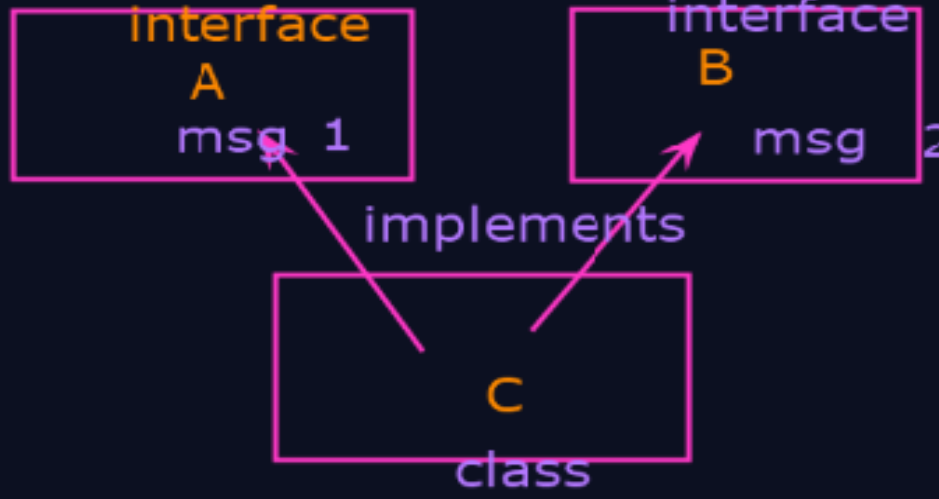

interface


extends
interface


implements
class


interface


implements
class


interface
A
msg 1
interface
B
msg 2
implements
C
class


class
class

Interfaces:

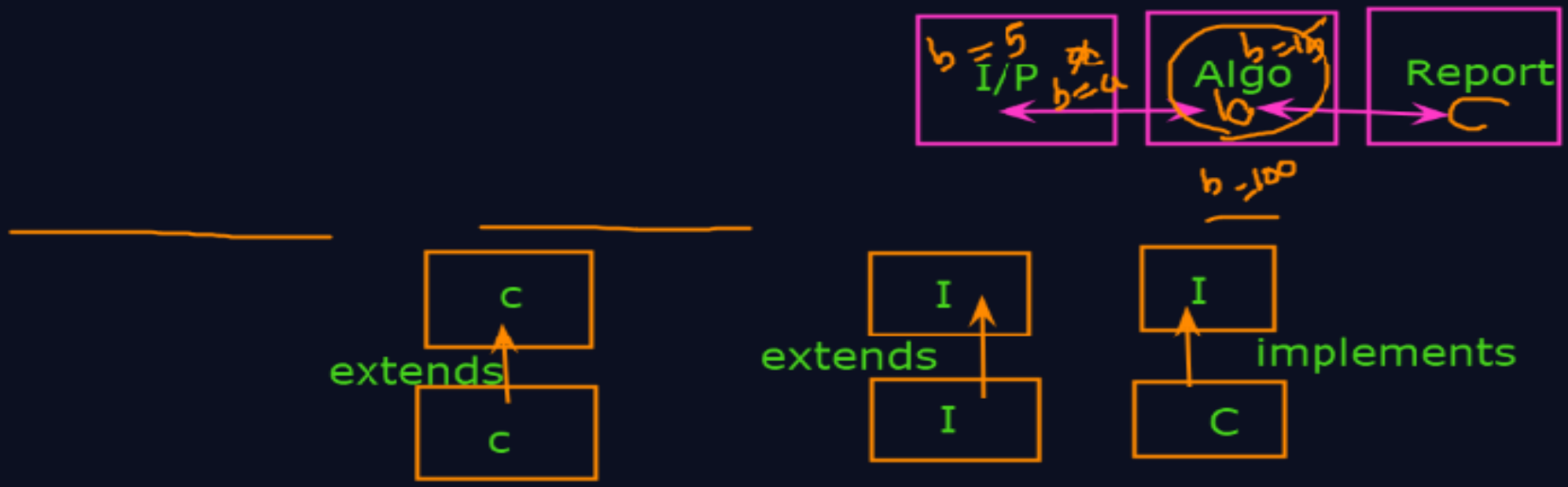
-An interface in Java is a blueprint of a class.
-It contains only abstract methods and constants by default (before
-It cannot have a method body, only method declarations.
-Interfaces are used to achieve abstraction and multiple inheritanc
-Java 8: Interfaces can have default and static methods.
-Java 9: Interfaces can have private methods.

Why to use interfaces in Java to:

-Achieve abstraction
-Enable multiple inheritance (behavior)
-Support loose coupling & polymorphism
-Create standard contracts
-Allow flexible implementations
-Make code scalable, maintainable, and extensible

Why to use interfaces in Java to:

-Achieve abstraction
-Enable multiple inheritance (behavior)
-Support loose coupling & polymorphism
-Create standard contracts
-Allow flexible implementations
-Make code scalable, maintainable, and extensible



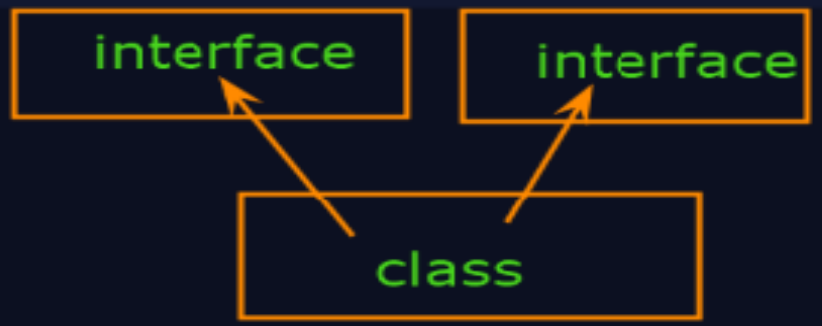
Why to use interfaces in Java to:

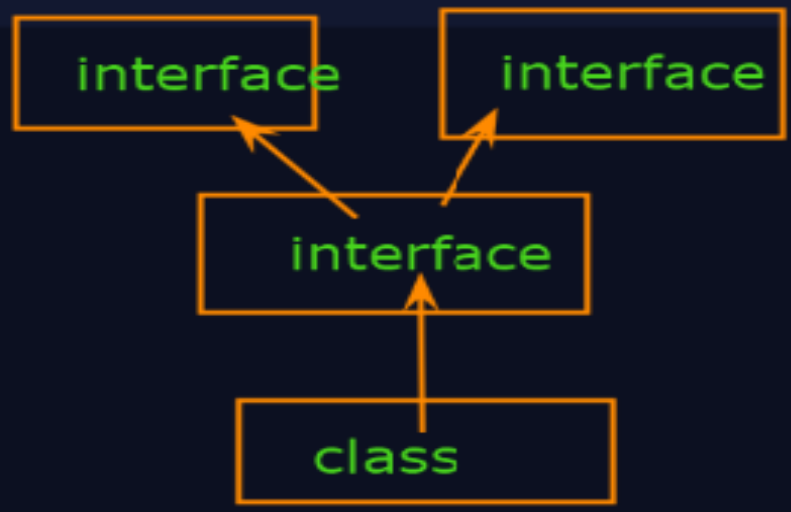
-Achieve abstraction
-Enable multiple inheritance (behavior)
-Support loose coupling & polymorphism
-Create standard contracts
-Allow flexible implementations
-Make code scalable, maintainable, and extensible

Relationship between Classes and Interfaces

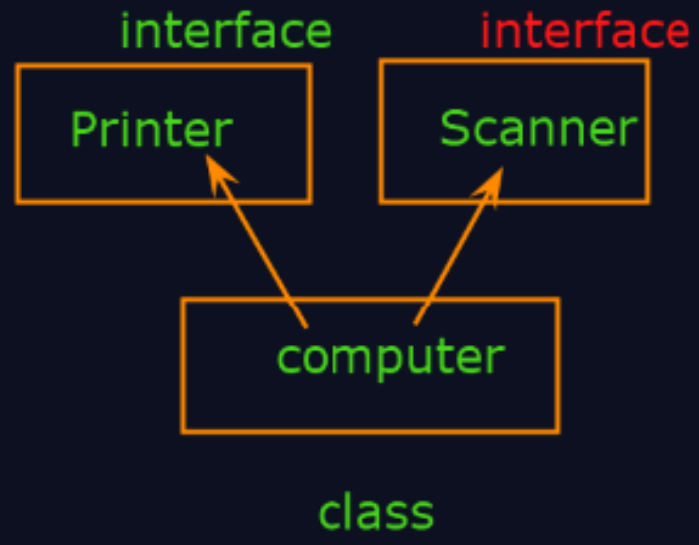
```java
    //static method
    static void display(){
        System.out.println("Static method");
    }

    //private method
    private void showbuddy(){
        System.out.println("Private method");
    }
}
```

Multiple Inheritane by using interfaces:

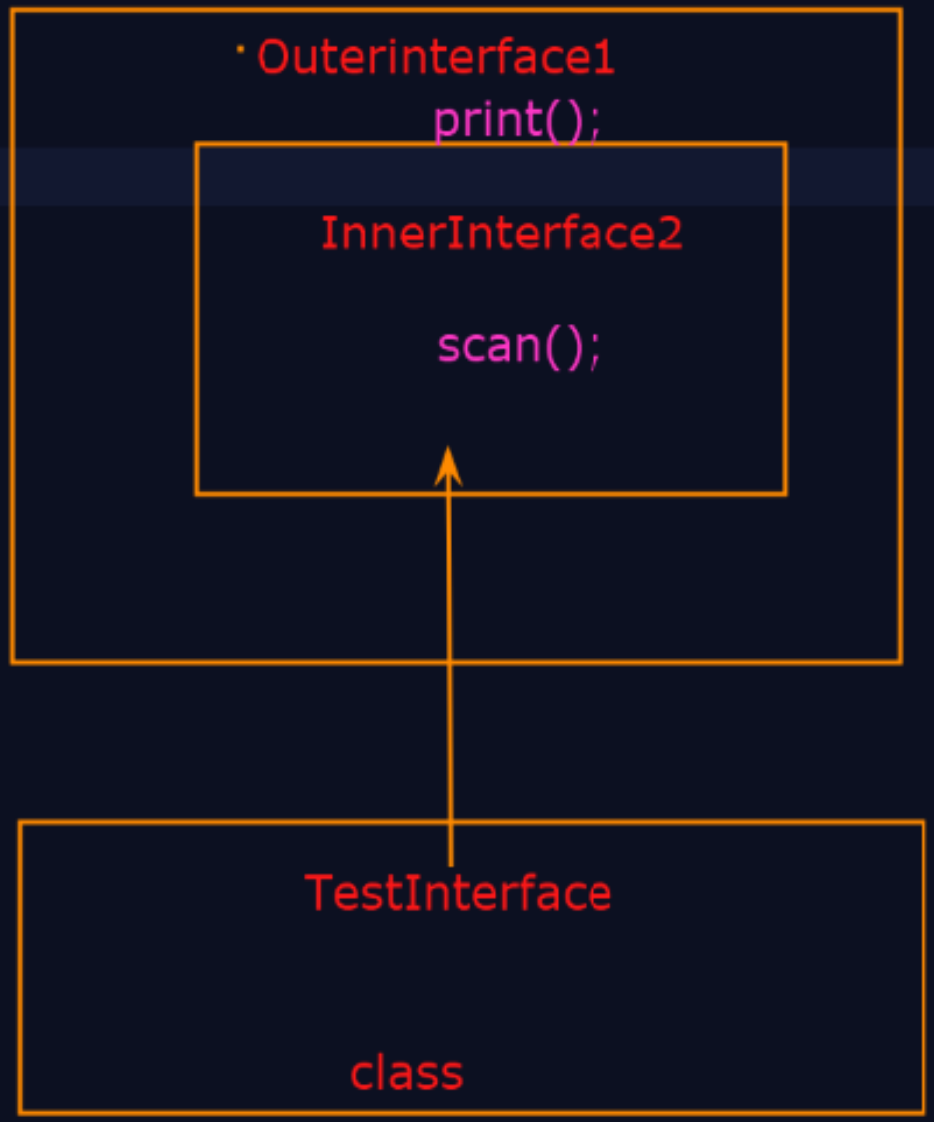




```java
class Computer implements Printer, Scanner{
    public void print(){
        System.out.println("Printer method");
    }

    public void scan(){
        System.out.println("Scanner method");
    }
}

class MultipleInheritanceDemo1{
    public static void main(String args[]){
        .
        Computer c = new Computer();
        c.print();
        c.scan();
    }
}
```



```java
class TestInterface implements OuterInterface1.InnerInterface2{
    public void scan(){
        System.out.println("Outer interface !");
    }

}

class MultipleInheritanceDemo4{

    public static void main(String args[]){

        TestInterface t1 = new TestInterface();
        t1.scan();
    }
}
```



```java
//Nesting of interfaces

interface OuterInterface1{
    void print();

    interface InnerInterface2{
        void scan();

    }
}
//NEsting interfaces are allowed to access methods of nested interfaces
class TestInterface implements OuterInterface1.InnerInterface2{
    public void scan(){
        System.out.println("Outer interface !");
    }

}

class MultipleInheritanceDemo4{

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