

abstract class

abstract method
abstract — enter
out ab 17

end }
m. ✓

} —

Java multiple inheritance

(1) directly — X

(2) indirect ✓

↓
Interface

abstract class {

abs method 1;

abs method 2;

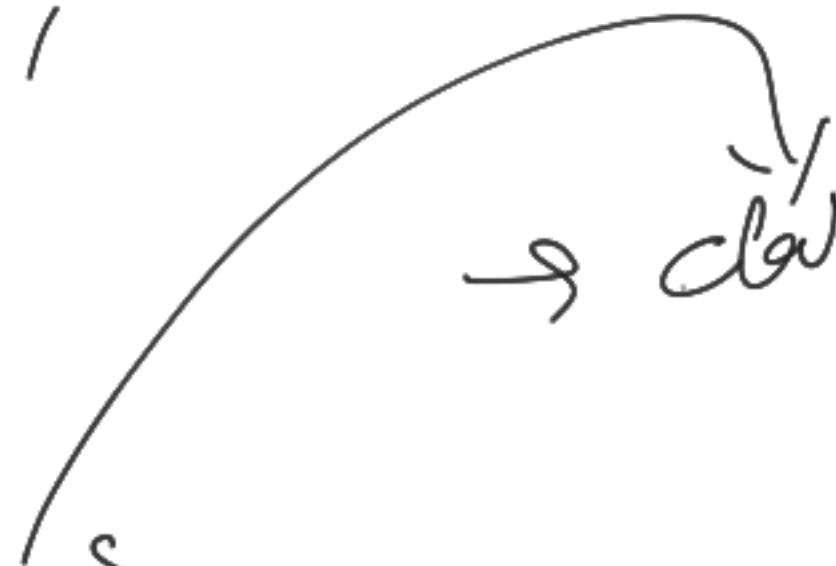
non-abs

}

abstract class 2 {

non-abs

}



→ class Abc {

AI

↳ Speed -

↳ Risk ↑

↳ Under →

↳ Calam

↳

Ab -

=



At

}

-

}



Interface

```
interface _____  
    abs + met ( );  
    abs + meth ( );  
}
```

```
interface Abc {  
    int a = 5; → final  
    int sum(); → abs  
    void display(); → abs  
    int sub(int a);  
}
```

~~id~~ // abstract class Abc {

[abstract int sum();
abstract void display();] ✓
✓

}

//

class Def extends Abc {

/// @ -

//

interface Abc {

int sum();

void display();

override ✓
=

}

class Def implements Abc {

int sum();

}

}

void

display();

}

extend - inherit
implements - inteface

interface Abs

method → no body / no def

int abc();

int a = 5; → constant / final → can't be changed

} → multiple inheritance can be applied =

new featur → After Java 8

```
interface Abe {
```

```
    int sum();
```

default

```
    void display() {
```

```
        sout("welcome to Our program");
```

```
    }
```

```
}
```



```
interface Demo {
```

```
    int var = 5;
```

```
    // abs methd
```

```
    void display();
```

```
    default void disp2() {
```

```
        // method with default fun
```

```
        // if you will not override
```

```
    }  
    static void disp3() {
```

```
        // - cant be override  
        // - Java
```

```
    }
```

final / ~~val~~ const Value

Demo

Static

}

}

abstract class — x

vs

interface → multiple inheritance

Class vs interface

Class Abc {

int i;
int j;

void display() {
=

}
Public, Pr. Priv. & Abstr methods
}

interface Def {

int i = 5;
int j = 6;

void display();

Public - ✓ Access Modifier

```
interface Newshpae{  
    void draw();  
    int radius=10;  
  
}
```

```
class NewCircle implements NewShape{  
    public void draw(){  
        radius =1;  
        System.out.println("Radius : "+radius);  
    }  
}
```

```
class InterfaceVar{  
    public static void main(String arg[])  
    {  
        NewCircle nc = new NewCircle();  
        ns.draw();  
    }  
}
```

```
interface Newshpae{  
    void draw();  
}
```

```
interface Circle extends NewShape{  
    void getRadius();  
    int radius = 10;  
}
```

```
class NewCircle implements Circle{  
    public void getRadius(){  
        System.out.println("Radius : "+radius);  
    }  
}
```

```
class InterfaceVar{  
    public static void main(String arg[])  
    {  
        NewCircle nc = new NewCircle();  
        ns.draw(); nc.getRadius();  
    }  
}
```