Abstrat class

(1) Java me abstract class ko ham abstract keyword se declare kate hai.

(2) Abstract class ka object nhi bna skte hai.

```
abstract class A
{
  void show()
{
  S.o.p ("class A"); }
}
class Demo
{
 psvm (String... ar)
{
 A a = new A();
 a.show(); }
}
```

Error: A is abstract cannot be instantiated

(*) Abstract class ke data ko access kane ke liye use inherit kana compulsory hai.

subclass ke object se abstract class ka data access haga.

```
abstract class A
{
void show()
{
  S.o.p ("class A"); }
}
class B extends A
{
}
class Demo
{
 psvm (String... ar)
{
 B b = new B();
  b.show (); }
}
```

Class A

-: Abstract Method :-

Beena body vali method ko abstract method kehte hai. Abstract method ko abstract keyword se declare krna compulsory hota hai.

```
class A
{
   void show ();
}
class Demo
{
  psvm (string... ar)
{
}
```

Error: Missing method body or declare abstract.

* Agr kisi class me ek bhi abstract method hui toh us class ko abstract declare krna compulsory hota hai.

```
class A
{
 abstract void show();
}
class Demo
{
 psvm (String... ar)
{ }
}
```

Error: A is not abstract and does not override abstract method in A

```
abstract class A
{
abstract void show();
}
 class Demo
{
 psvm (String... ar)
{
  A a = new A();
}
}
```

Error: A is abstract can not be instantiated.

```
abstract class A
{
  abstract void show();
}
class B extends A
{
}
class Demo
{
  psvm (String... ar)
{ }
}
```

Error: B is not abstract and does not override abstract method in A

Super class me jitni bhi abstract method hai un sabhi ko sub-class me overRide krna compulsory hai.

Agr ham aisa nhi krte hai to subclass ko bhi abstract declare krna hoga..

9, Nov, 2022

```
abstract class A
{
abstract void show();
}
class B extends A
{
  void show()
{
  S.o.p("ram"); }
}
Class Demo
{
  psvm (String...ar)
{
   B b = new B();
    b.show(); }
}
```

ram

```
abstract class A
{
  abstract void show();
  abstract void show2();
}
class B extends A
{
  void show(){
S.o.p("ram"); }
}
class Demo
{
psvm (String ar[])
{  B  b= new B();
 b.show(); } }
```

Error: B is not abstract and does not override abstract method show2 in A

```
abstract class A
{
  abstract void show ();
void show 2 ()
{
  S.o.p (" class A show 2 "); }
}
class B extends A
{
   void show ()
{
  S.o.p (" Bhanu "); }
}
class Demo
{
 psvm (string ... ar)
{
   B ambu = new BU;
 ambu. show();
 ambu. show 2();
}
}
```

ram
class A show 2

we can make both abstract and non- abstract method in a abstract class.

This non - abstract method in abstract class is known as Concrete method

```
abstract class A
{
 static void show ()
{
 S.o.p (" Static Method "); }
}
class Demo
{
psvm (String ... ar)
{
   A.show () }
}
```

Static Method

We can create an static method in abstract class

```
abstract class Demo
{
 public static void main (String ar[])
{
 System.out.print (" TaNu");
}
}
```

TaNu

* We can also declare a class as Abstract who has main method.

Interface

(1) We can create a interface by using interface keyword.

(2) Only abstract method can be created in a interface

(3) We can't create a object of an interface.

```
interface Inter1
{
    public abstract void show();
}
class Demo
{
psvm (String... ar)
{
 Inter1 i = new Inter();
}
}
```

Error :- Inter1 is abstract can not be instantiated.

(1) If we want to access the interface data then we have to implement it

(2) All the method of interface should be overridden by the compul subclass is compulsory.

(3) If we didn't override all the methods of interface then we have to declare subclass as abstract.

```
interface Inter1
{
  public abstract void show();
}
class A implements Inter1
{
   void show()
{
    S.o.p ("Ram"); }
}
class Demo
{
psvm (String... ar)
{
 A a = new A();
 a.show();
}
}
```

Error: attempting to assign weaker access privileges; was public

Reason :- Interface method = public
overridden method = Default

```
interface Inter1
{
  public abstract void show();
}
class A implements Inter1
{
  public void show ()
{
  S.o.p ("Golu "); }
}
class Demo
{
  psvm (String ... ar)
{
  A a = new A();
  a.show () }
}
```

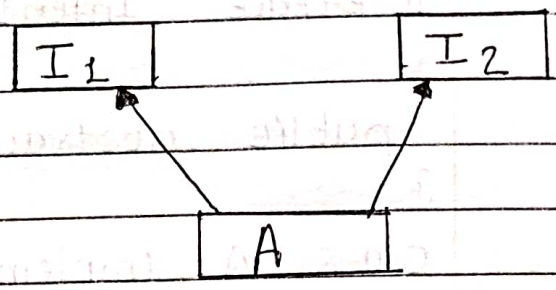
Golu

Java Compiler by default adds public & abstract keyword in front of method

```
Interface Inter1
{
  // public abstract void show();
  // abstract void show()
  // void show ()
  // public void show();
}
```

All cases will run properly.

```
interface Inter1
{
 void show1();
}
 interface Inter2
{
  void show2();
}
class B implements Inter1, Inter2
{
 public void show1()
{
   S.o.p ("show1");
}
public void show2()
{
  S.o.p ("show2"); }
}
class Demo
{
A a = new A();
a.show1();
a.show2();
}
}
```



Show1
Show2

multiple inheritence example.

By using interface we can create multiple inheritence in java.

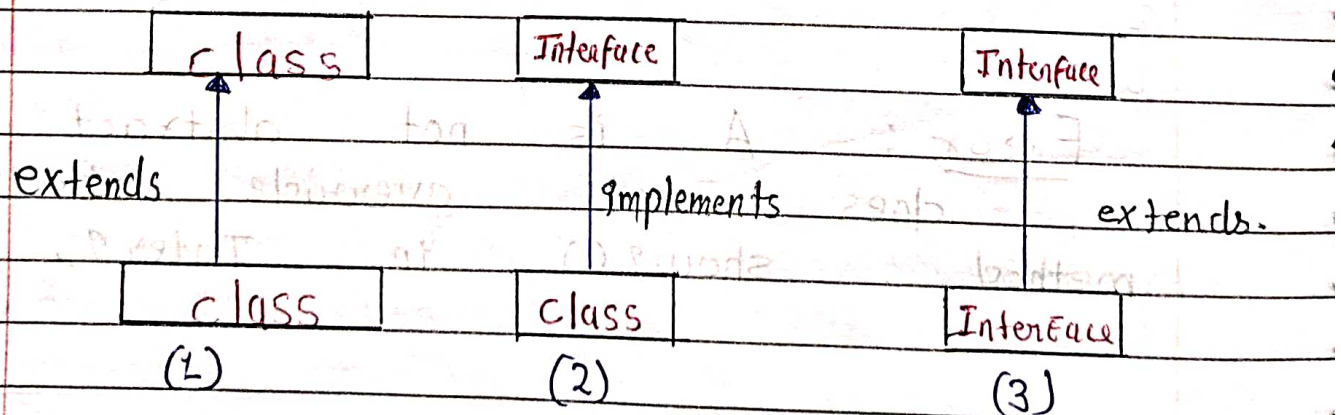
```
Interface Inter1
{
   void show1();
}
 Interface Inter2
{
   void show2();
}
class A implements Inter1, Inter2
{
     public void show1()
{
  S.o.p("show1"); }
}

Class Demo
{
psvm (String... ar)
{
   A a = new A();
  a.show1();
}
}
```

Error : A is not abstract and does not override abstract method show2() in Inter2.

```
interface Inter1
{
  void show();
}
interface Inter2
{
  void show();
}
class A implements Inter1, Inter2
{
  public void show()
{
    S.o.p ("show method"); }
}
class Demo
{
  psvm (String ar[])
{
   A a = new A();
  a.show(); }
}
```

Show method



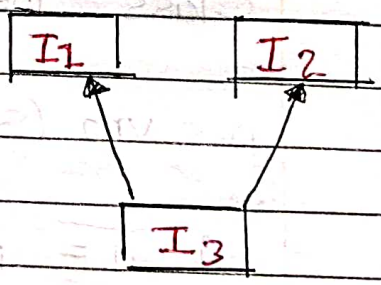
```
interface Inter1
{
   void show1();
}
interface Inter2 extends Inter1
{
    void show2();
}
 class A implements Inter2
{
   public void show1()
{
  S.O.p ("show1");
}
 public void show2()
{
 S.o.p ("show2");
}
}
Class Demo
{
p.s.vm (String ar[])
{
  A a = new A();

a.show1();
 a.show2();
}
}
```

Show1
Show2

```
interface Inter1
{
  void show1();
}
interface void show2(); Inter2
{
  void show2();
}
interface Inter3 extends Inter1, Inter2
{
  void show3();
}
class A implements Inter3
{
public void show1()
{
   s.o.p ("show1");
}
 public void show2()
{   s.o.p ("show 2");
}
public void show3()
{   s.o.p ("show 3"); }
}
class Demo
{
 psvm (String ar[])
{
 A a = new A();
 a.show1();
 a.show2();
 a.show3();
}
```

show1
show2
show3



(multiple inheritance)

```
Class A
{
  public void show()
{
   S.o.p ("class A"); }
}
class B extends A
{
  void show ()
{
   S.o.p ("Class B"); }
}
class Demo
{
  psvm (String ... ar)
{
   B b= new B();
  b.show ();
```

Error: attempting to assign weaker access privileges; was public

```
class A
{
 private void show ()
{  S.o.p ("Class A");
} }
class B extends A
{
  void show ()
{    S.o.p ("Class B"); }
}
 B b= new B();
  b.show ();
```

Class B

Private data inherit nhi hota that's why wo overRide nhi hota hai.

```
class A
{
 protected void show()
{
  S.o.p ("class A");
}
}
class B extends A
{
  void show()
{
  S.o.p ("Class B"); }
}
class Demo
{
 psvm (String... ar)
{
  B b= new B(); }
}
```

Error: attempting to assign weaker accers privileger is protected.

Private => Default => Protected => Public

whenever we use its reverse form an error will occur.

10, Nov, 2022

```
interface Inter1
{
  int x = 10;
}
class A implements Inter1
{
  void show()
  {
    S.o.p ("x =" + x); }
}
class Demo
{
psvm (String ar[])
{
  A a = new A();
    a.show (); }
}
```

x = 10

```
interface Inter1
{
  int x = 10;
}
class A implements Inter1
{
  void show()
  {
    x = 20;
    S.o.p ("x =" +x); }
}
class Demo
{ psvm (String... ar)
{
    A a = new A();
      a.show(); }
```

Error :- Can not assign to final variable x

Compiler by default adds public, static and final keywords to interface variables.

```
interface Inter1
{
   int x = 10;
}
class A implements Inter1
{
  int x = 20;
  void show ()
{
  int x = 30;
S.o.p ("x = "+x);
S.o.p (" x = " + this.x);
S.o.p (" x = " + Inter1.x);
}
```

```
interface Inter1
{
   int x = 10;
}
interface Inter2
{
   int x = 20;
}
class A implements Inter1, Inter2
{
   void show ()
{
```

```
S.o.p ("x = " +x); }
}
Class Demo
{
psvm (String... ar)
{
A a = new A(); }
}
```

Error : Reference to x is ambigious.

```
Class A
{
A()
{
S.o.p ("Class A Cons");
}
}
class Demo
{
psvm (String... ar)
{
new A(); } // object
}
```

Class A Constru

data type. var_name

int x = 10; int

A a = new A();

type. reference variable. object