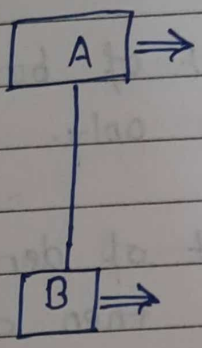
Inheritance

- connecting two or more classes is c/a Inheritance
- Creating new class from old existing class
- Inherihance use to reuse of datamember & member function. base class to
- In Inheritance firstly created class is called base class : Base / parent / super.
- New created class is called = Derived / child / subclass



Base / parent / super

Derived / child / subclass

- "extends" keyword is used to connect two classes.
- Inheritance works top to bottom.
  i.e. In Inheritance firstly allocate memory to the base class then allocate memory to derived class.
- It is important concept of OOP.

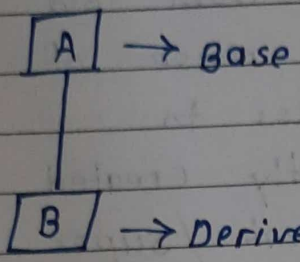
* Types of Inheritance :-

① Single
② Multilevel
③ Hierarchical
④ Multiple
⑤ Hybrid

PAGE No.
DATE / /

① Single Inheritance :-

- It has only one base class and only one derived class is called single Inheritance.



- In the Single Inheritance.

① When we create an object of base class then allocate memory to the base class only.

② When we create an object of derived class then firstly allocate memory to the base class then derived class.
i.e. (Top to Bottom)

Syntax :-

```
class Baseclass
{ =
}
class Derivedclass extends Baseclass
{ ==
}
```

e.g.

```
class A
{ =
}
class B extends A
{ ==
}
```

of baseclass as well as derived class.

⇒ base class variable are used in derived derived class also data functn & member functn.

Example :-

```
class A
{     A()
      {   s.o.p ("I am base A");
      }
}
class B extends A
{     B()
      {   s.o.p ("I am derived B");
      }
}
class MDemo
{     p.s.v.m (——)
      {
            B ob = new B();
      }
}
```

o/p

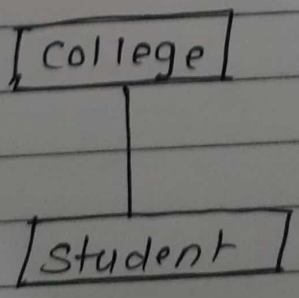
I am base A

I am derived B

Note :-

* Example of Single Inheritance

Que:



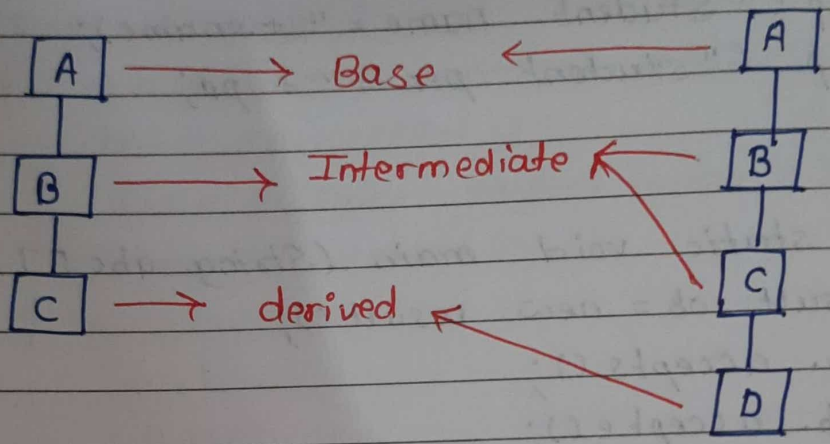
→

```
import java.util.*;
Class student
{   int rno;
    String sname;
    Scanner sc = new Scanner(System.in);
    Void accepts()
    {  s.o.p ("enter student rno:");
       rno = sc.nextInt();
       s.o.p ("enter student name:");
       Sname = sc.next();
    }
}

class exam extends student
{  int m1, m2, m3, m4, m5, m6;
   Void accepte()
   {  s.o.p ("enter six sub marks:");
      m1 = sc.nextInt();
      m2 = sc.nextInt();
      m3 = sc.nextInt();
      m4 = sc.nextInt();
      m5 =
      m6 =
   }
}

class result extends exam
{   int t;
    float p;
```

```
Void cal ()
{   t = m1 + m2 + m3 + m4 + m5 + m6;
    p = (float) t/6;
}
void disp ()
{  S.o.p (" student roll no = " + rno);
   S.o.p (" Student name = " + sname);
   S.o.p (" student per = " + pd;
}

public static void main (String abc [])
{   result ob = new result ();
    ob. accepts ();
    ob. accepte ();
    ob. cal ();
    ob. disp ();
}

}
```

② Multilevel Inheritance :-

- Only one main class, only one derived class and one ore more Intermediate classes is called Multi-level Inheritance.
- It is also called chain of classes.



Syntax :-

```
Class A
{ =
}
Class B extends A
{ =
}
Class C extends B
{ =
}
```

e.g:

```
Class A
{ =
}
Class B extends A
{ =
}
Class C extends B
{ =
}
Class D extends C
{ =
}
```

- When we create an object of derived class then firstly calling Base class then intermediate class & then the derived class (i.e. Top to bottom)
- If we create base class object then only one calling base class method not intermediate & derived.
- If we create intermediate class object then calling only one base class & intermediate class only not derived class.

Example

```
class A
{   int x,y,z;
    A()
    {  x=10;
       S.o.p ("I am base class");
    }
}
Class B extends A
{   B()
    {  y=20;
       S.o.p ("I am intermediate class");
    }
}
class C extends B
{   C()
    {  z=30;
       S.o.p ("I am derived class");
       S.o.p ("Addition =" + (x+y+z));
    }
}
class Demo
{   p.s.v.m (—)
    {   C ob = new C();
    }
```

O/p

I am base class
I am intermediate class
I am derived class

PAGE No.
DATE / /

Slip5 Q.1

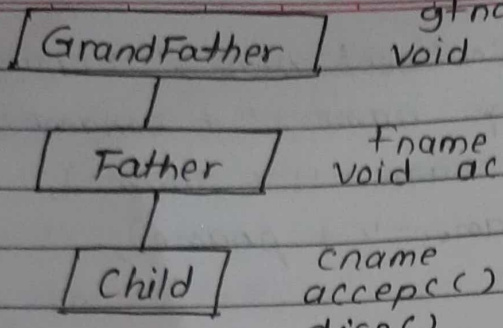
Que: W/p multilevel inheritance such that country is inherited from continent. state is inherited by country. Display the place, state, country and a continent.

→

```
import java.util.*;
class continent
{   String cname;
    Scanner sc = new Scanner(System.in);
    Void acceptc()
    {  s.o.p ("enter continent name:");
       cname = sc.next();
    }
}
class country extends continent
{   String coname;
    Void acceptco()
    {  s.o.p ("enter country name:");
       coname = sc.next();
    }
}

Class state extends country
{  String sname, pname;
   Void acceptS()
   {  s.o.p ("enter state name");
      sname = sc.next();
      S.o.p ("enter place name:");
      pname = sc.next();
   }

   Void disp()
   {
```

```
s.o.p ("The Continent name =" +cname);
s.o.p (" Country name =" + coname);
s.o.p (" State name =" + sname);
s.o.p (" place name =" + pname);
}
}

class Demo
{   p.s.v.m (——)
    {   state ob = new state ();
        ob. accept c ();
        ob. accept co ();
        ob. accept s ();
        ob. disp ();
    }
}
```

Que:

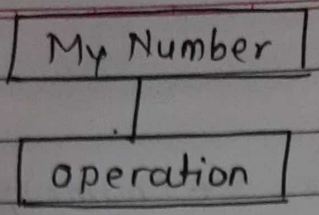


→

```
import java.util.*;
class gfather
{   String gname;
    Scanner sc = new Scanner (System.in);
    Void acceptg()
    {   s.o.p ("enter grand father name:");
        gname = sc.next();
    }
}

class father extends gfather
{   String fname;
    Void acceptf()
    {   s.o.p ("enter father name:");
        fname = sc.next();
    }
}

class child extends father
{   String cname;
    Void acceptc()
    {   s.o.p ("enter child name:");
        cname = sc.next();
    }
```

```
void disp()
{   s.o.p(" child name = " + cname);
    s.o.p(" father name = " + fname);
    s.o.p(" groundfather name = " + gname);
}

p.s.v.m (——)
{   child ob = new child();
    ob.acceptg();
    ob.acceptf();
    ob.acceptc();
    ob.disp();
}

}
```

Que:



→

```
import java.util.*;
class mynumber
{   int n;
    Scanner sc = new Scanner (System.in);
    void accept()
    {  s.o.p ("enter num:");
       n = sc.nextInt();
    }
}

class operation extends mynumber
{   void prime
    {  int i;
       for(i=2; i<n; i++)
       {  if (n%i==0)
             break;
       }
      if (i==n)
       s.o.p ("no is prime");
      else
       s.o.p (" no. not prime");
    }

    Void fact()
    {   int f=1;
        for (int i=1; i<=n; i++)
        {  f=f*i;
        }
```

```
s.o.p ("factorial of nos = " +f);
}

Void perfect()
{   int i, s=0;
    for (i=1; i<n; i++)
    { if (n%.i==0)
        s=s+i;
    }
    if (s==n)
      s.o.p (" num is perfect");
    else
      s.o.p (" num not perfect");
}
  p.s.v.m (String abc[])
{ operation ob = new operation();
    ob.accept();
    ob.prime();
    ob.fact();
    ob.perfect();
}
}
```

PAGE No.
DATE / /

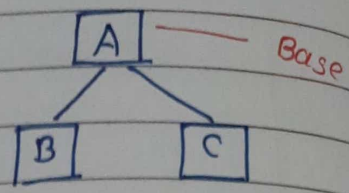
Que: accept two array and display union of it.

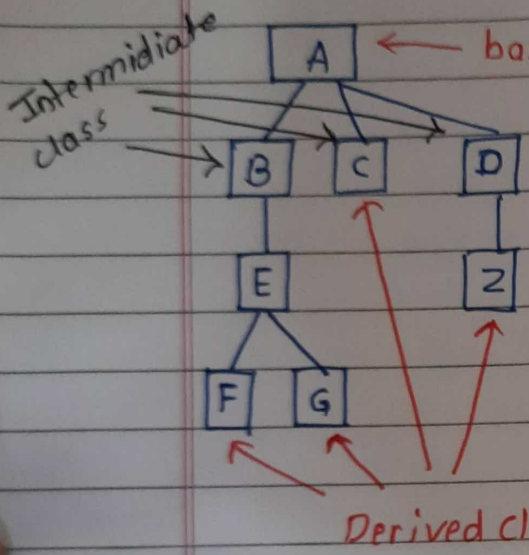
```
import java.util.*;
class demo
{  public static void main (String abc[])
   {  int i;
      int a = new int [10];
      int b = new int [10];
      Scanner sc = new Scanner (System.in);
      s.o.p ("limit of first array :");
      int n1 = sc.nextInt();
      s.o.p ("limit for second array:");
      int n2 = sc.nextInt();
      s.o.p ("enter 1st array elements:");
      for (i=0; i<=n1; i++)
      {  a[i] = sc.nextInt();
      }
      s.o.p ("enter 2nd array elements:");
      for (i=0; i<=n2; i++)
      {  b[i] = sc.nextInt();
      }
      s.o.p ("Union of two array =");
      for (i=0; i<=n1; i++)
      { s.o.p (a[i] + " ");
      }
      for (i=0; i<=n2; i++)
      { s.o.p (b[i] +" ");
      }
   }
}
```

PAGE No. | Monday.
DATE | 3/6/24

③ Hierarchical Inheritance

- Only one base class and two or more derived classes is called Hierarachical Inheritance.





- Only one base class two or more intermediate OR Derived classes is called Hierarachical Inheritance.

- It is like Tree structure Inheritance.

Note :-

In it create object of all derived classes.

Ex.

```
class A
{ =
}
class B extends A
{ =
}
class C extends A
{ =
}
```

DATE / /

Ex.

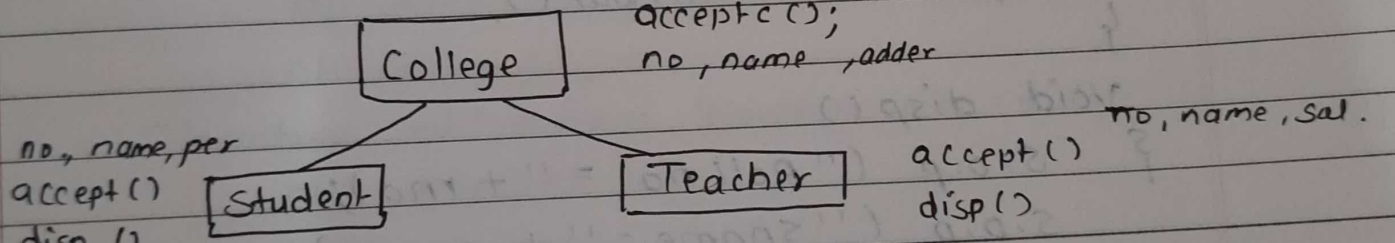
```
class A
{ =
}
class B extends A
{ =
}
class C extends A
{ =
}
class D extends A
{ =
}
class E extends B
{ =
}
class z extends D
{ =
}
```

```
class F extends E
{ =
}
class G extends E
{ =
}
```

When we create object of class G then allocate memory to the.

A → B → E → G.

Que:



→

```
Import java.util.*;
class college
{ int cno;
  String cname;
  Scanner sc = new Scanner (System.in);
  void acceptc()
{ System.out.println ("enter college no:");
    cno = sc.nextInt();
    S.o.p ("enter college name:");
    cname = sc.next();
}
```

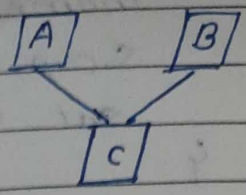
```
Void dispc()
{ s.o.p ("college no = " + cno);
  s.o.p ("college name = " + cname);
}
}

Class student extends college
{ int rno;
  String sname;
  float per;
 void accept ()
{ s.o.p ("enter roll no:::");
  rno = sc.nextInt();
  s.o.p ("enter name:::");
  sname = sc.next();
  s.o.p (" enter per :::");
  per = sc.nextFloat();
}
void disp()
{ s.o.p ("Roll no = " + rno);
  s.o.p ("sname = " + sname);
  s.o.p ("per = " + Per);
}
}

Class teacher extends college
{ int tno;
  String tname;
  float sal;
 Void accept()
{ s.o.p ("enter teacher no:::");
  tno = sc.nextInt();
```

```
S.o.p ("enter teacher name");
tname = sc.next();
S.o.p ("enter sal");
sal = sc.nextFloat();
}
Void disp()
{ S.o.p ("Teacher no = "+ tno);
  S.o.p ("Teacher name = "+tname);
  S.o.p ("Teacher sal = " + sal);
}
}
class MD
{ p.s.v.m (String arg[])
  { student ob = new student();
    ob. accept();
    ob. accept ();
    ob. disp ();
    ob. disp ();
  Teacher obl = new Teacher ();
    obl. accept ();
    obl. accept ();
    obl. disp ();
    obl. disp ();
  }
}
```

PAGE No.
DATE / /

④ Multiple Inheritance

- Two or more base classes & only one derived class is c/a Multiple Inheritance.



Syntax :-

```
Class A
{ =
}
Class B
{ =
}
Class C extends A, B
{ =
}
```

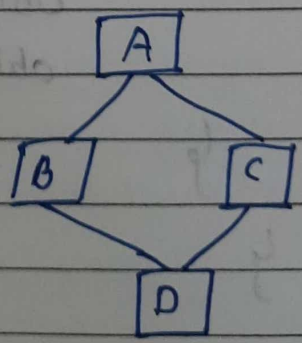
✗ Not allowed.

- In java only one class extends at a time.
- java does not support Multiple Inheritance directly

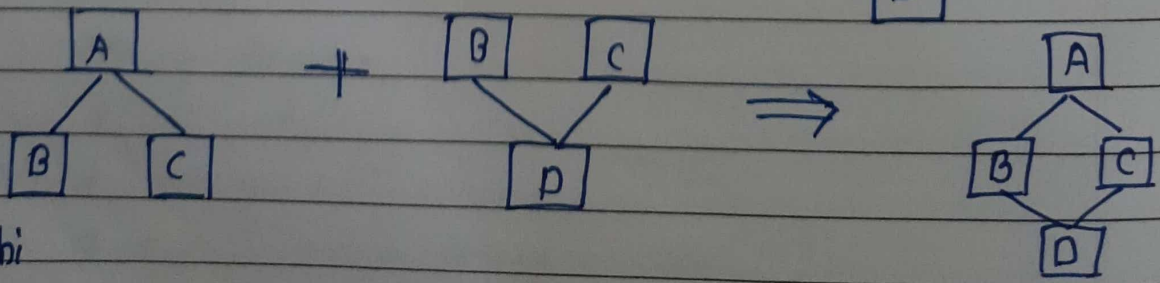
Note :- we can overcome multiple Inheritance using Interface

⑤ Hybrid Inheritance

- combination of two or more inheritance including multiple Inheritance is c/a Hybrid Inheritance.



⇒



Combi

PAGE No.
DATE / /

e.g. class A
{ =
}
class B extends A
{ =
}
class C extends A
{ =
}
class D extends B, C ✗ Not allowed.
{ =
}