It is a purely object oriented language which is basically robust language and also platform independent (7575 973 or 725) language.

OOPS features :-

① Class

② Object

③ Polymorphism
   a) Function overloading
   b) Function overriding

④ Encapsulation

⑤ Abstraction

⑥ Data Hiding

⑦ Inheritence
   a) Single level Inheritence
   b) Multi level Inheritence
   c) Multiple Inheritence
   d) Hierarchical Inheritence
   e) Hybrid Inheritence

⑧ Package
   a) Same package same class
   b) Same package sub class
   c) Same package non-sub class
   d) Different package sub class
   e) Different package non-sub class

⑨ Exception
   a) Try
   b) Catch
   c) Through
   d) Throughs
   e) Finally

⑩ Interface

⑪ Access Specifier
   a) Public
   b) Private
   c) Protected
   d) Default

(12) Applet
(13) Awt (Abstruct Window Toolkit)
(14) Constructor
(15) Destructor

CLASS

It is a user defined datatype which consist of variable and function Variable within a class is known as Member variable and function within a class is known as Member function of a class.

Member variable and member function together forms data member of a class

*** Why JAVA is called purely object oriented language?

→

Because,

① Multiple inheritence is a drawback of OOPS which is solved in JAVA programming language ~~other~~ by using interface concept that means through Diamond inheritence concept JAVA can implement the multiple inheritence concept

② Every JAVA program must start with a class

System

◉ Rules of Naming ~~User~~ defined Class :-

First letter of each word must be capital as JAVA is a case sensitive programming language. Example -

(i) Scanner
(ii) Buffered_Reader

◉ Rules of Naming System defined function :-

Except first word all the other word's first letter must be capital. Ex- nextInt()
readLine()

- Arpan
  - Himangshu
    - Depra.Java
    - Subham.Java
    - Maharsi
      - Sammar
        - Ankan.Java
        - Arutra
  - Anirban
    - Srinjoy
      - moumita.Java
      - Debaleena.Java

import Arpan.Anirban.*;

Arpan package এর ভিতর Anirban package টা খুলবে

" " . " . Srinjoy *;

Arpan package এর ভিতর Anirban→Srinjoy এর ভিতর দুটো file-ই দেখা যাবে

" " . " . " . moumita ←

(file কে call করলে extension লাগবে না)

শুধু moumita file টাই দেখা যাবে.

```
Class A        ← Class: keyword;  A: name of the class
{
public static void main (String h[]) args[ ]
{
System.out.print ( )
System.out.println ( )   → S.D.F / S.D.m
}
```

(String h[]) → Argument/Parameter pasing; String → class

- ⊙ <u>public</u> → is a access specifier. main ( ) must be public because main ( ) can be accessible anywhere within the program/class.

- ⊙ <u>static</u> → It is a keyword through which if we declare main as static then we don't have to create an object of the class to call the function.

⊙ String h[ ] :-

As Java is high level language it will take string as input by default so, to store the string input JAVA will take string as argument

⊙ System.out.print ( ) ;-

⊙ System :- It is a system defined class.

⊙ Out :- a property or object to ~~take~~ print anything on the screen for the user

⊙ print :- It is a system defined function which is used to print anything on the screen and put the cursor on the same line

• Addition of two inputs :-

⊙ Program of how to take input :-

```
import java.util.*;
  class A
  {
   public static void main ( String h[ ])
   {
    int a, b, c;
   Scanner ob = new Scanner (System.in);
```

(System defined classname) ← Scanner

(name of the object) ← ob

new → ram memory ਨਾਲ object ਦਾ ਰਾਹ ਦੱਸੇ ਤਾਂ ਜੋ help ਨਾਲ constructor function ਨੂੰ

Scanner (System.in); ← constructor function

```
   System.out.print (" Enter the 1st no = ");
     a = ob.nextInt ( );
   System.out.print (" Enter the 2nd no = ");
     b = ob.nextInt ( );
      c = a + b ;
   System.out.print (" Addition = " + c );
    }
  }
```

+ → c language ਵਿੱਚ (,), C++ ਵਿੱਚ << ਹੁੰਦਾ ਹੈ ਜਿਸਦਾ ਮਤਲਬ concatenate operator

or,

```
System.out.print (" Addition of " + a + " & " + b + " is = " + c );
```

Addition of 10 & 20 is = 30

- Scanner :-

It is a system defined class which is present within JAVA util package It is used to convert any string value to corresponding integer or float or double or string It consist of four inbuild function -

1) nextInt ()
2) nextFloat ()
3) nextDouble ()
4) nextLine ()

- ob :-

It is an user defined object of scanner class which is used to allocate memory at runtime of the class

- new :-

new is a Keyword which is used to allocate the memory and help the constructor to create an object within the RAM memory.

- Scanner (system.in) :-

It is constructor which will take input as an argument so that Keyboard is activated to take input

Polymorphism

- Poly = Many and Morphism = forms function having same name with many forms i.e either different argument or parameter passing or either number of parameter passing is different is known as Polymorphism.

Note Java supports two types of Polymorphism.

1. Function Overloading
2. Function Overriding

- **Function Overloading :-**

In case of function overloading function name and return type must be same and parameter or argument passing must be different i.e either by types of parameter or by no. of parameter. Function Overloading can be applicable within the same class. i.e no inheritence is required. function which has more parameter can overload or hide the function which has less parameter. In other terms we can say that function overloading is depends on calling of the function i.e within the main function accessibility is depends on function call through parameter passing.

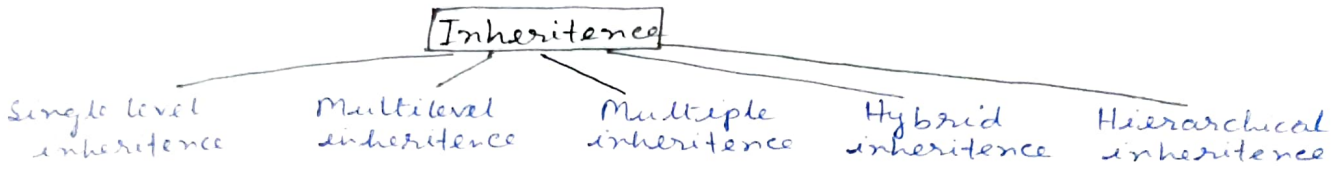
- **Function Overriding :-**

It is applicable incase of inheritance i.e two class is minimum required and within both the class function name, parameter passing, return type must be same. Always child class will priorities the function of itself i.e child class function's output will be displayed. i.e child class will hide the data or parent class through Overriding.

Inheritence

It is one of the most important oops concept in JAVA In case of inheritence minimum two classes are required One is parent class which is known as Base class or Super class And another is child class which is also known as Subclass or derived class

Always child class will hold the property of parent class That means Parent class's variable and function will be present within the child class through inheritence "extends" is the keyword which is used to inherits parent class data through child class



Why Java is known as Purely Object Oriented language ?

Ans: Because JAVA doesn't support Multiple inheritence To implement the concept of multiple inheritence JAVA introduced interface concept

Access Specifier in JAVA

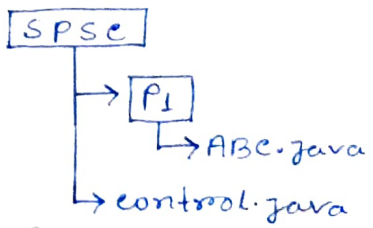
1. Private
2. Default
3. Protected
4. Public

- Folder → consists of files + subfolder
- package → consists of classes + subclasses
- SPSC → Same Package same class
- SPSUC → Same Package sub class
- SPNSUC → Same Package non subclass
- DPSUC → Different package sub class
- DPNSUC → Different Packdge non sub class

ABC.java

```
package P1;
public class ABC
{
 public int a = 10;
 protected int b = 20;
 private int c = 30;
         int d = 40;

 void show()
 {
 System.out.println("Public = "+a);
 System.out.println("Protected = "+b);
 System.out.println("Private = "+c);
 System.out.println("Default = "+d);

 }
}
```

Structure :-



Private ✓
Default ✓
Protected ✓
Public ✓

control.java

```
import P1.*;
class control
{
 public static void main(String as[])
 {
  ABC ob = new ABC();
  ob.show();
 }
}
```

cmd:-

E:\SPSC> javac control.java ↵

E:\SPSC> java control ↵

output:-

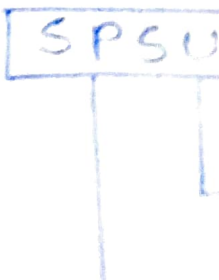
```
Public = 10
Protected = 20
Private = 30
Default = 40
```

ABC.java

```
package P1;
public class ABC
{
 public int a = 10;
 protected int b = 20;
 private int c = 30;
         int d = 40;

}
```

X42.java

Stru



Output :-

```
Public = 10
Protected = 20
Private = 30
Default = 40
```

ABC.java

```
package P1;
public class ABC
{
  public int a = 10;
  protected int b = 20;
  private int c = 30;
      int d = 40;
}
```

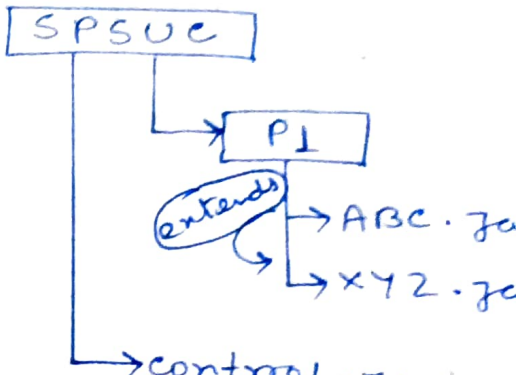
XYZ.java

```
package P1;
public class XYZ extends ABC
{
 void show()
 {
 System.out.println("Public = " + a);
 System.out.println("Protected = " + b);
// System.out.println("Private = " + c);
System.out.println("Default = " + d);
 }
}
```

control.java

```
import P1.*;
class control
{
   Public static void main(String as[])
   {
    XYZ ob = new XYZ();
    ob.show();
   }
}
```

Structure :-



CMD:-

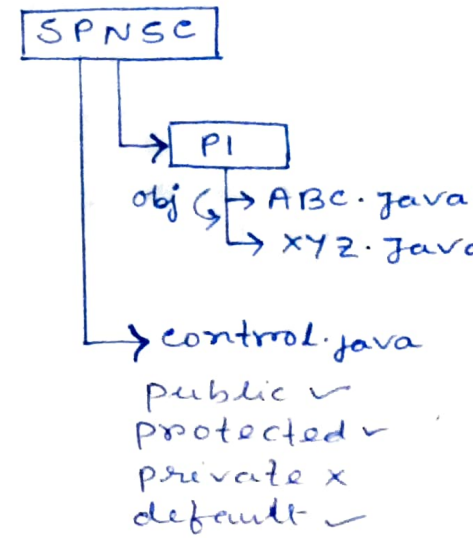
```
E:\SPSUC> javac *.control.java
E:\SPSUC> java *.control
```

output:-

```
Public = 10
Protected = 20
Default = 40
```

ABC.Java

```
package P1;
Public class ABC
{
 Public int a = 10;
 protected int b = 20;
 private int c = 30;
         int d = 40;
}
```



XYZ.Java

```
package P1;
Public class XYZ
{
void show()
{
 ABC ob = new ABC();
 System.out.println("Public = " + ob.a);
 System.out.println("Protected = " + ob.b);
 System.out.println("Default = " + ob.d);
 }
}
```

CMD:-

```
E:\SPNSC> Javac *.control.java
E:\SPNSC> Java *.control
```

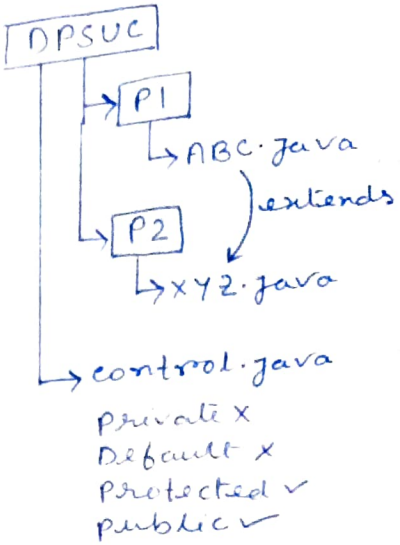
```
Public = 10
Protected = 20
Default = 40
```

ABC.java

```
package P1;
public class ABC
{
  public int a = 10;
  protected int b = 20;
  private int c = 30;
          int d = 40;
}
```

control.java

```
import P1.*;
import P2.*;
class control
{
Public static void main(String as[])
{
 XYZ ob = new XYZ();
   ob.show()
 }
}
```



XYZ.java

```
package P2;
import P1.*;
public class XYZ extends ABC
{
 void show()
 {
 System.out.println("Public = " + a);
 System.out.println("Protected = " + b);
 System.out.println("Default = " + d);
 System.out.println("Private = " + c);
 }
}
```

cmd

```
E:\DPSUC>javac control.java
E:\DPSUC>java control
```

Output

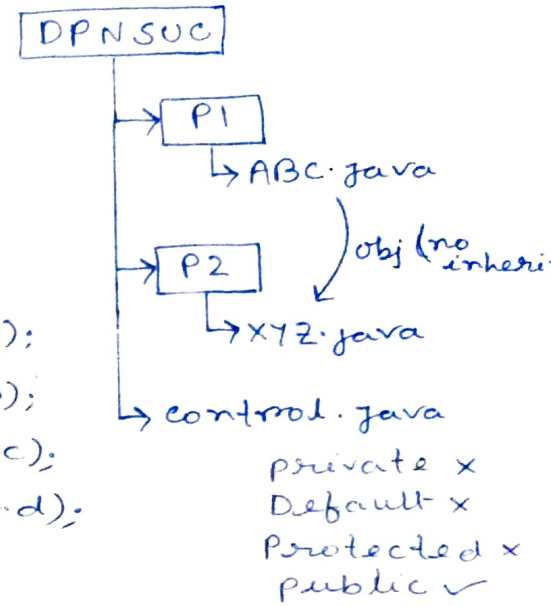
```
Public = 10
Protected = 20
```

ABC.java

```
package P1;
public class ABC
{
  public int a = 10;
  protected int b = 20;
  private int c = 30;
          int d = 40;
```

XYZ.java

```
Package P2;
import P1.*;
Public class XYZ
{
void show()
{
ABC ob = new ABC();
System.out.println("Public = "+ob.a);
// System.out.println("Protected = "+ob.b);
// System.out.println("Private = "+ob.c);
// System.out.println("Default = "+ob.d);
}
}
```



Control.java

```
import P1.*;
import P2.*;
class control
{
Public static void main (String as[])
{
XYZ ob = new XYZ();
ob.show();
}
}
```

CMD

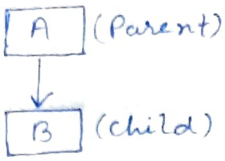
```
E:\DPNSUC> javac control.java ↵
E:\DPNSUC> java control ↵
```

Output

```
Public = 10
```

Single - level inheritence

In this inheritence maximum 2 class is required. First class is always parent class. Second class is always child class



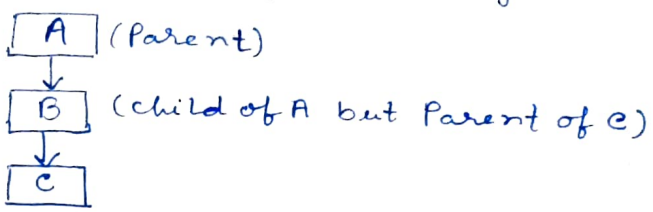
- **Armstrong no & Palindrome no :-**

```
import java.util.*;
class Arm
{
  int n,x,rem,s=0,c=0,p;
  public void input()
  {
   Scanner ob = new Scanner(System.in);
   System.out.print("Enter the no. = ");
   n = ob.nextInt();
  }
 public void check()
  {
   for(x=n;n!=0;n/=10)
     {
       c++;
     }
   for(n=x;n>0;n/=10)
    {
     rem = n%10;
     p = (int)Math.pow(rem,c);
     s = s+p;
    }
 if(x==s)
  System.out.println(x+" is Armstrong no. ");
 else
  System.out.println(x+" is NOT Armstrong no ");
  }
}

class Palin extends Arm
{
 int rev=0;
 public void check1()
 {
  s=0;
  for(n=x;n>0;n/=10)
   {
    rev = (n%10);
    s = (s*10)+rev;
   }
if(s==x)
  System.out.println(x+" is Palindrome no");
```

```
else
  System.out.println(n+" is NOT Palindrome no ");
}

public static void main(String as[])
{
  Palin ob = new Palin();
  ob.input();
  ob.check();
  ob.check1();
 }

}
```

Multi-level inheritence

In this concept minimum three class is required. First class is Parent class in that respect second class is child class when third class occurs, in this case second class is the parent class and third class is the child class. In this way it will go on.

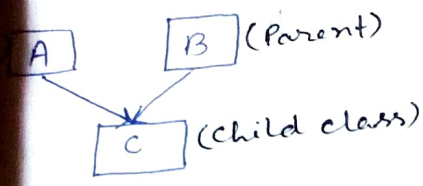


- **SPY no, Neon no & Krishnamurthy no :-**

```
import java.util.*;
class Neon
{
  int n, sq, rem, s = 0;
  public void input()
    {
     Scanner ob = new Scanner(System.in);
     System.out.print("Enter the no. = ");
     n = ob.nextInt();
    }
  public void check()
    {
     for(sq = n*n; sq>0; sq/=10)
     { rem = sq%10;
       s = s+rem;
     }
     if (s==n)
     System.out.println(n+" is Neon no.");
     else
     System.out.println(n+" is NOT Neon no.");

    }

}
```

```
class Spy extends Neon
{
 int n, p=1;
 public void check1 ()
 {
    s=0;
    for (x=n; n>0; n/=10)
    {
      rem = n%10;
      s+ = rem;
      p* = rem;
    }
   if (s==p)
   System.out.println (x+ " SPY no");
  else
   System.out.println (x+ " is NOT SPY no");
 }
}
class Krishnamurty extends Spy
{
 int j, f=1;
 public void check2 ()
  {
   s=0;
   for (n=x; n>0; n=n/10)
    {
     rem = n%10;
    for (j=1; j<=rem; j++)
     {
       f = f*j;
     }
     s=s+f;
     f=1;
    }
 if (s==n)
  System.out.println (x+ " is Krishnamurty no.");
 else
  System.out.println (x+ " is NOT Krishnamurty no");
 }
 public static void main (String as [])
 {
  Krishnamurty ob = new Krishnamurty ();
    ob.input();
    ob.check();
    ob.check1();
    ob.check2();
  }
```

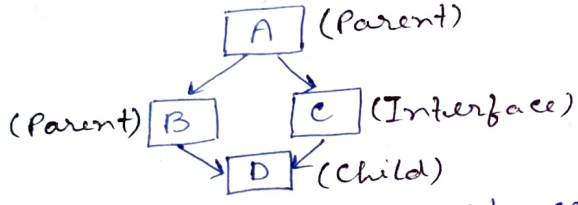
MULTIPLE INHERITENCE



When more than one class's data is inherited by a single class, this is known as Multiple inheritence.

In purely object oriented language like Java, only one class can be inherited by one class that's why JAVA doesn't support Multiple inheritence.

To implement multiple inheritence Java allows Diamond inheritence through interface concept.



Diamond inheritence

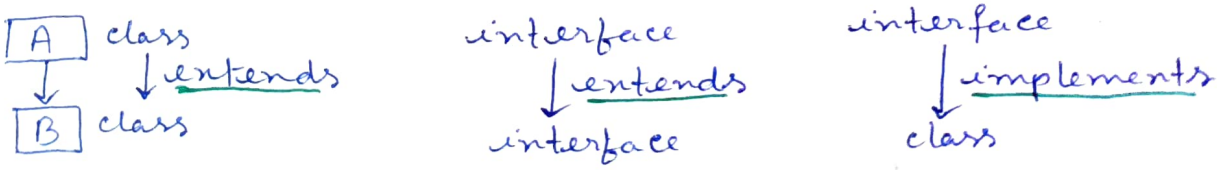
interface :-

It is like a class but within the interface all the function with the interface must be declared not be defined and all the variables within the interface must be final so that it's value can't be change.

Within the interface as function is not defined so we can't create an object of that interface, the class who inherites the interface will have the responsibility to define the function within it.

'implements' is the keyword through which we can inherite interface within a class.

'extends' is the keyword to inherite a class within another class as well as interface within another ~~inte~~ interface.



- **'final' keyword :-**

class -ज़ respect में

If we declare a class as final then that class can't be inheritent within the child class.

function -ज़ respect में

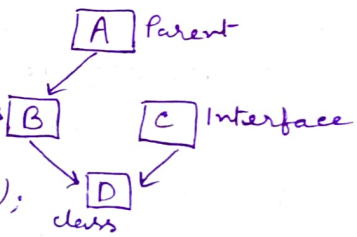
If we declare a function as final within the same class or different class then that function can't be overloaded or overrided.

variable -ज़ respect में

If we declare a variable as final then the variable's value can't be changed throughout the program.

- **Multiple inheritence or Diamond inheritence :-**

```
import java.util.*;
class A
{
public int len, br, ar, pr;
public void input()
{
Scanner ob = new Scanner(System.in);
System.out.print("Enter the length = ");
len = ob.nextInt();
System.out.print("Enter the Breadth = ");
br = ob.nextInt();
}
public void calc()
{
ar = len * br;
pr = 2 * (len + br);
System.out.println("Area of Rect. = " + ar);
System.out.println("Perimeter of Rect. = " + pr);
}
}

class B extends A
{
public int x, ar1, pr1;
public void input1()
{
Scanner ob = new Scanner(System.in);
```



```
    System.out.print (" Enter the Side = ");
    x = ob.nextInt ();
    }
  public void calc1 ()
    {
     ar1 = x * x;
     pr1 = 4 * x;
    System.out.println (" Area of Sq = " + ar1);
    System.out.println (" Perimeter of Sq = " + pr1);
    }
}

interface C
{
  public final double pi = 3.14;
  public void cir_area (double r);
  public void cir_perimeter (double r);
}

class D extends B implements C
{
  public void cir_area (double r)
    {
     System.out.println (" Area of Circle = " + (pi * r * r));
    }
  public void cir_perimeter (double r)
    {
     System.out.println (" Perimeter of Circle = " + (2 * pi * r));
    }
public static void main (String as [])
{
  D ob = new D ();
  double rad;
  Scanner ob1 = new Scanner (System.in);
  System.out.println (" Enter the Radius = ");
  rad = ob1.nextDouble ();
  ob.cir_area (rad);
  ob.cir_perimeter (rad);
  ob.input ();
  ob.calc ();
  ob.input1 ();
  ob.calc1 ();
}
}
```

- Final Keyword :-

i) respect to class :-

```
final class ABC
{
  public int x = 5;
  public void show ()
  {
   System.out.println ("x = " + x);
  }
}
class XYZ extends ABC
{
 public static int y=10;
 public void show1 ()
 {
  System.out.println ("y = " +y);
 }
public static void main (String as [])
 {
  XYZ ob = new XYZ ();
   ob.show ();
   ob.show1 ();
 }
}
```

ii) respect to variable :-

```
class x
{
 public static final int a = 5;
 public void show ()
 {
  System.out.println ("x's value = " + a);
  a++;
  System.out.println ("x's value = "+ a);
 }
public static void main (String as [])
 {
   x ob = new x ();
   ob.show ();
 }
}
```

iii)

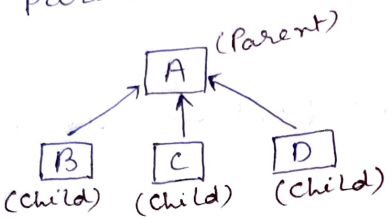
```
import java.util.*;
interface Rect
{
 public void rect_area (int len , int br);
 public void rect_perimeter (int len, int br);
}
```

```
interface cir extends Rect
{
  public final double pi = 3.14;
  public void cir_area (double r);
  public void cir_perimeter (double r);
}

class Final1 implements cir
{
 int ar, pr;
 double ar1, pr1;
 public void rect_area (int len, int br)
  {
   ar = len * br;
  }
 public void rect_perimeter (int len, int br)
  {
   pr = 2 * (len + br);
  }
public void cir_area (double r)
 {
  ar1 = pi * r * r;
 }
public void cir_perimeter (double r)
 {
  pr1 = 2 * pi * r;
 }
}
class Final2 extends Final1
{
 static int len, br;
 static double rad;
 public void show()
 {
  System.out.println ("Area of Rect = " + ar);
  System.out.println ("Perimeter of Rect = " + pr);
  System.out.println ("Area of Cir. = " + ar1);
  System.out.println ("Perimeter of Cir. = " + pr1);
 }

public static void main (String as[])
 {
  Scanner ob = new Scanner (System.in);
  System.out.print ("Enter the length = ");
  len = ob.nextInt ();
  System.out.print ("Enter the breadth = ");
  br = ob.nextInt ();
  System.out.print ("Enter the radius = ");
  rad = ob.nextDouble ();
  Final2 obj = new Final2 ();
  obj.cir_area (rad);
  obj.cir_perimeter (rad);
```

```
obj.rect_area(len, br);
obj.rect_perimeter(len, br);
obj.show();
}
}
```

Hierarchical inheritence

In this inheritence only one parent class is present and all the child class are depends on parent class.



perfect no, armstrong no, spy no :-

```
import java.util.*;
class First
{
  int n;
  public void input()
  {
  Scanner ob = new Scanner(System.in);
  System.out.print("Enter the no. = ");
  n = ob.nextInt();
  }
}

class Second extends First
{
 public int i, s=0;
 public void check()
  {
   for(i=1; i<=n/2; i++)
     {
       if(n%i==0)
         s+=i;
     }
   if(s==n)
    System.out.println(n+" is Perfect no.");
   else
    System.out.println(n+" is NOT Perfect no.");
  }

}
```

```
class Third extends First
{
  int x, rem, s=0, p, c=0;
  public void check()
  {
   for (x=n; n>0; n/=10)
    c++;
   for (n=x; n!=0; n/=10)
    {
     rem = n%10;
     p = (int) Math.pow (rem, c);
     s+=p;
    }
   if (s==x)
    System.out.println (x+" is Armstrong no.");
   else
    System.out.println (x+" is NOT Armstrong no.");
  }
}

class Fourth extends First
{
  int s=0, p=1, x, rem;
  public void check()
  {
    for (x=n; n>0; n/=10)
     {
       rem = n%10;
       s+= rem;
       p*= rem;
     }
    if (s==p)
     System.out.println (x+" Spy no.");
    else
     System.out.println (x+" NOT Spy no.");
  }
}

class E
{
 public static void main(String as[])
 {
  Second ob = new Second();
  Third ob1 = new Third();
  Fourth ob2 = new Fourth();
  ob.input();
  ob.check();
  ob1.input();
  ob1.check();
  ob2.input();
  ob2.check();
 }
}
```

Polymorphism

- **Function Overloading :-**

```
import java.util.*;
class A
{
   public void calc (int x)
   {
   System.out.println ("Area of Sq. = " + (x * x));
   System.out.println ("Perimeter of Sq. = " + (4 * x));
   }
 public void calc (int x, int y)
  {
   System.out.println ("Area of Rect. = " + (x * y));
   System.out.println ("Perimeter of Rect. = " + 2 * (x+y));
  }
 Public void calc (double r)
  {
   System.out.println ("Area of circ = " + (3.14 * r * r));
   System.out.println ("Perimeter of circ = " + 2 * (3.14 * r));
  }
 public static void main (String as [])
  {
   Scanner ob = new Scanner (System.in);
   System.out.println ("Enter the side = ");
   int a = ob.nextInt ();
   A obj = new A ();
// obj.calc (a); //1st func.
   System.out.print ("Enter the length = ");
   int len = ob.nextInt ();
  System.out.print ("Enter the breadth = ");
   int br = ob.nextInt ();
// obj.calc (len, br); //2nd func.
   System.out.print ("Enter the radius = ");
   double rad = ob.nextDouble ();
   obj.calc (rad); //3rd func.
  }

}
```

- **Function Overridding :-**

```
import java.util.*;
class Pen
{
  public int x;
  public void input()
  {
   Scanner ob = new Scanner (System.in);
   System.out.print (" Enter the no. = ");
    x = ob.nextInt ();
  }
 public void show()
  {
   System.out.println (" Parent class x = " + x);
  }
}
class Pencil extends Pen
 {
  public int x;
  public void input1()
   {
    Scanner ob = new Scanner (System.in);
    System.out.print (" Enter the no. = ");
    x = ob.nextInt ();
   }
public void show()
  {
   System.out.println (" Child class x = " + x);
   System.out.println (" Parent class x = " + super.x);
  }
public static void main (String as[])
  {
    Pencil ob = new Pencil();
     ob.input();
     ob.input1();
     ob.show();

  }
}
```

- **Super Keyword :-**

Super is a keyword which is used if we want to display parent class data through child class if overridding is there.