```
Class mStatic
{
Public static void main (String [] args)
{
   Sample static. putdata ();
}
}
```

Nesting Methods

```
Class Sample
{
   int k = 5, F = 9;
   void putdata ()
   {
      System . out . Println (k);
   }
   void display ();
   {
      Put data ();
      System . out . Println (F);
   }
}
```

```
class m static
{
  public static void main (String [] args)
  {
    sample s1 = new sample();
    s1.display();
  }
}
```

V.V.I Inheritance:

Deriving a class from a base class or existing class.

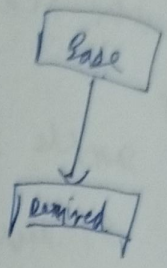
Types of Inheritance:

(i) Single Inheritance.

```
class Base
{
  int a = 3;
  void showdata()
  {
    System.out.println(a);
  }
}
```

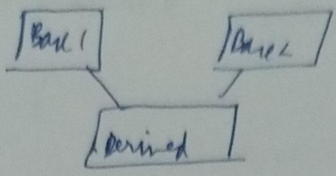
```
Class derived extends Base
{
    int c = 5;
    void showvalue()
    {
        System.out.println(c);
    }
}
Class mBase
{
    Public Static void main (String[]args)
        {
        derived d = new derived();
            d1.showvalue();
            d1.showdata();
        }
}
```

Single



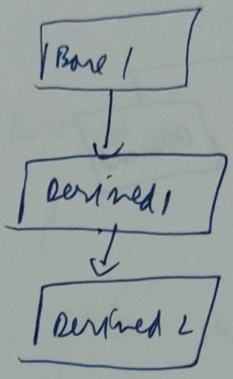
1) Single Inheritance
2) Multiple Inheritance - which is not supported by Java.

Multiple Inheritance is replaced by Interface.

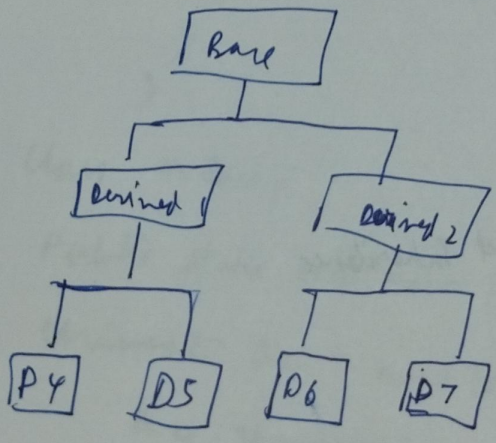


3) Multilevel Inheritance

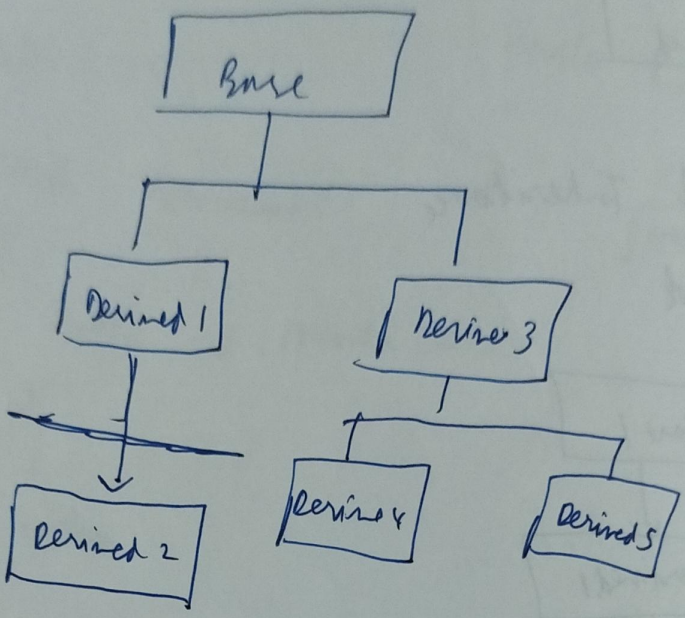
Level by level

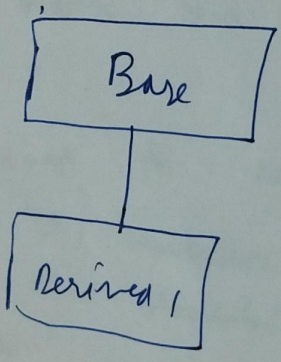


4) Hierarchical Inheritance



Hybrid Inheritance





Class Base Multilevel Inheritance

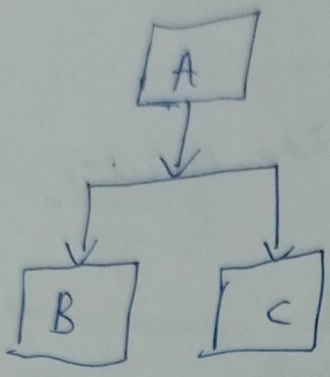
```
{
    int k=4;
        void show ()
        {
        } System.out.println(k);
}
```

```
class Derived1 extends Base
{
    void showdata()
    {
        System.out.println("I am Derived");
    }
}

class Derived2 extends Base
{
    void showdata()
    {
        System.out.println("Derived2");
    }
}

class mBase{
public static void main(String[] args){
Derived2 d2 = new Derived2();
    d2.showvalue();
    d2.showdata();
    d2.show();
}
}
```

Hierarchical Inheritance

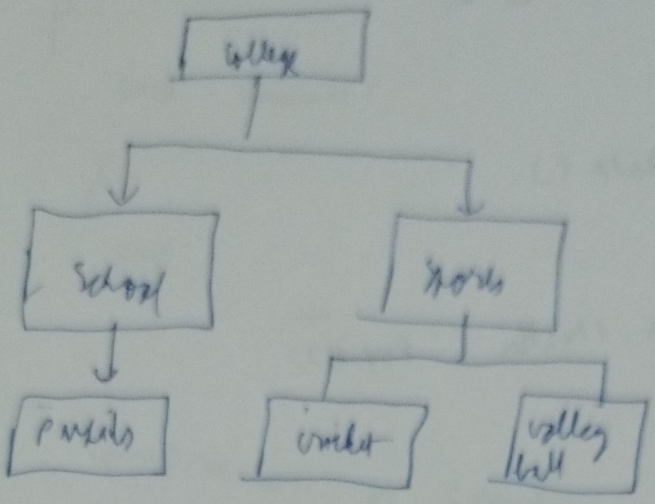


Program

```
Class A
{
    void display()
    {
        System.out.println("I am Base");
    }
}

Class B extends A
{
    void displaydata()
    {
        System.out.println(received);
    }
```

Hybrid Inheritance




```
class College
{
    int x = 4
    void show()
    {
        System.out.println(x);
    }
}

class School extends College
{
    void showdata()
    {
        System.out.println("Derived 1");
    }
}
```

```
class Pupil extends college
{
    void showdata()
    {
    System.out.println("Pupil");
    }
}

class Sports extends college
{
    void showdata()
    {
    System.out.println("Sports");
    }
}

class Cricket extends college
{
    void showdata()
    {
    System.out.println("Cricket");
    }
}
```

```
class volleyball extends college
{
    void showdata()
    {
    sysout
```

Constructor in Inheritance

```
class Base
{
    int a;
    Base (int x)
    {
        a = x;
    }
void putdata()
    {
        System.out.println("a=" + a);
    }
}
```

```
class derived1 extends Base
{
    int y;
derived1(int k, int m)
  {
     super(k);
     y = m;
    }
void putdata()
  {
     super.putdata();
   System.out.println(y);
  }
}

class mderived
{
  public static void main(String[] args)
    {
     derived d1 = new derived(5,6);
     d1.putdata();
    }
  }
```

```
class A
{
    void display()
    {
        System.out.println("JAVA");
    }
}

class B extends A
{
    void display()
    {
        super.display();
        System.out.println("PROGRAM");
    }
}

class mBase
{
    public static void main(String[] args)
    {
        B b1 = new B();
        b1.display();
    }
}
```

JAVA
PROGRAM

CONStructor - super method

overriding method - super keyword

Interface

Abstract class

multiple Inheritance

↳ No. of base class.

only one derived class

```
interface one
{
    int a = 5;
    void display();
}

class mentor implements one
{
    int x = 2;
    void display()
    {
        System.out.Println("HAI");
    }
}
class x12{
public static void main (String[] args)
{
one m1 = new mentor()
    m1.display();
}
}
```

```
interface I1
{
    int x;
   void show();
}
interface I2
{
   int x;
  void putdata();
}
class board implements I1, I2
void putdata()
{
 system.out.println("welcome");
}

class board computer
 {
    Public static void main (String [] args)
    {
        board b1 = new board();
        b1..show();
        b1.putdata();
    }
}
```

```
interface I1
{
  int x;
  void show();
}
interface I2
  {
    int x;
    void putdata();
  }
class board implements I1, I2
{
  int z;
  void show()
  {
  System.out.println("Hello");
  }
}
```

One and another interface => extends is used.

```
class Computer
{
  Public static void main (String[] args)
  {
    I2    obj = new board();
    obj.show();
    obj.putdata();
  }
}
```

Create a three interface in Java program namely school, college, university with a class of home main method class name - house.

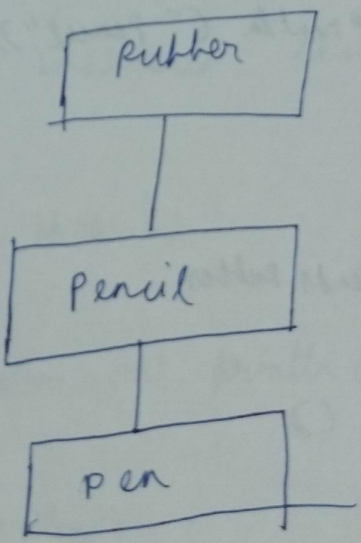
```
interface School
{
  int x;
  void show();
}

interface College
{
  int y;
  void show();
}
interface University
{
  int z;
  void putdata();
}

class home implements School, College, University
void putdata();
{
  System.out.println("welcome");
}
```

```
class home
{
public static void main (String [] args)
 {
   home h1 =      new home ();
   h1.show ();
   h1.putdata ();
 }
}
```

2) create a classname pen from the existing class pencil which was derived under rubber class. Implement the constructor as well as what type of Inheritance is implemented in the above



```
class Rubber
{
    int k = 2;
    void show()
    {
        System.out.println(k);
    }
}

class Pencil extends Rubber
{
    void showdata()
    {
        System.out.println("Pencil");
    }
}

class Pen extends Rubber
{
    void showdata()
    {
        System.out.println("Pen");
    }
}
```

```
class PBase{
public static void main (String [] args){
Per Per P1 = new Per ();
        P1. showvalue ();
        P1. showdata ();
        P1. show ();

}
}

interface name

{
  void show ();
  interface inner interface
  {
  void display ();
  }
}

class Book implements name. inner interface
  {
     void show ()
     {
        System . out . println ("PARENT");
     }
   void display()
   {
   System . out . println ("CHILD");
   }
}
```

```
class Sample
{
public static void main (String [] args)
    {
        Book b1 = new Book ();
        b1.show ();
        b1.display ();
    }
}
```

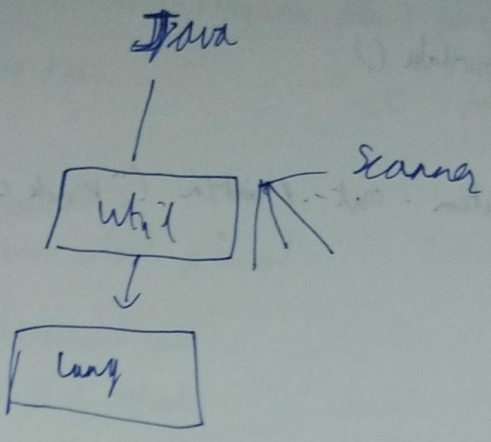
Object Creation

It is possible by creating instance variables

```
{
    name.inner interface
    ni;
    Book b1 = new Book ();
    ni = b1;
    ni.display ();
    ni.show ();
}
}
```

Packages

import java.util.*;



md => making directory

cd => changing directory

Package

User defined Package

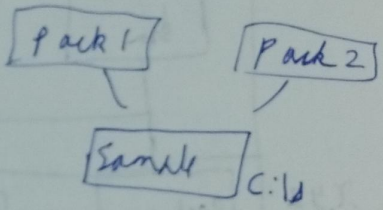
Built-in package (or) predefined Packages

Ex import java.util.*;

Application

② java.lang

↳ String Builder
String Buffer



C:\d

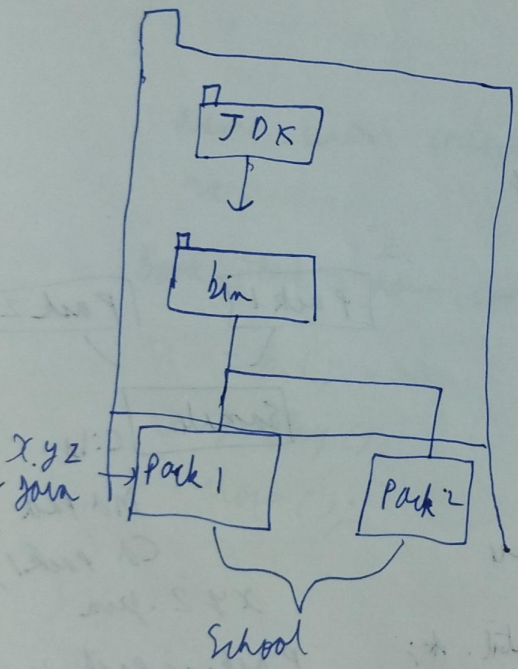
Md pack1;

Cd pack1;

xyz2.java

```
Package pack1;
class xyz2
{
 void display()
 {
  ..
System.out.Print
("PACKAGE");
 }
}
```

```
Package pack3;
class ABC
{
  void showdata()
  {
    System.out.println("Pack2");
  }
}
```




```
import pack1.xyz;
import pack2.ABC
class School
{
  xyz x1 = new xyz();
  ABC a1 = new ABC();
    x1.display();
    a1.showdata();
}
```

C:\> javac School.java

Set the Path

C:\> cd documents

C:\documents> javac c

set path = "C:\Program Files\Java"

set classpath = "   "<.C:\documents> javac

C:\documents>.
    javac school.java ↵

C:\document ↵
    java school

Exceptions

catch (Exception e)

{ System.out.println ("denominator must not be zero" + e);

}

finally

{ int a, b;

int c = a/b;

System.out.println (" " c);

Thread light weight

Thread is a ~~lifetime~~ Process. Multi Thread is a heavy weight Process.

- run()
- start()
- sleep()
- wait()
- notify()

Interface → Runnable

class → Thread

class { mythread implements Runnable

int a;

mythread (int x)

{

a = x;

}

void run()

{

System.out.println (a);

}

Write a java program implementing the concept of exception Handling (user defined and Predefined).

```
import java.util.Scanner;
class Main
{
public static void main (String [] args)
{
  try
  {
    int a,b;
    int c=a+5;
    System.out.println (c);
  }

    catch (Arithmetic exception e)
    {
    System.out.println (" correct the value"+e);
    catch ( Divide by zero Exception ex)
        {
          System.out.println (" Division not be zero"+ex);
        }
  try
    {
      int a=5, b=0;
       int e=a/b;
    }
```

```
class mainthread
{
Public static void main (String [] args)
    {
      mythread mt = new mythread (5);
       mt start ();
    }
}

class mythread extends Thread
{
    int a, b;
    mythread (int x, int y)
          {
            a = x;
            b = y;
          }
    void run ()
      {
        System . out . Println (a);
      }
class mainthread
  {
    Public static void main (String [] args)
    {
      mythread mt = new mythread (5, 6);
      mt. start ();
    }
}
```

```
String query = "Select * FROM customers";
ResultSet resultSet = statement.executeQuery(query);
while (resultSet.next()){
    String name = resultSet.getString("name");
    int age = resultSet.getInt("age");

System.out.println("Name:" + name + "Age:" + age);
```