Module-II

30.03.2019

class and object

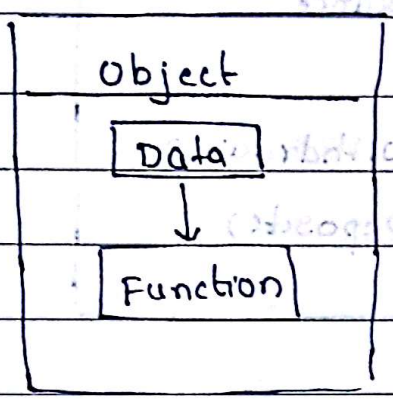
object – critical elt. of oop.

runtime entity

characterstics (Data)

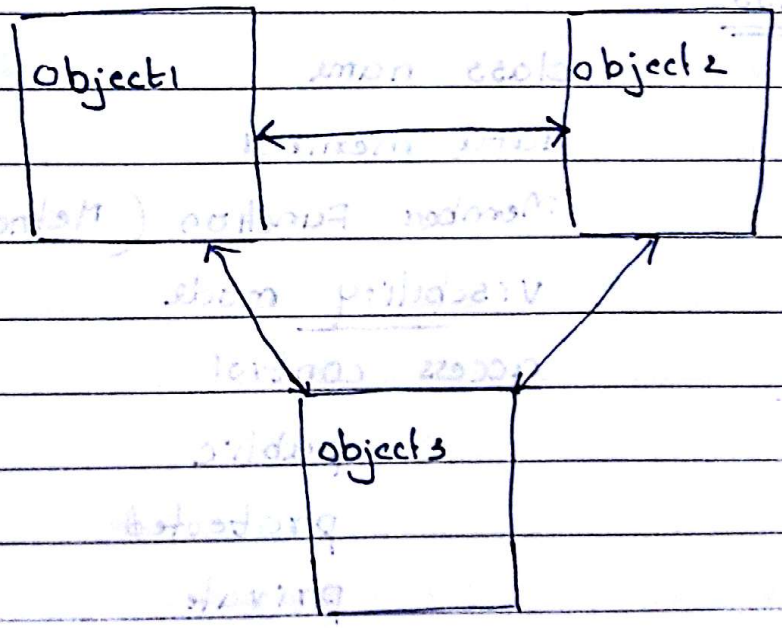
behaviour (Function)

an object need not be a real life entity alone; it could also be a mathematical or vector entity.
eg: matrix in add^n of 2 matrices.
vector in union of 2 vectors.



- Disadvantages of procedure oriented programming:
  (i) Data has no emphasis.
  (ii) Data is not protected.

To overcome the disadvantages of procedure oriented pgming oop was introduced.



There can be many objects & each of the objects communicate with each other.

Scanned by CamScanner

object – customer	object – student
Data	Data
Acc-No	rollno
Name	name
Address	marks
Acc-Type	Function
Balance	calculate()
Function	Print
withdrow()	
Deposit()	

class

Class is an user-defined datatype. However during its implementation, it functions as an in-built datatype.

class name
data member
Member Function (Method)
visibility mode
access control
public
protected
private

```
class  class name
{      Access control  data member declaration
       "               method definition
}
```

Scanned by CamScanner

Scanner

scan ( )

Scanner scan = new Scanner (System.in);

scan nextInt()

nextFloat()

nextDouble()

next()

int a;

a = scan.nextInt()

int a;

Scanner scan = new Scanner (System.in);

System.out.print("Enter a number");

a = scan.nextInt();

Define a class to read and print the details of a student. (rollno, name, mark)

class — student

Data — rollno

name

mark

" Methods

getdata() — read data

Display() — print "

Scanned by CamScanner

```
import java.io.*;
class Student
{
        private int rollno;
        private String name;
        private int mark;
        public void getdata()
        {
                Scanner scan = new Scanner(
                                System.in);
                System.out.print(" Enter rollno"),
                rollno = scan.nextInt();
                System.out.print("Enter name:");
                name = scan.next();
                System.out.print("Enter mark:");
                mark = scan.nextInt();

        }
        public void Display()
        {
                System.out.println("RollNo:" +rollno)
                                    ("Name : "+name);
                                    ("Mark: "+mark),
        }
}
```

Scanned by CamScanner

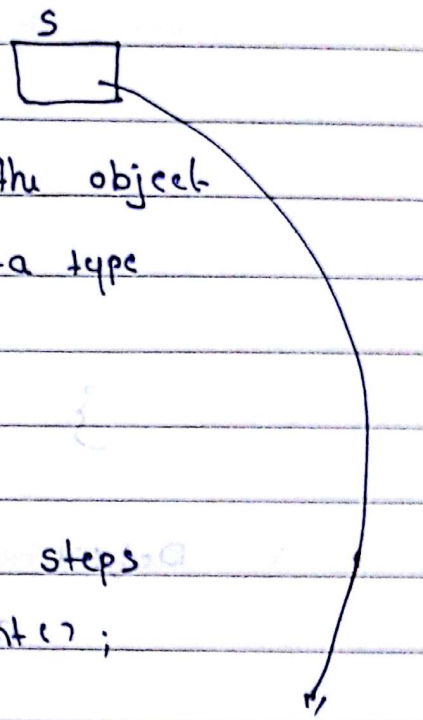
object creation

define a reference name

Allocate memory for the object

class name object name;

eg student s;



new op^n is used

to allocate memory for the object

reference name = new data type

s = new student()

eg: Student s;

s = new student();

we can combine these 2 steps

student s = new student();

Accessing Members of an object

object name . member name

. member op^n

s.rollno

s.getdata()

getdata()

Display()

Scanned by CamScanner

```
class Test
{
    public static void main(String
                        args[])
    {
        Student s = new Student();
        s.getdata();
        s.Display();
    }
}
```

Q Define a class To represent a complex no
a + bi real imag Float

Ans:
```
import java.io.*;
class Complex {
    private float a;
    private float b;
    public void ~~display (int i, int j)~~ getdata()
    {
        Scanner sc = new Scanner(System.in);
        System.out.println("Enter real part:");
        a = sc.nextFloat();
        System.out.println("Enter imaginary part:
```

Scanned by CamScanner

```
    b= sc. nextFloat ( );
  }
  public void display ( )
  {
        System. out. print ( a + "+" + b + "i" );
  }
}


class Test {
  public Static void main ( String args[ ])
  {
        Complex c = new Complex ( );
        c. getdata ( );
        c. display ( );
  }
}
```

Constructors

```
class factorial
{   private int n, f;
    public void getdata ()
    { Scanner s = new Scanner (System. in);
      System. out. print ( " Enter a no" )
      n = s. nextInt ( )
    }
```

Scanned by CamScanner

```
public void fact()
{       int i;
        f=1;
        for(i=1; i<=n; i++)
        {   f= f*i;
        }
}
public void Display()
{   S.o.p("factorial = " + f);
}
}
```

int n, i, f=1

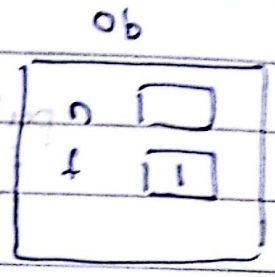
```
class factorial
{   private int n, f;
    public void initialize()
        {   f=1;
        }
    public void getdata()
    {
    }
    public void fact()
    {   int i;
        for(i=1; i<=n; i++)
        {   f= f*i;
        }
    }
```

Scanned by CamScanner

```
public void Display()
{
}
}

class factorial
{ private int n,f;
  public factorial()
  { f=1;
  }
  public void getdata()
  { S


  }
  public void fact()
  {


  }
  public void Display()
  {
```

Scanned by CamScanner

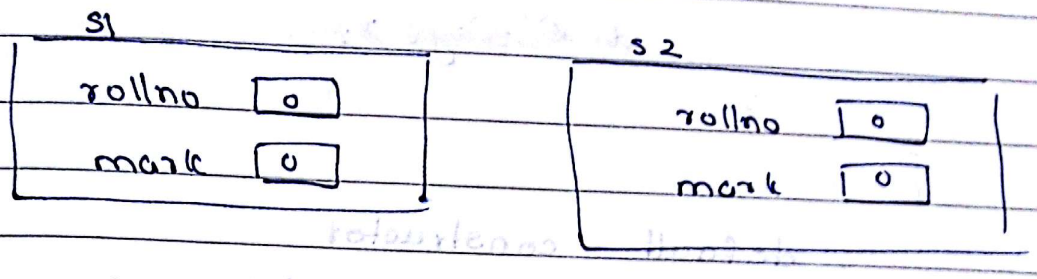
```
    }
}

class Test
{
    public static void main (String args[])
    {
        Factorial ob = new Factorial()
        ob.getdata();
        ob.fact();
        ob.Display();
    }
}
```




```
class student
{   private int rollno, mark;
    public Student()
      { rollno = 0;
        mark = 0;
      }

    =
}
```

Scanned by CamScanner

Student s1 = new Student();
" s2 = "



WAP to read and print todays date
Use constructor to set the date
30/03/2019

```
import java.io.*
class ~~Test~~ Date1
{
    Private int day, month, year;
    public ~~Test~~ Date1 ( ) {
        day = 30;
        month = 03;
        year = 2019;
    }
    public void Display( ) {
        S.o.p ("Today's date is :" + day + "/" + month + "/" +
                year);
    }
}
class Test {
```

Scanned by CamScanner

```
    public static void main (String args[])
    {
        Date1 d = new Date1();
        d.Display();
    }
}
```

default constructor

parameterized constructor

```
class student
{   private int rollno, mark;
    public student( int r, int m)
        { rollno = r;
          mark = m;
        }
    public void Display()
        {
            s.o.p ("Roll no:" + r);
            s.o.p ("Mark:" + m);
        }
}

class Test
{
```

Scanned by CamScanner

```
public static void main( )
{
    Student s1 = new Student (111, 75);
        "   s2 =    "   (112, 60);
    s1. Display();
    s2. Display();
}
```

s1

rollno	111
mark	75

s2

rollno	112
mark	60

Dynamic initialization

```
main(
{ int r, m;
    Scanner s1 = new Scanner (System.in)
    System.out.print ("Enter rollno"),
    r = s1.nextInt(),
    System.out.print ("Enter mark"),
    m = s1.nextInt()
    Student s = new Student (r, m);
    s.Display();
}
```

Scanned by CamScanner

```
class Numbers
{ private int c;
    public void sum (int a, int b)
      { int c = a+b;
      }
    public void Display ()
      { s.o.p ("Sum = " + c);
      }
}
class Test
{ public static void main (String args[])
    { Numbers ob = new Numbers()
        ob.sum (76, 95);
        ob.Display();
    }
}
```

Overloading

function overloading (Method Overloading)

operator " X

Scanned by CamScanner

polymorphism

circle πr^2

Rectangle lb

triangle $\frac{1}{2}bh$

square a^2

```
class shape
{
  public float Area(float r)
  {  float a;
     a = 3.14*r*r;
     return(a);
  }
  public int Area(int l, int b)
  {  int a;
     a = l*b;
     return(a);
  }
  public float Area(float b, float h)
  {  float a;
     a = 0.5*b*h;
     return(a);
  }
  Public int Area(int a)
  {  int ar;
     ar = a*a;
     return(ar);
}
}
```

Scanned by CamScanner

```
class Test
{ public static void main (String args[])
  int n, l, b, float base, leng, r;
  {   Shape S = new Shape()
      float a = S. Area(r);
      System. out. println ("Menu \n 1. circle
      2. rectangle \n 3. triangle \n 4. square");
      System. out. print ("Enter choice:");
      Scanner sc = new Scanner(System. in);
      n = sc. nextInt();
      switch (n)
      {
        case 1 : System. out. print ("Enter radius
                 r = sc. nextFloat ();
                 S.O.P ("Area = "+ s.Area (r));
                 break;
        case 2: System. out. print ("Enter length
                 l = sc. nextInt ();
                 ~~S.O.P("Area = r+ Area(~~
                 S.O.P("Enter breadth");
                 b = sc. nextInt ();
                 S.O.P("Area = " + s.Area (l, b));
                 break ();
        case 3: S.O.P("Enter base : ");
                base = sc. nextFloat ();
                S.O.P("Enter height : ");
                leng = sc. nextFloat ();
                S.OP ("Area = " + s.Area (base, l));
                break ();
```

Scanned by CamScanner

```
case 4: s.o.p ("Enter side:");
        a = sc.nextInt();
        s.o.p ("Area = " + m.Area(a));  break;
default : s.o.p ("Invalid choice");
}}}
```


Calculate volume of a box, cylinder and cube

```
class Volume
{   public ~~float~~ void ~~box~~ vol (float l, float b, float h)
    {       float ~~x~~ v;
            v = l * b * h;
            s.o.p ("Volume=" + v);
    }
    public ~~float~~ void vol (float r, float h)
    {       float v;
            v = 3.14 * r * r * h;
            s.o.p ("Volume =" + v);
    }
    public void vol (float a)
    {       float v;
            v = a * a * a;
            s.o.p ("Volume=" + v);
    }
}

class Test {
    P s v m (String args [ ])
    {       int n; ~~l, b, h;~~ float l, b, h;
            Volume v = new Volume ();
            s.o.pln("Menu:\n 1. Box \n 2. Cylinder \n 3. Cube");
            s.o.p("Enter choice");
            Scanner sc = new Scanner(System.in);
            n = sc.nextInt ();
```

Scanned by CamScanner

31.03.2019

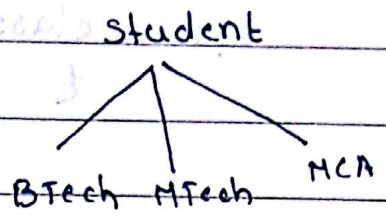
```
switch (m)
{
    case 1: s.o.p ("Enter length: ");
            l sc. nextFloat ();
            s.o.p ("Enter breadth: ");
            b = sc. nextFloat ();
            s.o.p ("Enter height: ");
            h = sc. nextFloat ();  ~~s. Area(l, b, h)~~
            v. Volume (l, b, h);
            break;
    case 2: s.o.p ("Enter radius: ");
            l = sc. nextFloat ();
            s.o.p ("Enter height: ");
            h = sc. nextFloat ();
            v. Volume (l, h);
            break;
    case 3: s.o.p ("Enter side length: ");
            l = sc. nextFloat ();
            ~~s.o.p~~ v. Volume (l);
            break;
    default: System. out. println ("Invalid
                                   choice");
    }
  }
}
```

Scanned by CamScanner

Module – III

Inheritance – mechanism by which a class inherits the data members & fns. of another class.

BTech
MTech
MCA



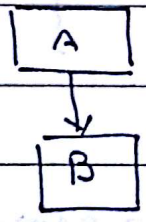
Base class (Super)
Derived " (sub class)

oop – code reusability

✓ 1) Single Inheritance
2) Multiple " — Not directly supported
✓ 3) Multi level "
✓ 4) Hierarchical "
5) Hybrid —

o Inheritance :
– code reusability
– describes the relation b/w classes.

Single Inheritance



1 : 1

```
class A
{ _
}
```

extends ⇒ that class is not an independent class.

```
class derived class name extends base class nar
{ =
}
```

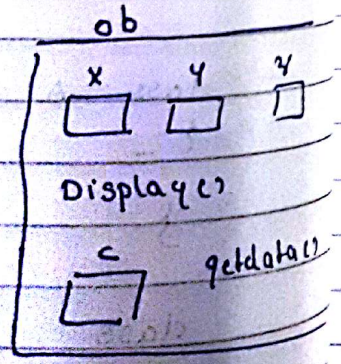
Scanned by CamScanner

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

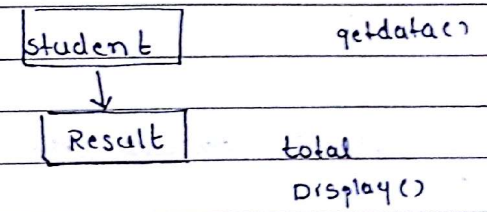
eg:

```
class A
{ private int a,b;
  protected int c;
  public void getdata()
  {
  =
  }
}
class B extends A
{ private int x,y,z;
  public void Display()
  {
  =
  }
}
```

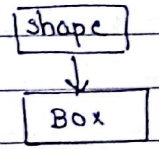
B ob = new B()



1. Define a class student with data members rollno, name, mark in 2 subjects. Derive a class Result from class student and calculate total mark.



2. Define a class shape with 2 data members length and breadth. Derive a class Box from class shape and calculate Area and volume.



Ans:

```
import java.io.*
class Student {
  protected ~~private~~ int rollno;
  protected ~~private~~ String name[25];
  protected ~~private~~ ~~mark~~ float mark1, mark2;
  public void getdata ( )
  {
    Scanner sc = new Scanner (System.in);
    s.o.p ("Enter roll no:");
    rollno = sc.nextInt ();
    s.o.p ("Enter name:");
    name = sc.next ();
```

Scanned by CamScanner

```
        s.o.p ("Enter marks in subject 1:")
        mark1 = sc.nextFloat();
        s.o.p ("Enter marks in subject 2:")
        mark2 = sc.nextFloat();
    }
}
class Result extends Student
{
    private int total;
    public void Total {
        total = mark1 + mark2;
    }
    public void Display {
        s.o.pln("Rollno:" + rollno);
        s.o.pln("Name:" + name);
        s.o.pln("Mark1:" + mark1);
        s.o.pln("Mark2:" + mark2);
        s.o.pln("Total=" + total);
    }
}
class Test {
    Psvm (String args[])
    {   Student s = new Student();
        Result r = new Result();
```



```
        r.getdata();
        r.Total();
        r.Display();
    }
```

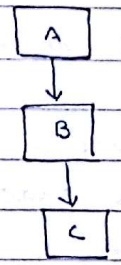
Ans 2: import java.io.*

```
class Shape {
    protected float length, breadth;
    public void getdata()
    {
        Scanner sc = new Scanner (System.in
        System.out.print ("Enter length:")
        length = sc.nextFloat();
        System.out.print ("Enter breadth:");
        breadth = sc.nextFloat();
    }
}
class Box extends Shape {
    private float height;
    public void getdata1()
    {   Scanner sc = new Scanner (System.in
        s.o.p ("Enter height:");
        height = sc.nextFloat();
    }
```

Scanned by CamScanner

```
public void Area ( )
{   float a;
        a = length * breadth;
        s.o.p("Area = " + a);
    }
public void Volume ( )
{   float v;
        v = length * breadth * height;
        s.o.p("Volume = " + v);
    }
}
class Test {
  public Static void main (String args[])
  {
        Shape s = new Shape ( );
        Box b = new Box ( );
        b. getdata ( );
        b. getdata1( );
        b. Area ( );
        b. Volume ( );
  }
}
```


Multi level Inheritance

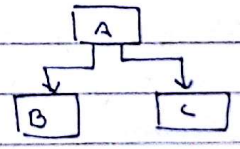


1:1

3. Define a class Student (rollno, name, mark1, mark2) Derive a class Sports from class student with data member grace mark. Derive a class Result from class sports. Calculate total mark

student
↓
sports
↓
Result

Hierarchical Inheritance



1 : M

4. eg: Define a class shape with data member. ~~Def~~ Derive 2 class circle and Rectangle from class shape. calculate Area of circle and Rectangle

only one.

Scanned by CamScanner

Ans 3:
```
class Student {
    protected int rollno;
    protected String name[25];
    protected float mark1, mark2;
    public void getdata () {
        Scanner sc = new Scanner(System.
        System.out.print ("Enter roll no:");
        rollno = sc.nextInt ();
        s.o.p ("Enter name:");
        name = sc.next ();
        s.o.p ("Enter mark1:");
        mark1 = sc.nextFloat ();
        s.o.p ("Enter mark2:");
        mark2 = sc.nextFloat ();
    }
}

class Sports extends Student {
    protected float gracemark;
    public void grace () {
        Scanner sc = new Scanner (System.in
        s.o.p ("Enter grace marks");
        gracemark = sc.nextFloat ();
    }
}
```



```
class Result extends Sports {
    private float total;
    public void findTotal () {
        total = mark1 + mark2 + gracemark;
    }
    public void Display () {
        s.o.p ("Total marks =" + total);
    }
}

class Test {
psvm (String args[])
{
    Result r = new Result ();
    r.getdata ();
    r.grace ();
    r.findTotal ();
    r.Display ();
}
}
```

Ans 4:
```
class Shape {
    protected float r; }
    ~~public~~
class circle extends Shape {
    public void getdata1 () { Scanner sc = new Scanner
                                (System
        s.o.p ("Enter radius:");
        r = sc.nextFloat ();
```

Scanned by CamScanner

```
public void Area1( )
{       ~~float a~~;
        a = 3.14 * r * r ;
}
public void Display1( )
{       s.o.p (" Area=" + a);
}
}
class Rectangle extends Shape {
    ~~public void get~~
  private float b, a;
  public void getdata2 ( )
  {
      Scanner sc = new Scanner (System.in);
      ~~sc = nex~~ s.o.p ("Enter length: ");
       r = sc. nextFloat ( );
       s.o.p ("Enter breadth :");
       b = sc. nextFloat ( );
  }
  public void Area2 ( )
  {   ~~float a~~;
       a = ~~b~~ r * b;
  }
public void Display2( ) { s.o.p ("Area= " +a
```

```
class Test {
  prvm (String arg[ ])
  {
      Circle c = new Circle ( );
      Rectangle rect = new Rectangle ( );
      c. getdata1 ( );
      c. Area1( );
      c. Display1( );
      rect. getdata2 ( );
      rect. Area2 ( );
      rect. Display2 ( );
  }
}
```

Method Overriding

```
class A
{   public vord Display()
      { s.o.p ( " Display Base class");
      }
}

class B extends A
  {   public vord Display ()
        { s.o.p( " Display Derived class");
        }
  }
```

Scanned by CamScanner

```
class Bank
{
    public float getIntRate ()
        { return (8);
        }
}

class SBI extends Bank
  {
      =
      public float getIntRate ()
          { return (14);
          }
  }
class CSB extends Bank
  { =
  }

class Fed extends Bank
  { public float getIntRate ()
        { return (12)
        }
  }
```


Abstract class

class with no object

```
abstract class student
{
    =  abstract void display ();
}
```

compulsory sub-class

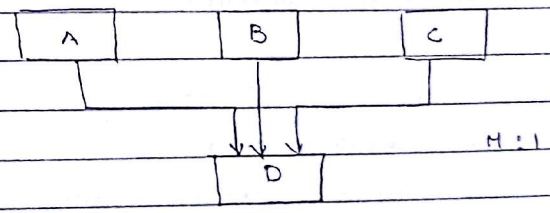
compulsory method overriding.

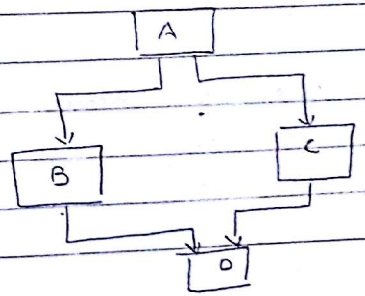
if ; at end of fn. heading ⇒ by default its abstract ⇒ requires method overriding.

∵ an abstract class has no object

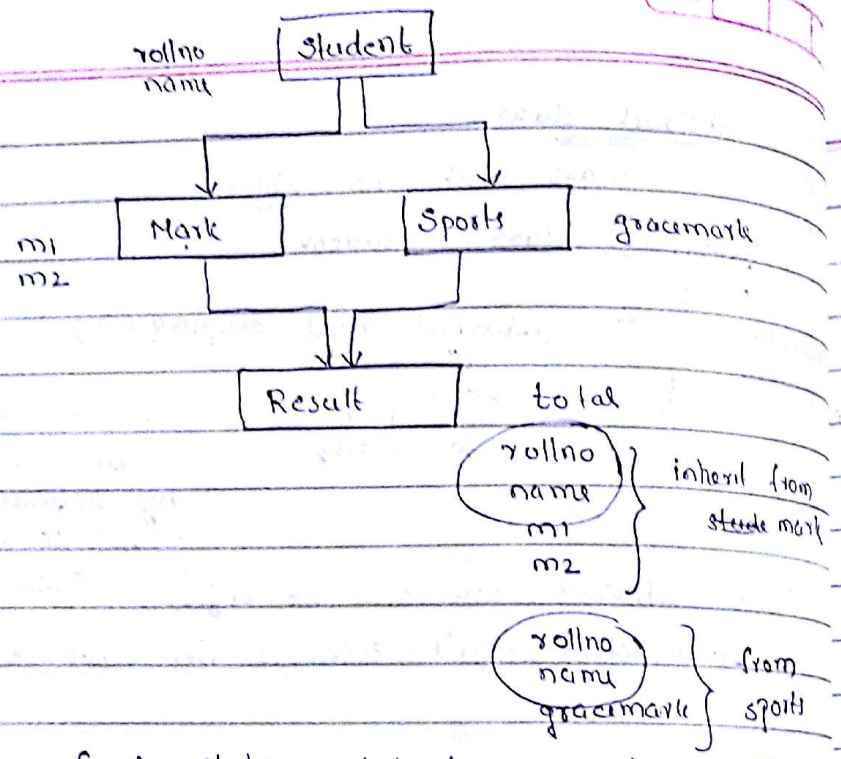
⇒ constructors aren't allowed in abstract class.

interface





Scanned by CamScanner



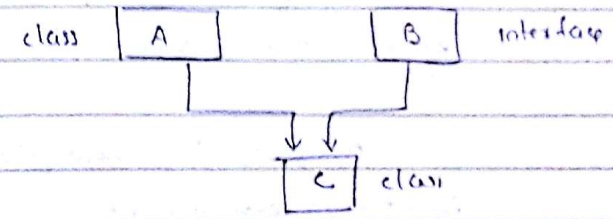
final static data type member = value

static final float pi = 3.14;

return type function name (arg. list);

```
Interface interfacename
{ data member
    methods ~~defini~~ declaration
}

interface Circle
{ static final float pi = 3.14;
    float Area (float );
}
```


implements




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

Define a class student with data member rollno name and mark in 2 Subjects. Define an interface Sports with one data member grace mark. The gracemark is constant and its value is 5. Derive a class Result from class student and Interface sports

```
class Student {
protected int rollno;
```

Scanned by CamScanner

```
protected String name[25];
protected float mark1, mark2;
public void getdata ()
{
    Scanner sc = new Scanner(System.in);
    ~~sollno=sc.nextInt();~~
    s.o.p ("Enter sollno:");
    sollno = sc.nextInt();
    s.o.p ("Enter name:");
    name = sc.next();
    s.o.p ("Enter mark1 :");
    mark1 = sc.nextFloat();
    s.o.p ("Enter mark2:");
    mark2 = sc.nextFloat();
    }
}
interface Sports {
    final static float gracemark=5;
}
class Result extends Student implements Sports {
 private float total;
 public void gettotal ()
 {      total = mark1 + mark2 + gracemark;
 }
```

```
public void Display ()
 {   s.o.p ("Total marks =" + total);
 }
}
class Test {
psvm ( String args[ ])
{
    Result r = new Result ();
    r.getdata ();
    r.gettotal ();
    r.Display ();
}
```

- Difference b/w class & interface :

	Class	Interface
1.	The members of a class can be constant / variables	The members of an interface are always declared as constant ie their values are final.
2.	The class definition can contain the code for each of its methods, ie, methods can be abstract / non-abstract.	The methods in an interface are abstract in nature, ie, there is no code associated with them. It's later

Scanned by CamScanner

		defined by the class that implements the interface.
3.	It can be instantiated by declaring objects.	It can't be used to declare objects. It can only be inherited by a class.
4.	It can use various access specifiers like public, private, or protected.	It can only use the the public access specifier.

c = a|b ;

int a[] = new int[100]

S.o.println(a[i]) i / 108

Hit the exceptron

Throw "

Catch "

Handle "

Scanned by CamScanner

```
try                    {
                        =
catch                  }
    try                        class Numbers
    {                          {
      =
                               public static void main(
    }                          {
                                   float a, b, c
                                   Scanner s = new Scanner(
                                                System.in)
25  0                              s.o.p( "Enter 2 nos")
25  5                              a = s.nextFloat();
                                   b = s.nextFloat();
                                   try
                                   { c = a/b;
                                     s.o.p(c);
                                   }
                                   catch( ArithmeticException e)
                                   {
                                     s.o.p( "Division by zero");
                                   }

                               }
```

Scanned by CamScanner

```
try
{
    c=a/b;
    S.o.p(c)
    S.o.p(a[i])
}
catch(ArithmeticException e)
{
    S.o.p("Division by zero");
}
catch(ArrayIndexOutOfBoundException e1)
{
    S.o.p("Index no. is out of
                range");

}
finally
{
}
```

Scanned by CamScanner

```
class ErrorException
{
  =
}

main()
{      ErrorException  E = new ErrorException
  =
  try
  {
     if (b==0)
        throw  new ErrorException()
     =
  }
  catch(ErrorException  e)
     {
       =
     }
```

throws IOException

Scanned by CamScanner

```
try
{
    =
    try
    {
        =
    }
    catch( )
    {
        =
    }
}
```

```
try
{
    =
}
catch( )
{
    =
}
```

```
try
{
    sum()
    =
}
catch
```

```
sum()
{
    try
    {
        =
    }
    catch()
    {
        =
    }
}
```

Scanned by CamScanner