OOPS

Class: It contains members and methods.

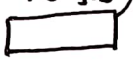
object - class variable.

Difference between C and Java

C	Java
1. Structure	1. object oriented
2. Platform -dependant	2. Platform-independant.
3. Pointer	3. Object
4. function	4. method

: } member access operature

C (DMA)

1. malloc →  10GB → Garbage value
   allocate meomeory (only Block)
2. calloc →  10GB
   Block with cells.
3. realloc → appled to both malloc and calloc.
4. free → destroy.

Java

1. Dynamic (new).

import java.io.*; (import keyword)

Nameing conversion:

1. In java all the class names, first letter start with upper key.

e.g: class StudentInformation.

2. All the methods in java is following the camel case.

camelcase:

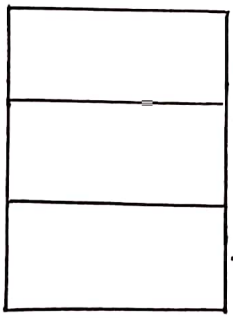
findArea
L U

3. In java, if all the letters are upper case are called as constant (final).

eg: class StudedentInformation

```
{
  final int AGE;
}
```

class diagram contains three parts:



➡ class will be created inside package.
➡ The package contains only class.
➡ template/Libray class – without main

Variables (3 types):

1. local variables - { }
2. instance variable
3. static variable

instance variable are called class member. static variable are declare only instance variable.

static keyword:

It can be used as members, methods, nested class and static block.

Acces specifer:

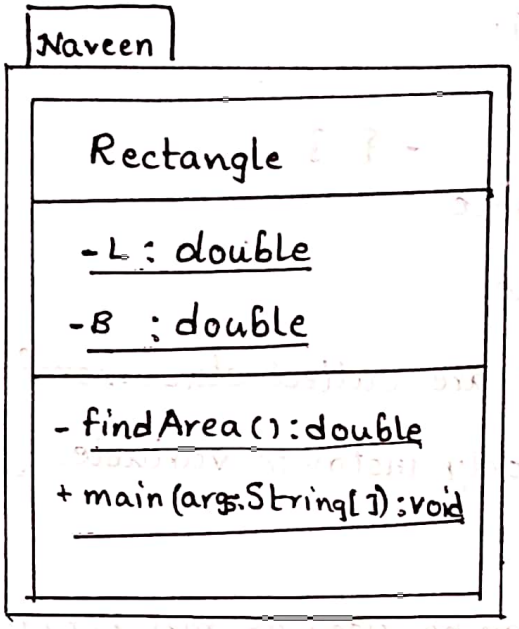
- Private — class level private. (inside class only)
~ default — Package level security.
# Protected — same package or different package child class.
+ Public — any class or package can access.

If any members or methods are underlind are ~~class~~ called as static.

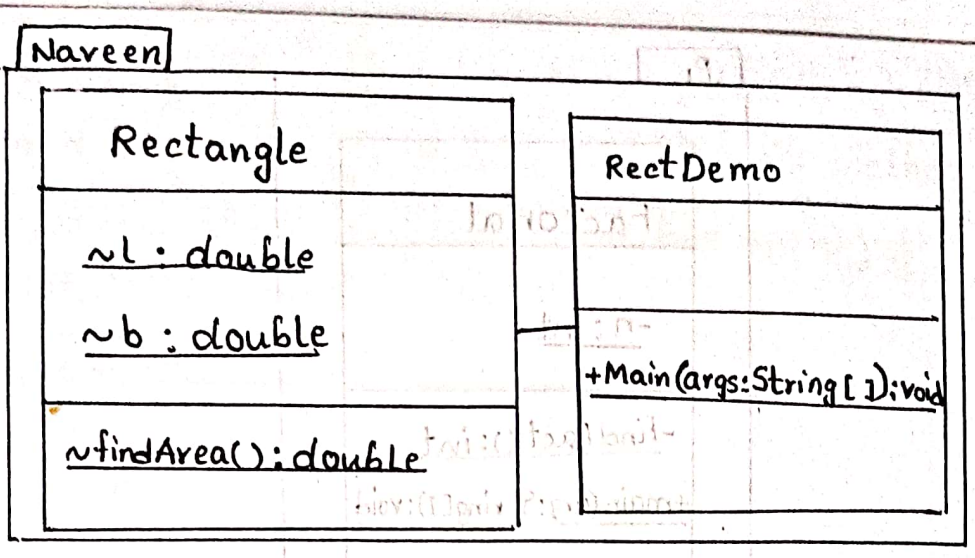
java file name as class name. Static members or methods can be accessed without objects. Static method can access only static member. Outside the class static members or methods can be accessed through class. name.

— Associate line

→ Parent line




```
Package Naveen;

class Rectangle
{
    Private Static double L;

    Private Static double B;

    Private Static double findArea()
    {
        return L*B;
    }
    Public Static void main(String args[])
    {
          L=12;
          B=16;
        System.out.println("area= "+findArea());
    }

}
```



Package Naveen;

Class Rectangle

{

Static double l;

Static double b;

Static double findAreac)

{

return l*b;

}

}

Rectangle.java.

Package Naveen;

Class RectDemo

{

Public Static voidmain(String args[])

{

Rectangle.l=12;

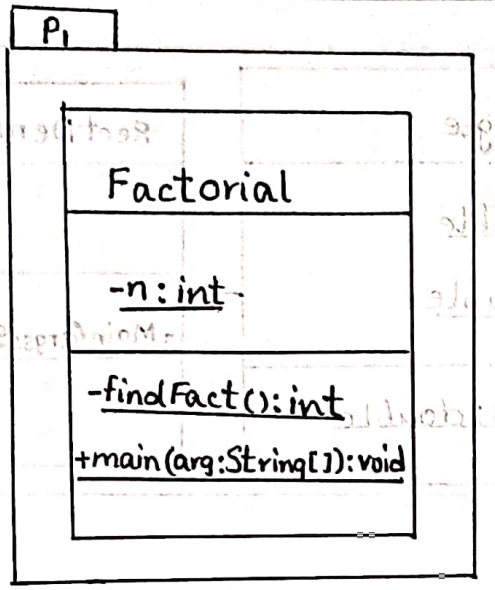
Rectangle.b=16;

System.out.println("area = " + Rectangle.findArea());

}

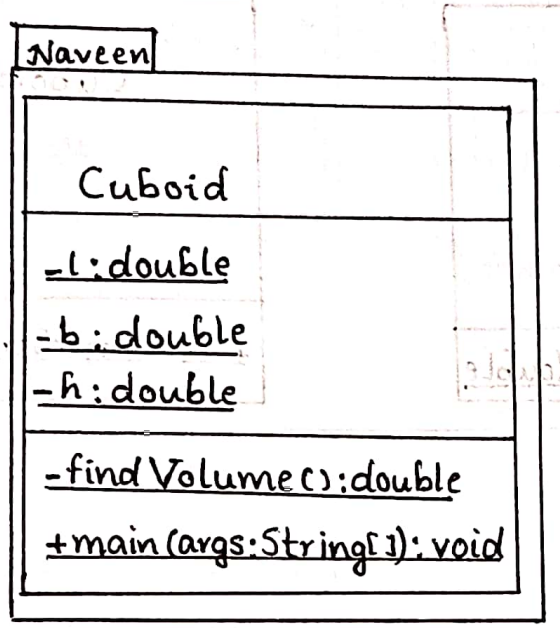
}

RectDemo.java



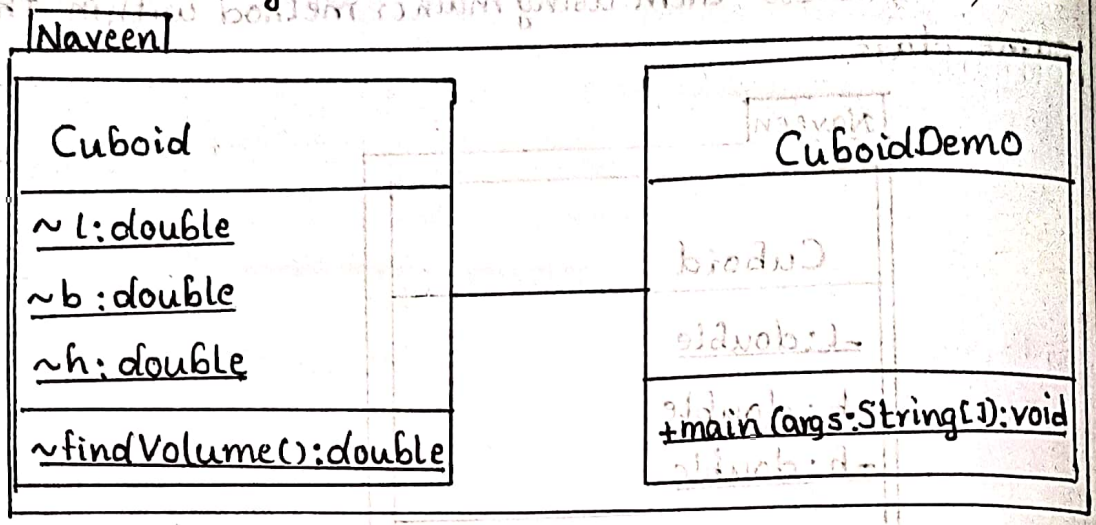

```
package P1;
class Factorial
{
  private static int n;
  private static int findFact()
  {
      int i,f=1;
      for (i=1; i<=n;i++)
        f=f*i;
       return f;
  }
  public static void main (String args[])
  {
      n=5;
      System.out.println (" Factorial= " +findFact());
  }
}
```

Write a Cuboid class with 3 static variables length, breadth and height of type double and a static method volume(), access them using main() method within the same class.



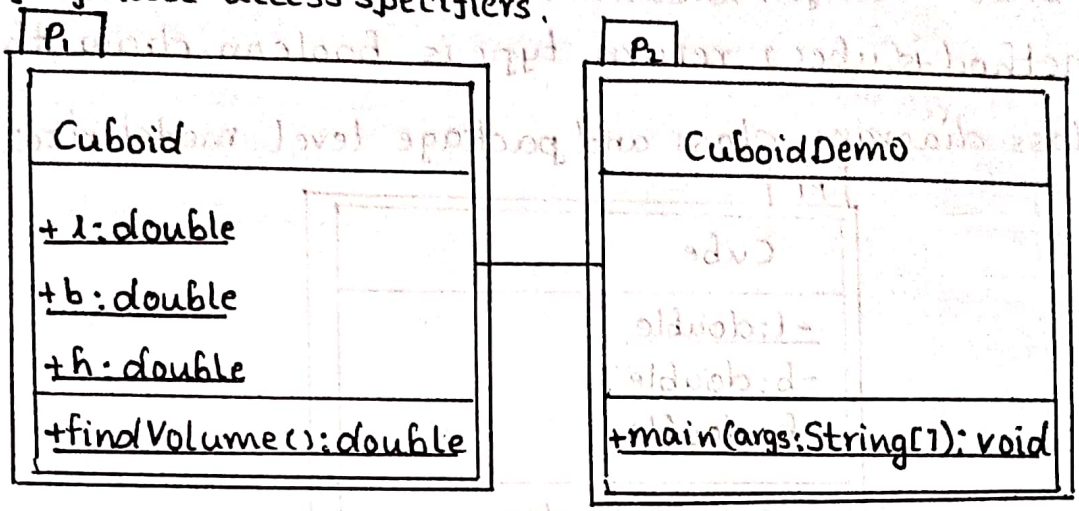

```
package Naveen;
class Cuboid.
{
   private static double l,b,h;
   private static double findVolume()
   {
       return l*b*h;
   }
   public static void main(String args[])
   {
       l=12;
       b=15;
       h=17;
       System.out.println("Cuboid volume="+findVolume());
   }
}
```

Write a Cuboid class with 3 static variables length, breadth, and height of type double and a static method volume(), access them using main() method with another class demo.




```
package Naveen;
class Cuboid
{
   static double l,b,h;
   static double findVolume()
   {
      return l*b*h;
   }
}
package Naveen;
class CuboidDemo
{
   public static void main(String args[])
   {
      Cuboid.l=12;
      Cuboid.b=15;
      Cuboid.h=17;
      System.out.println("Volume = "+Cuboid.findVolume());
   }
}
```

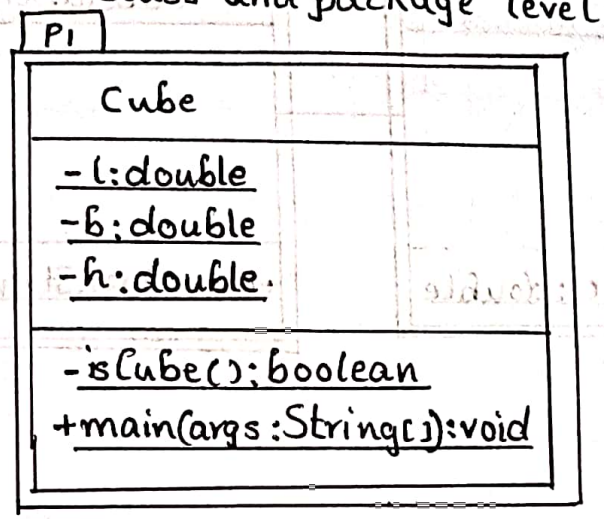
Modularize the Cuboid class to a package level with appropriate access specifiers.



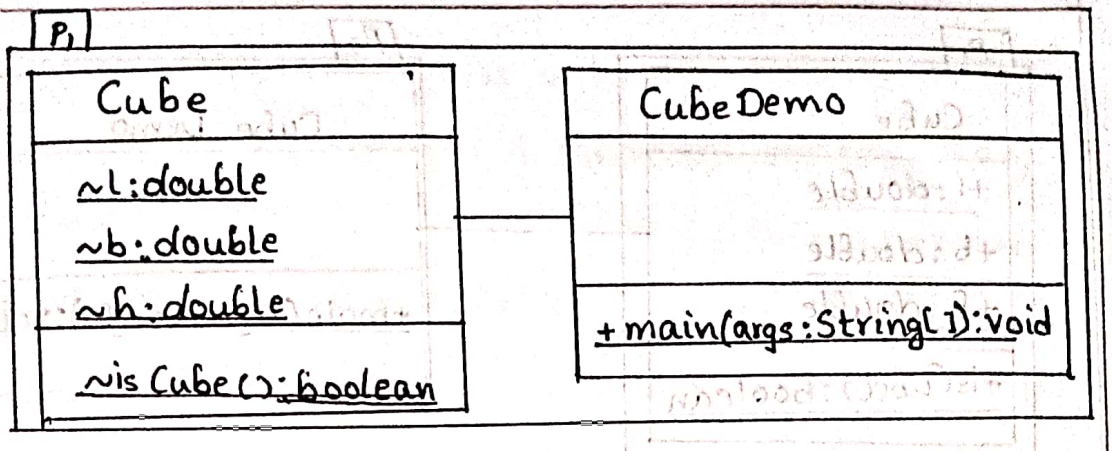

```
package P1;
class Cuboid
{
  public static double l,b,h;
  public static double findVolume()
  {
    return l*b*h;
  }
}
```

```
package P2;
import P1.Cuboid;
class CuboidDemo
{
  public static void main(String args[])
  {
     Cuboid.l=12;
     Cuboid.b=15;
     Cuboid.h=17;
     System.out.println("Volume="+Cuboid.findVolume());
  }
}
```

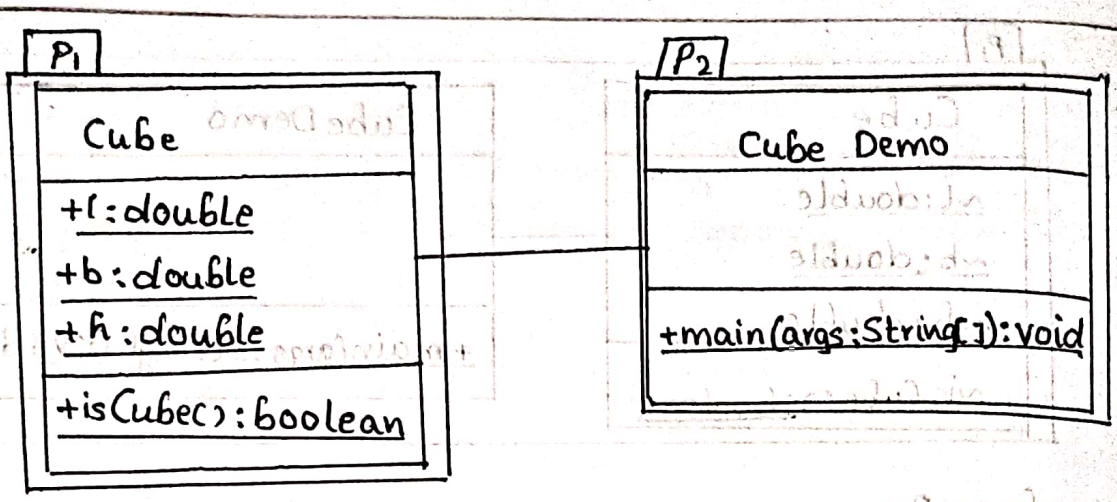
develop a java program. class name Cube, it has 3 static length, breadth. and height with double. method isCube(), return type is boolean. draw the class diagram class and package level modelerisation.




```
package P1;
class Cube
{
   private static double l,b,h;
   private static boolean isCube()
   {
      if(l==b && b==h)
         return true;
         else
          return false;
    }
   public static voidmain(String args[])
   {
        l=12;
        b=12;
        h=12;
       if(isCube())
        System.out.println("It is Cube.");
         else
          System.out.println("It is not Cube");
    }
}
```



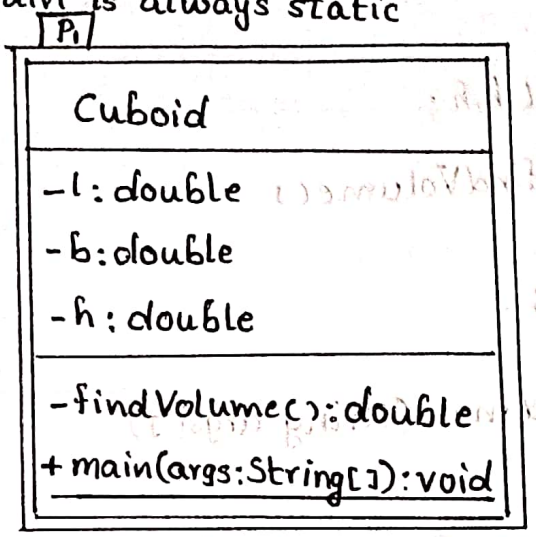

```
package P1;
class Cube
{
    static double l,b,h;
    static boolean isCube()
    {
      if (l==b && b==h)
        return true;
      else
        return false;
    }
}
package P1;
class CubeDemo
{
  public static void main(String args[])
  {
      Cube.l=12;
      Cube.b=12;
      Cube.h=12;
      if(Cube.isCube())
        System.out.println("It is Cube");
      else
        System.out.println("It is not a Cube");
  }
}
```




```
package P1;
class Cube
{
  public static double l,b,h;
  public static boolean isCube()
  {
     if(l==b && b==h)
        return true;
     else
        return false;
  }
}
package P2;
import P1.Cube;
class CubeDemo
{
  public static void main(String[] args)
  {
     Cube.l=5;
     Cube.b=5;
     Cube.h=5;
    if(Cube.isCube())
      System.out.println("It is Cube");
    else
      System.out.println("It is not Cube");
  }
}
```

Object: It is instance class. with the help of object with can access non-static members and non-static methods.

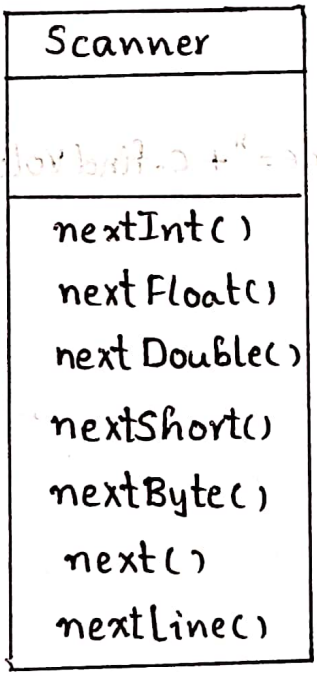
Note: Main is always static



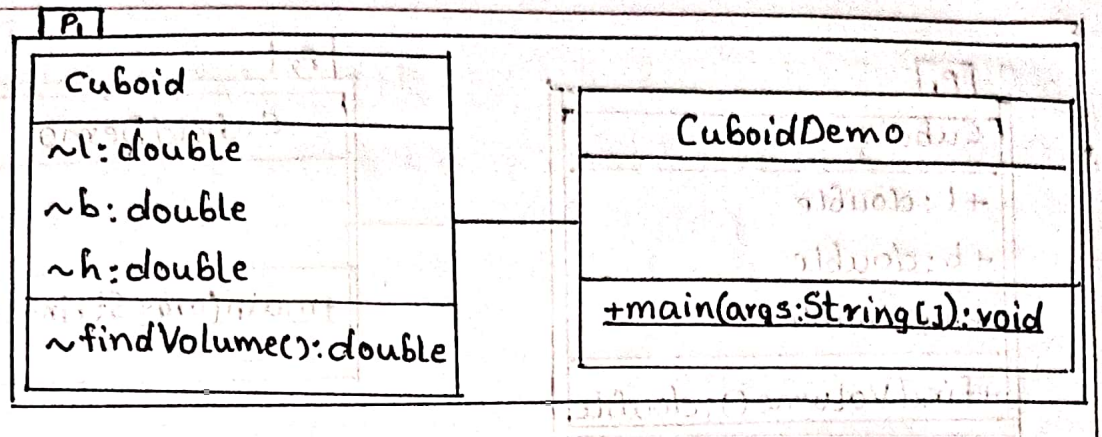

```
Cuboid c = new Cuboid ();
```

class name ↑ obj ↑ DMA ↓ constructor ↓

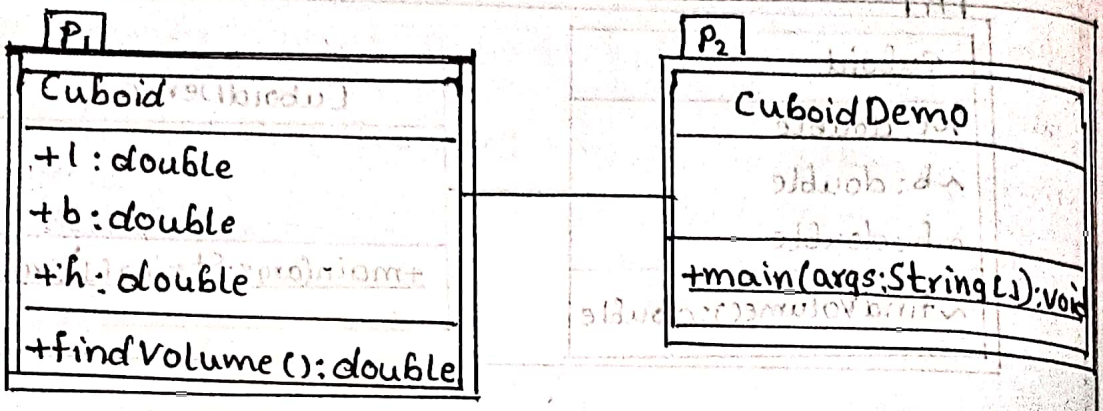
constructor: class name as constructor.



```
package P1;
import java.util.Scanner;
class Cuboid
{
   private double l,b,h;
   private double findVolume()
   {
      return l*b*h;
   }
   public static void main(String args[])
   {
      Scanner sc= new Scanner(System.in);
      Cuboid c=new Cuboid();
      c.l = sc.nextDouble();
      c.b=sc.nextDouble();
      c.h=sc.nextDouble();
      System.out.println("Volume="+ c.findVolume());
      sc.Close();
   }
}
```




```
package P1;
class Cuboid
{
  double l,b,h;
  double findVolume()
  {
     return l*b*h;
  }
}
package P1;
import java.util.Scanner;
class CuboidDemo
{
   public static void main (String[] args)
   {
      Scanner sc=new Scanner(System.in);
      Cuboid c = new Cuboid();
       c.l=sc.nextDouble;
       c.b=sc.nextDouble;
       c.h=sc.nextDouble;
      System.out.println("Volume="+c.findVolume());
     sc.close();
   }
}
```




```
package P1;
class Cuboid
{
  public double l,b,h;
  public double findVolume( )
  {
     return l*b*h;
  }
}
package P2;
import P1.Cuboid;
class CuboidDemo
{
  main
  public static void main(String[ ] args)
  {
     Cuboid c=new Cuboid();

     c.l=1;
     c.b=2;
     c.h=3;

    System.out.println("Volume="+c.findVolume());
  }
}
```

Accessor and Mutator:

getter — setter.

1. develop a java program cuboid and cuboiddemo, the cuboids contains 3 private members as length, breadth and height. It contains 3 setter methods, the return type is boolean if value are positive. The cuboid class contains 3 getter method as return type is strings, to return the dimensions. To find the volume method is +findVolume return type is ~~void~~ double. to use two string method to print the dimension of Cuboid.

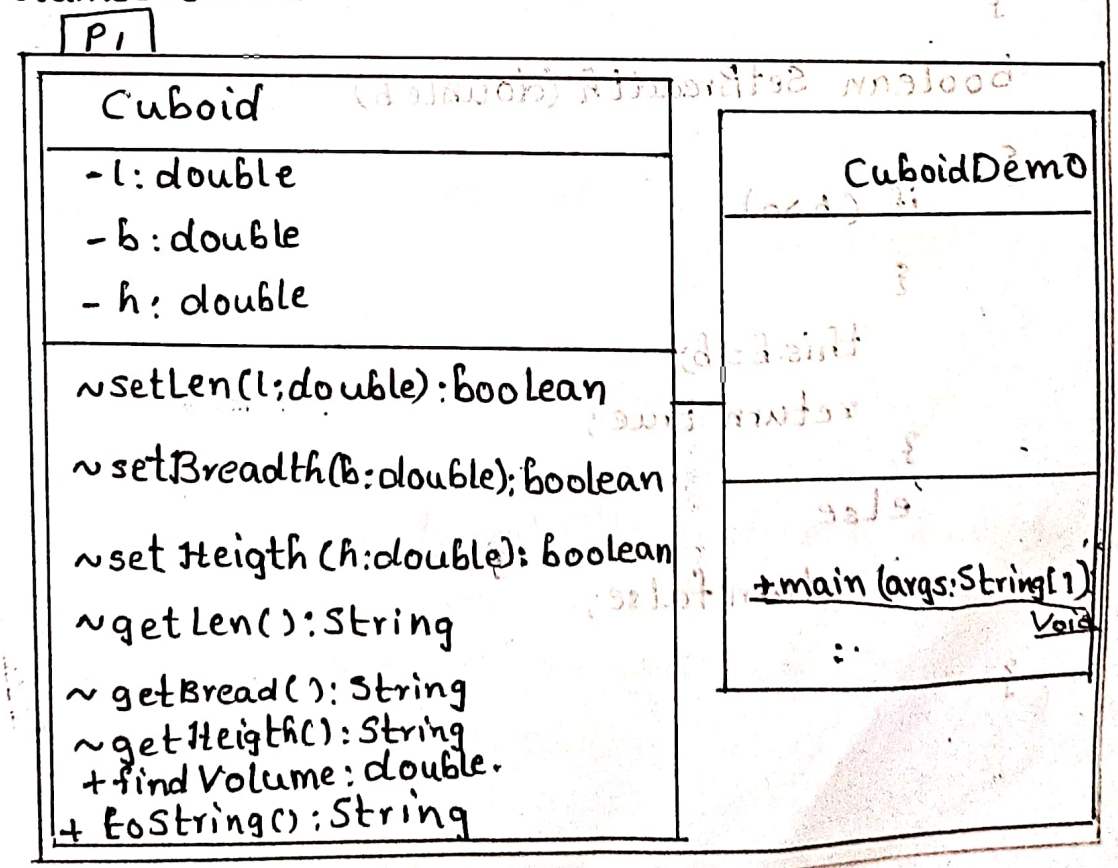
length = 10.0m

Breadth = 12.0m

Height = 15.0M

Volume = 1800CM.

P1



this keyword always refer to the instance of same class. Super keywords refer to parent class.

Program

```
package P1;
class Cuboid
{
   private double l,b,h;
   boolean setLen (double l)
   {
      if (l>0)
      {
         this.l=l;
         return true;
      }
      else
      return false;
   }
   boolean setBreadth (double b)
   {
      if (b>0)
      {
         this.b=b;
         return true;
      }
      else
      return false;
   }
```

```
boolean setHeight(double h)
{
    if (h>0)
    {
        this.h=h;
        return true;
    }
    else
        return false;
}
String getLen( )
{
  return l+ "M";
}
String getBreadth( )
{
   return b+"M";
}
String getHeight( )
{
    return h+"M";
}
public double findVolume( )
{
     return l*b*h;
}
public String toString( )
{
    String s=String.format ("Length=%.s\n Breadth=%s\n
                Height=%s\n Volume=%.f cu.m",
                getLen(),getBreadth().getHeigth(),
                findVolume());
    return s;
}
}
```

```
package P1;
import java.util.Scanner;
class CuboidDemo
{
   public static void main (String args[])
  {
     Scanner sc=new Scanner (System.in);
     Cuboid  c=new Cuboid();
     double l,b,h;
     System.out.println("enter Length");
           l=sc.nextDouble();
      while(!c.setLen(l))
         l=sc.nextDouble();
      System.out.println("enter Breadth");
          b=sc.nextDouble();
       while(!c.setBreadth(b))
          b=sc.nextDouble();
       System.out.println("enter Height ");
          h=sc.nextDouble();
       while(!c.setHeight(h))
          h=sc.nextDouble();
       System.out.println(c);// c.toString()
       sc.Close();
  }
}
```

Command line argument: There are 3 input systems.

1. console 2. file 3. command line

```
package tgan;
public class CmdLineArgs
{
  public static void main(String[] args)
  {
      int i=0;
      for(i=0; i<args.length;i++)
        System.out.println(args[i]);
   }
}
```

1. Write a java program to print sum of numbers using commandLine.

A:

```
package tgan;
public class CmdLineArgs
{
   public static void main(String[] args)
   {
      int i=0,sum=0;
      for(i=0; i<args.length ;i++)
        sum=sum+Integer.parseInt(args[i]);
      System.out.println("Sum="+sum);
   }
}
```

Wrapper class: To convert primitive datatype to wrapper class is called autobox.
if we convert wrapper class to primitive datatype is unboxing. need of wrapper class is only collection.

primitive data type to wrapper

byte - Byte

short - Short

int - Integer

float - Float

double - Double

char - Character

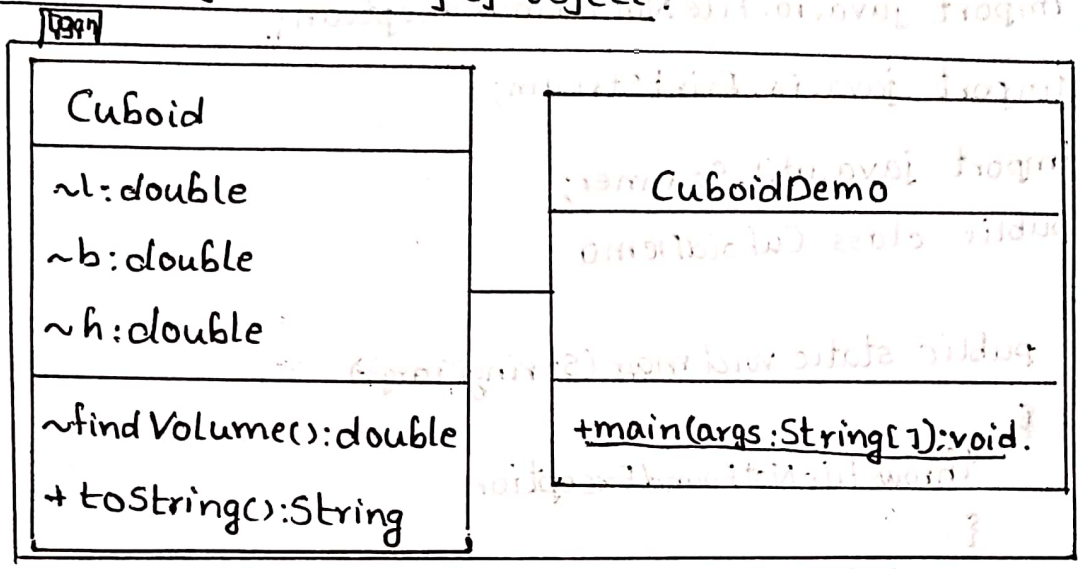
boolean - Boolean.

Program

```
package naveen;
public class CmdLineArgs
{
  public static void main(String[] args)
  {
    int a=12;
    Integer it=a; // autobox.
    System.out.println(a);
    System.out.println(it);
    Double d=12.454;
    double dl=d; // unboxing.
    System.out.println(d);
    System.out.println(dl);
  }
}
```

File input System:

File input system array of object:



1. write a java program to ten object read the input from file and store out in format.

L=10.m B=12.0m H=15.0M Vol=15 5cm

L=12.m B=15M H=12.M vol=186 cu.m.

```
int a[ ]= new int[10];
```

Program

```
package tgan;
public class Cuboid
{
   double l,b,h;
   double findVolume()
   {
      return l*b*h;
   }
   public String toString( )
   {
     return "L= "+ l+"m" +" B = " +b +"m" + ", H = "+h
      "m" + "  Volume = " + findVolume() + "cu.m";
} }
```

```
package tgan;
import java.io.File;
import java.io.FileNotFoundException;
import java.io.PrintStream;
import java.util.Scanner;
public class CuboidDemo
{
  public static void main(String[]args)
  {
     throws FileNotFoundException
     {
         Cuboid c[]=new Cuboid[10];
         File in=new File("D:\\input.txt");
         Scanner sc=new Scanner(in);
         int i=0,j;
         while(sc.hasNext())
         {
           c[i]=new Cuboid();
           c[i].l=sc.nextDouble();
           c[i].b=sc.nextDouble();
           c[i].h=sc.nextDouble();
           i++;
         }
         File ot=new File("D:\\output.txt");
         PrintStreamps=new PrintStream(ot);
         for(j=0;j<i;j++)
         {
             ps.println(c[j]);
             System.out.println(c[j]);
         }
         ps.close();
         sc.close();
     }
}
```

2. KLEF FED office has 2 files, CSE.txt and ECE.txt. Both files have 3 columns - ID, Name and moble number of CSE and ECE students respectively. The task is to read data from both files and store in a file students.txt and print the data of students.txt.

A

```
Package P1;
public class Student
{
    long id, m;
    String name;
    public String toString()
    {
      return "ID: " + id + "\n Name : " + name +
        "\nMobile :" + m;
    }
}
```

```
package esan;
import java.io.File;
import java.io.FileNotFoundException;
import java.io.PrintStream;
import java.util.Scanner;
public class StudentDemo
{
   public static void main(String[] args) throws FileNotFound
   Exception
   {
       Student s[] = new Student[10];
       File cs= new Scanner(cs);
```

```
int i=0;j;
while(sc.hasNext())
{
   s[i]= new Student();
   s[i].id = sc.nextLong();
   s[i].name= sc.next();
   s[i].m = sc.nextLong();
   i++;
}
File output = new File("E:\\s30\\student.txt");
PrintStream ps= new PrintStream(output);
for(j=0; j<i; j++)
{
   System.out.println(s[j]);
   ps.println(s[j]);
}
sc.close();
ps.close();
}
```

Method overload: Class contains same method more than one ~~time~~ format.

It depends no.of.arguments, type of arguments and order of arguments. it doesn't depend on return type.

e.g: int findArea(int a, int b, int c)

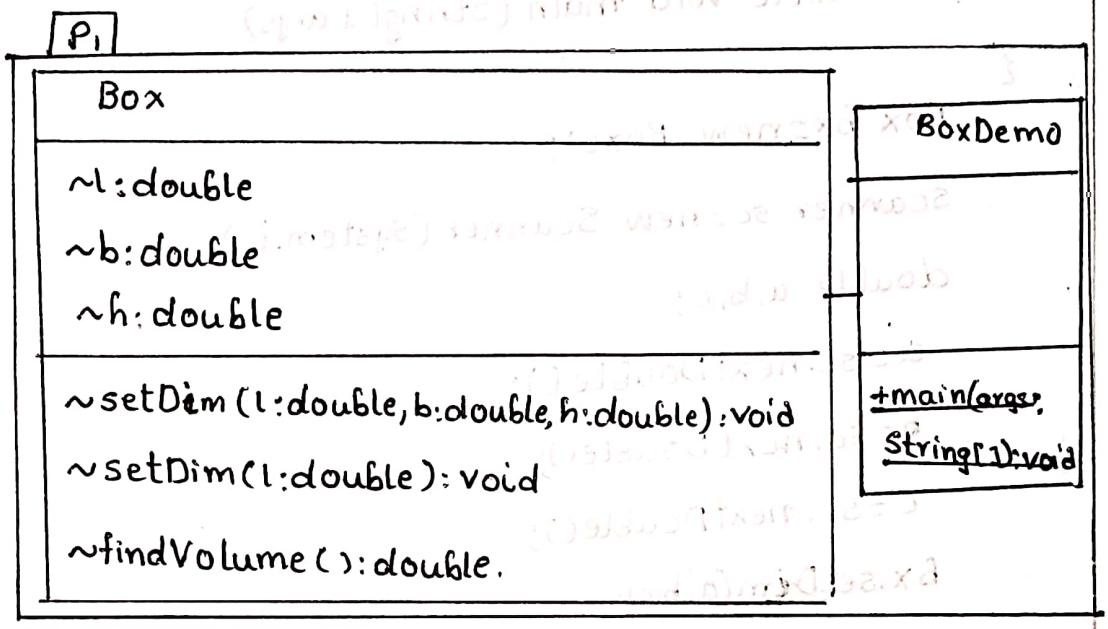
Void findArea(int a, int b) — no.of.arguments.

findArea(int a)

findArea()

int findArea (int a, double b)
int findArea (int a; int b)
} type of arguments.

findArea(int a, double b)
findArea(double a, int b)
} order of arguments.

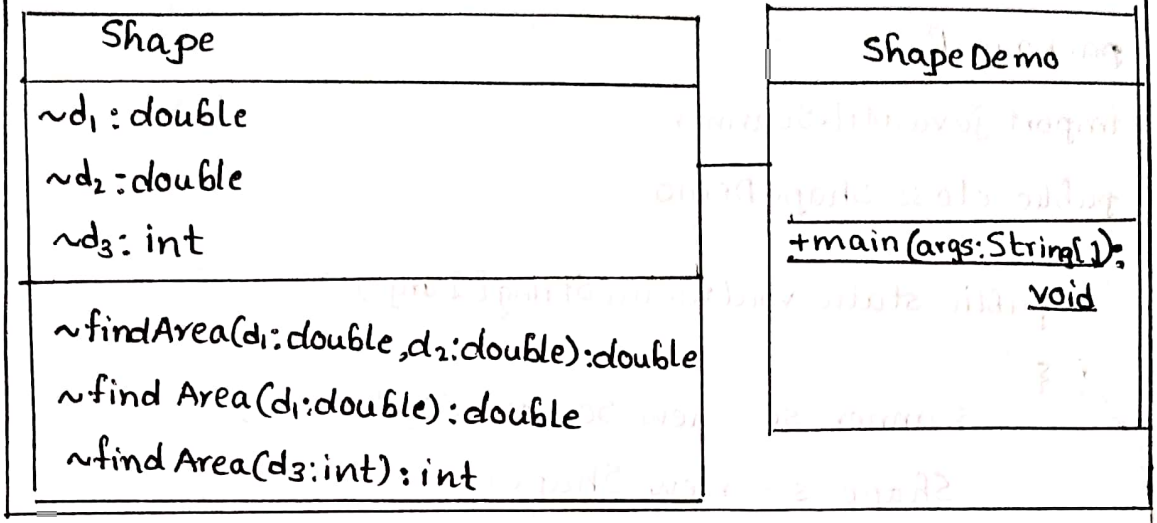



```
package P1;
class Box
{
   double l,b,h;
   void setDim(double l,double b,double h)
   {
        this.l=l;
        this.b=b;
        this.h=h;
   }
   void setDim(double l)
   {
        this.l=l;
          b=l;
          h=l;
    }
   double findVolume()
   {
      return l*b*h;
    }
}
```

```
package P1;
import java.util.Scanner;
class BoxDemo
{
  Public static void main (String[] args)
  {
     Box bx=new Box();
     Scanner sc=new Scanner (System.in);
     double a,b,c;
      a=sc.nextDouble();
      b=sc.nextDouble();
      c=sc.nextDouble();
     bx.setDim(a,b,c);
     System.out.println("Volume = "+bx.findVolume());
     bx.setDim(a);
     System.out.println("Volume="+bx.findVolume());
     sc.close();
  }
}.
```

3. The package P1 contains shape, the class contains 2 default members d_1, d_2 as double, one member as a integer. it contains method findArea() as two argument double. The same area findArease one argument double. same area find Area Argument interger. find the area of rectangle, square, circle. write a democlass to acces the shape.

P1




```
package P1;
public class Shape.
{
    double d1,d2;
     int d3;
    double findArea(double d1,double d2)
    {
       return d1 * d2; —②
        this.d1 =d1; —⑥
        this.d2 =d2; —①
    }
    double findArea(double d1)
    {
        this.d1 = d1.
        return d1 *d1;
    }
     int findArea(int d3)
    {
        this.d3 =d3;
        return 3.14 *d3*d3;
    }
}
```

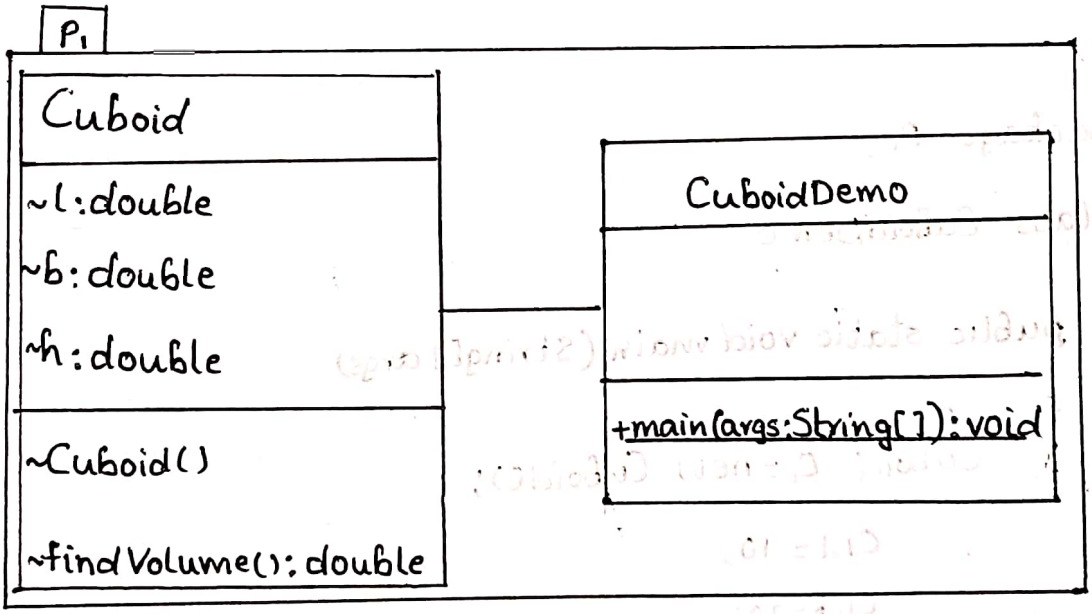
```
package P1;
import java.util.Scanner;
public class ShapeDemo
{
    public static void main(String[] args)
    {
        Scanner sc = new Scanner(System.in);
        Shape s = new Shape();
        System.out.println("Area of rectangle = "+s.findAre-
                    a(sc.nextDouble, sc.nextDouble));
        double a = s.findArea(sc.nextDouble());
        System.out.println("Area of square= "+a);
        System.out.println("Area of circle= "+s.findArea(sc.next
                                                Int());
        sc.close();
    }
}
```

Constructor : same className as Method. It doesn't have any return type. Constructor not be static, abstract and final.

Types of constructor:

1. default constructor.
2. No-argument constructor.
3. Parameterized constructor.

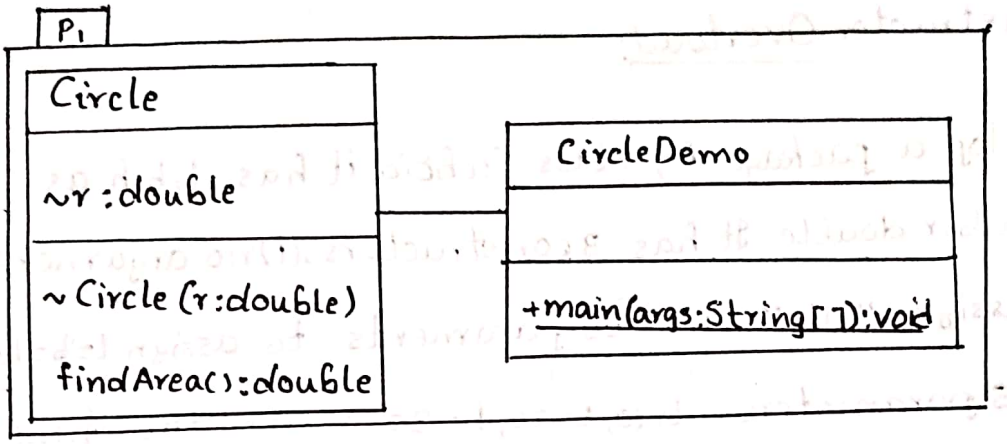
1. Develop a package P1 class Cuboid. it contains 3 parameters members l,b,h to assign default value -1. To use the no argument constructor to initizate default values. It contain findVolume method. CuboidDemo.



```
package P1;

class Cuboid
{
   double l,b,h;

   Cuboid()    // No-argument Constructor.
   {
       l=-1;
       b=-1;
       c=-1;
   }
   double findVolume()
   {
      return l*b*h;
   }
}
```

```
package P1;

class CuboidDemo
{
   public static void main (String[] args)
   {
      Cuboid C1 = new Cuboid();
            C1.l = 10;
            C1.b=20;
            C1.h=30;

      System.out.println("Volume = "+C1.findVolume());
   }
}
```

2.

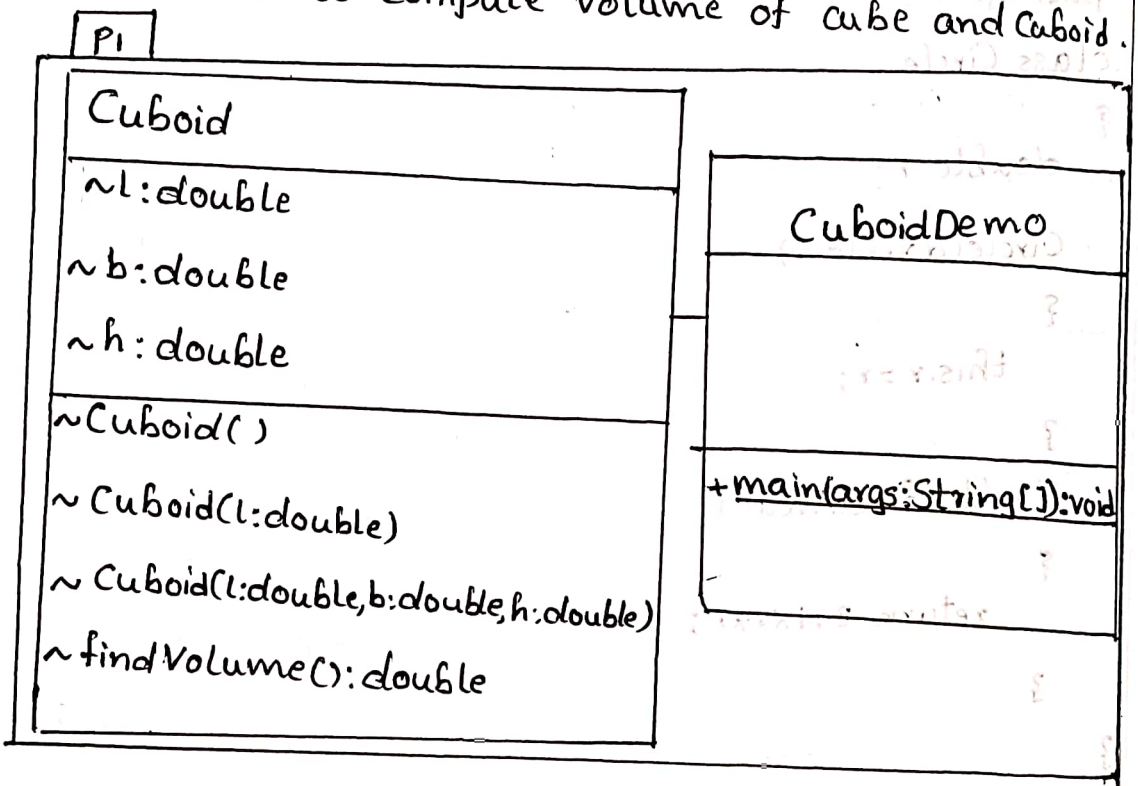



```
package P1;
class Circle
{
   double r;
   Circle(double r)
   {
      this.r=r;
   }
   double findArea()
   {
      return 3.14*r*r;
   }
}
package P1;
class CircleDemo
{
   public static void main(String args[])
   {
      Circle c= new Circle(10);
      System.out.println("Area="+c.findArea());
      c.r=21;
      System.out.println("Area="+c.findArea());
   }
}
```

Constructor Overload:

1. Develop a package P1, class Cuboid. it has l, b, h as member double. It has 3 constructors. (i) no argument to assign 10, 11, 12. (ii) ~~two~~ one parament to assign l=b=h=10; (iii) 3 parameters l=10, b=20, h=30, write the find value method to compute volume of cube and Cuboid.

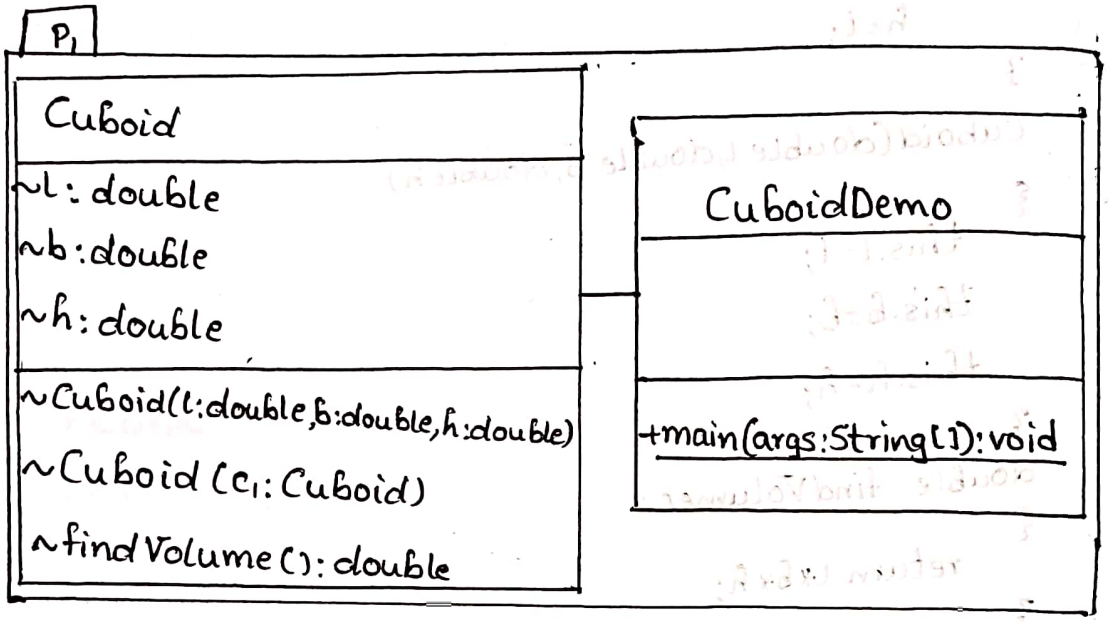
Ans:




```
package P1;
class Cuboid
{
   double l,b,h;
   Cuboid()
   {
        l=10;
        b=11;
        h=12;
   }
```

```
    Cuboid(double l)
    {
        this.l = l;
            b=l;
            h=l;
    }
    Cuboid(double l,double b, double h)
    {
        this.l=l;
        this.b=b;
        this.h=h;
    }
    double findVolume()
    {
        return l*b*h;
    }
}
package P1;
class CuboidDemo
{
  public static void main(String[] args)
  {
     Cuboid c1 = new Cuboid();
     Cuboid c2 = new Cuboid(10);
     Cuboid c3 = new Cuboid(10,20,30);
     System.out.println("volume = " + c1.findVolume);
     System.out.println("Volume=" + c2.findVolume);
     system.out.println("Volume =" + c3.findVolume);
     sc.close();
  }
}
```

Copy Constructor; To pass the object as the argument in constructor is called Copy constructor.




```
package P1;
class Cuboid
{
   double l,b,h;
   Cuboid(double l,double b,double h)
   {
       this.l=l;
       this.b=b;
       this.h=h;
   }
   Cuboid(Cuboid c1)
   {
       l=c1.l;
       b=c1.b;
       h=c1.h;
   }
```

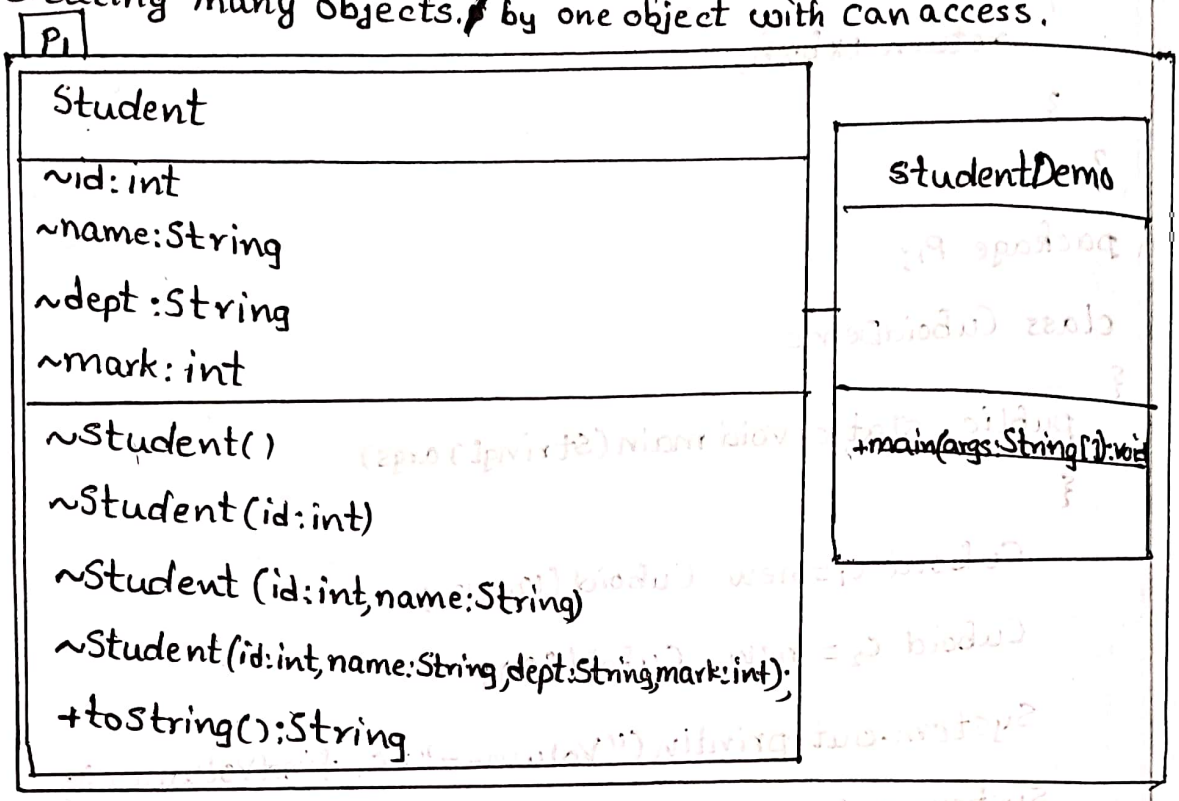
```
    double findVolume()
    {
        return l*b*h;
    }
};
```

```
package P1;
class CuboidDemo
{
    public static void main(String[] args)
    {
        Cuboid c1 = new Cuboid(10,11,12);
        Cuboid c2 = new Cuboid(c1);
        System.out.println("Volume=" + c1.findVolume());
        System.out.println("Volume = " + c2.findVolume());
            c2.l=16;
        System.out.println("Volume=" + c1.findVolume());
        System.out.println("Volume = " + c2.findVolume());
    }
}
```

this();

it can be used to access same constructor.

Constructor Chain: It is used to access constructors without creating many objects. by one object with can access.

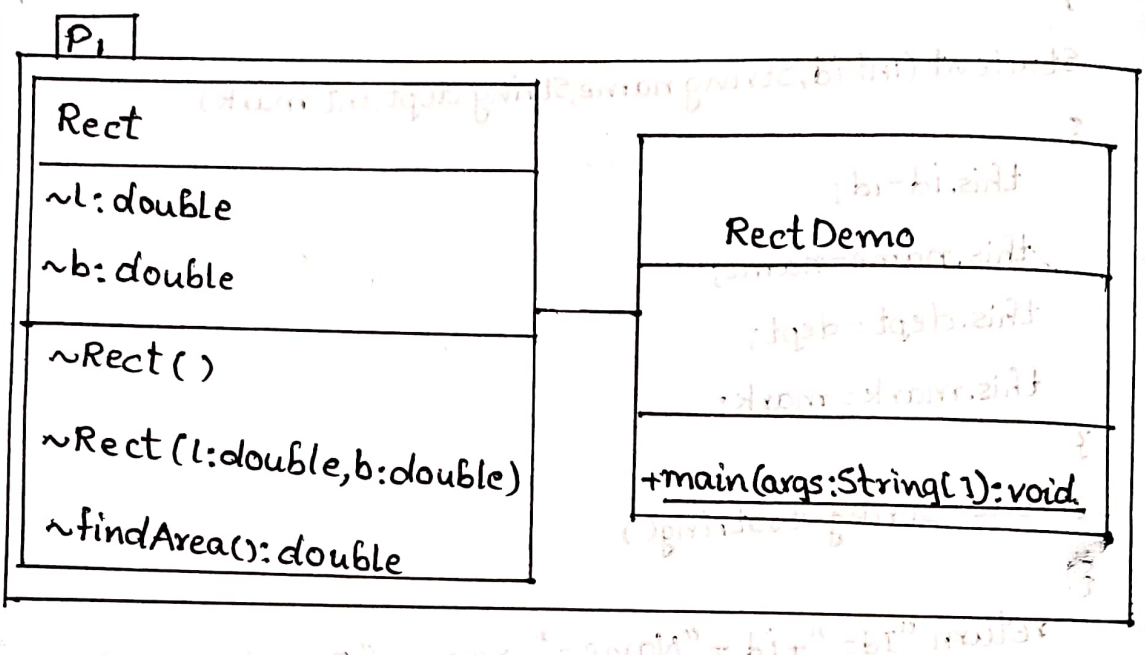



```
package P1;
class Student
{
   int id;
   String name;
   String dept;
   int mark;
   Student( )
   {
     this(200031509);
   }
   Student (int id)
   {
      this (id, "ravi");
  }
```

```
    Student(int id, String name)
    {
        this(id,name,"CSE",70);
    }
    Student(int id,String name,String dept,int mark)
    {
        this.id=id;
        this.name=name;
        this.dept=dept;
        this.mark=mark;
    }
    public String toString()
    {
        return "Id= "+id+"Name= "+name+"Department= "+dept+
        "Mark= "+mark;
    }
}
package P1;
class StudentDemo
{
    public static void main(String[]args)
    {
        Student s=new Student();
        System.out.println(s);
    }
}
```

2. To call the no argument constructor to initilize default rectangle size is 10,20. To call findArea() method to display the output.

A:



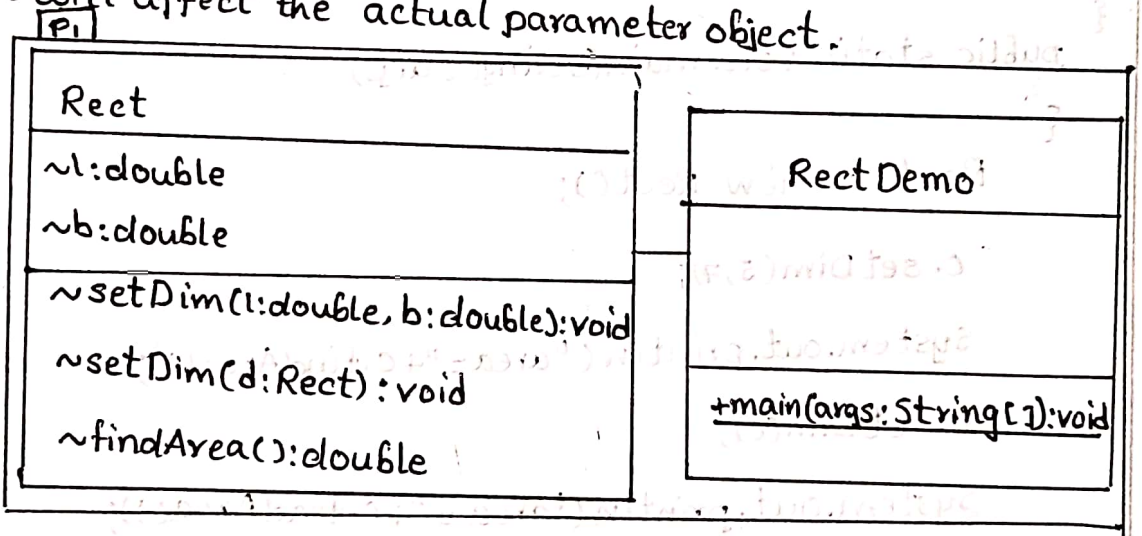

```
package P1;
class Rect
{
  double l,b;
  Rect()
  {
    this(10,20);
  }
  Rect(double l,double b)
  {
    this.l=l;
    this.b=b;
  }
  double findArea()
  {
    return l*b;
  }
}
```

```
package P1;
class RectDemo
{ public static void main(String[]arg){
   Rect r=new Rect();
   System.out.println(r);
  }
}
```

Call by value and Call by reference:

Call by value: Any changes in format parameter it does not effect the actual parameter.

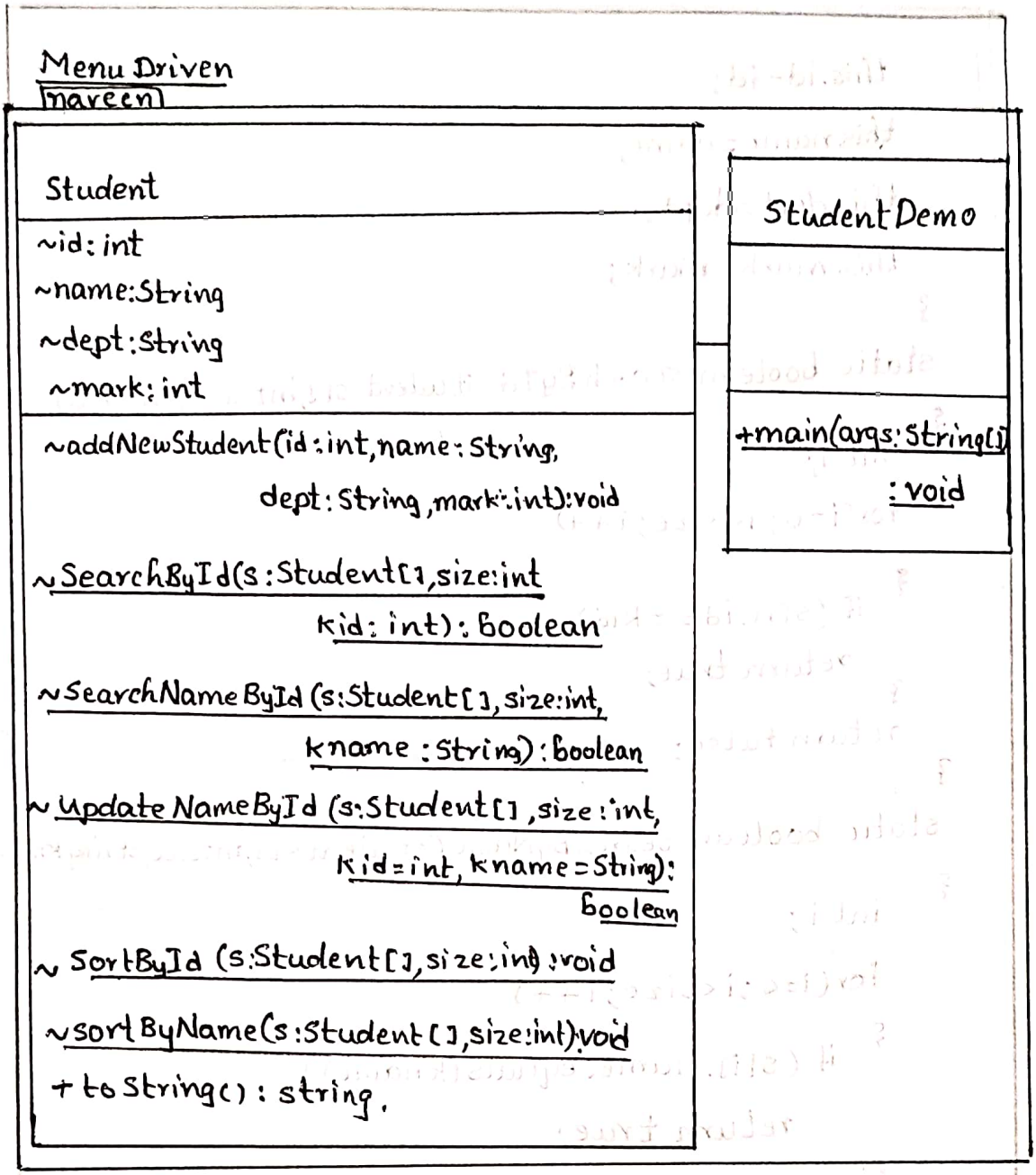
Call by reference: Any changes in formal parameter object it will affect the actual parameter object.




```
package P1;
class Rect
{
   double l,b;
   void setDim(double l,double b)
   {
       this.l=l;
       this.b=b;
   }
```

```
    void setDim(Rect d)
    {
        d.l=20;
        d.b=40;
    }
    double findArea()
    {
      return l*b;
    }
}
package P1;
class RectDemo
{
  public static void main(String[] args)
  {
      Rect c=new Rect();
       c.setDim(5,7);
       System.out.println("area="+c.findArea());
       c.setDim(c);
       System.out.println("area="+c.findArea());
  }
}
```

Menu Driven
(naveen)




```
package naveen;
public class Student
{
    int id;
    String name;
    String dept;
    int mark;
    void addNewStudent(int id, String name, String
     dept, int mark)
    {
```

```
    this.id=id;
    this.name=name;
    this.dept=dept;
    this.mark=mark;
}
static boolean searchById(Student s[],int size,int kid)
{
    int i;
    for(i=0; i<size;i++)
    {
        if (s[i].id==kid)
        return true;
    }
    return false;
}
static boolean searchByName(Student s[],intsize,String kname)
{
    int i;
    for(i=0;i<size;i++)
    {
        if (s[i].name.equals(kname))
            return true;
    }
    return false;
}
static boolean updateNameById(Student s[],int size,
        int kid,string kname)
{
    int i;
    for (i=0;i<size;i++)
    {
        if (s[i].id==kid)
        {
            s[i].name=kname;
        return true;
```

```
}
 return false;
}
static void sortById(student s[],int size)
{
    int i,p;
    student temp;
    for(p=1;p<size;p++)
    {
      for(i=0;i<size-p;i++)
      {
        if (s[i].id > s[i+1].id)
        {
            temp=s[i];
            s[i]= s[i+1];
            s[i+1]= temp;
        }
      }
    }
}
static void sortByName (Student s[],int size)
{
    int i, p;
    student temp;
    for(p=1; p<size;p++)
    {
      for(i=0; i<size-p;i++)
      {
        if(s[i].name.compareTo (s[i+1].name)>0)
        {
            temp=s[i];
            s[i]=s[i+1];
            s[i+1]= temp;
      } } }
}
public string toString(){
 return "Id="+ id +"name="+ name + "dept "+ depd +
    "mark" + mark +"\n";
} }
```

```
package naveen;
import java.util.Scanner;
public class StudentDemo
{
  public static void main(string[] args)
  {
    student s[]=new student[100];
    scanner sc=new Scanner(System.in);
    boolean repeat=true;
    int id,mark,size=0,i;
    string name,dept;
    while(repeat)
    {
      system.out.println("1.addNewStudent \n 2.search
      ById\n 3.searchByName\n 4.updateNameById \n 5.
      sortById\n 6.sortByName\n 7.display\n other exit");
      int option=sc.nextInt();
      switch(option)
      {
        case 1:
          s      s[size]=new Student();
                 id =sc.nextInt();
                 name=sc.next();
                 dept=sc. next();
                 mark=sc.nextInt();
                 s[size].addNewStudent(id,name, dept,mark);
                 size++;
                 break;
        case 2:
                 id=sc.nextInt();
                 if(student.search ById (s, size, id))
                    system.out.println("record found");
```

```
    else
      system.out.println("record not found");
    break;
case 3:
    name=sc.next();
    if(Student.searchByName(s,size,name))
       System.out.println("record found");
    else
       system.out.println("record not found");
    Break;
case4:
      id=sc.nextInt();
      name=sc.next();
      if(Student.updateNameById(s,size,id,name))
        system.out.println("Updated sucessfully");
      else
        system.out.println("not updated");
      break;
case 5:
      Student.sortById(s,size);
      break;
case 6:
      Student.sortByName(s,size);
      break;
case 7:
      Student.sortBy
      for(i=0;i<size;i++)
          System.out.println(s[i]);
        break;
 default:
      repeat=false;
  }
}.
 sc.close();
}
}
```

String Tokenizer:

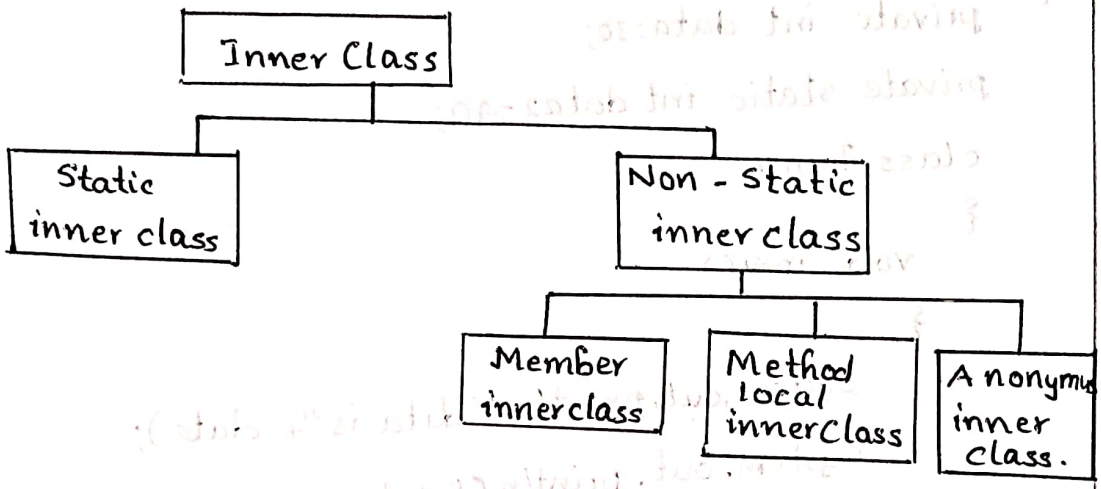
String s= " welcome to all";

StringTokenizer st=new StringTokenizer(s);→nextToken(" ");

StringTokenizer st=new StringTokenizer(s, "!") ;→nextToken();

```
package naveen;
import java.util.StringTokenizer;
public class StringTokenDemo
{
   public static void main(String[] args)
   {
      String s= "welcome to all";
      StringTokenizer st= new StringTokenizer(s);
      int c=st.countTokens();
     System.out.println("count= "+c);
     while(st.hasMoreTokens())
      System.out.println(st.nextToken());
   }
}
```

Inner Class: A class inside another class is called inner class.




```
class Outer
{
  public static int d1=30;
  public int d2=40;
  public static class Inner
  {
     void msg()
     {
        Outer o=new Outer();
        System.out.println(d1);
        System.out.println(d2);
     }
  }
  public static void main(String[] args)
  {
     Outer.Inner i = new OuterInner();
     i.msg();
  }
}
```

```
package innerClasses;
public class TestMemberOuter1
{
    private int data=30;
    private static int data2=40;
    class Inner
    {
        void msg()
        {
            System.out.println("data is"+ data);
            System.out.println("data2:"+data2);
        }
    }
    public static void main(String[] args)
    {
        TestMember outer1 obj=new TestMemberOuter();
        TestMemberOuter1.Inner in=obj.new Inner();
        in.msg();
    }
}
```

Method local class:

Defining a class inside a method is called Method local class.

```
package innerClass;
public class localInner1
{
    private int data=30;
    void display()
    {
        class local1
        {
```

```
        int a=20;
        void msg()
        {
          System.out.println(data);
          System.out.println(a);
        }
    }
    local l= new local();
    l.msg();
}
public static void main(String[] args)
{
    local Inner obj = new localInner();
    obj.display();
}
}
```

2.
```
package naveen;
class Outer
{
  static int temp1 = 1, temp2= 2;
  int temp3= 3, temp4= 4;
  public static class Inner
  {
    private static int temp5=5;
    Private static int getsum()
    {
      Outer o= new outer();
      return (temp1+temp2+ o.temp3 + o.temp4 +temp5);
    }
  }
  public static void main(string[] args)
  {
    Outer.Inner obj = new outerInner();
  } } System.out.println(obj.getsum());
```

String Class

== 3 obj memory

equals() → compare content

compareTo → ascii Diff.

Calendar class

*1. Write a program that reads a paraph of text from file. print the analytics as per the below format:

Line# Number of words Data

Total Count of words.

A.
```
package naveen;
import java.io.File;
import java.io.FileNotFoundException;
import java.util.Scanner;
import java.util.StringTokenizer;
public class TextCount
{
   public static void main(String[] args) throws FileNotFound
   Exception)
   {
      File in=new File("D:\\input.txt");
      scanner sc=new Scanner(inp);
      int total=0,line=0;
      while(sc.hasNextLine())
      {
        StringTokenizer st=new StringTokenizer(sc.nextLine());
        Line++;
        total+=st.countTokens();
        System.out.print("Line "+line+" word count="+
        st.countTokens()+" text is: ");
```

```
        while (st.hasMoreTokens())
        {
            System.out.print(st.nextToken()+" ");
        }
        System.out.println(" ");
    }
    System.out.println("total= "+ total);
    sc.close();
  }
}
```

2. Calender.java

```
package naveen;
import java.util.Calendar;
public class Calender
{
    public static void main(String[] args)
    {
        Calendar c= Calendar.getInstance();
        System.out.println(c.getTime());
        c.add(Calendar.MONTH,-5);
        System.out.println(c.getTime());
        c.add(Calendar.YEAR,-6);
        System.out.println(c.getTime());
        c.add(Calendar.DATE,23);
        System.out.println(c.getTime());
        c.set(2002,8,12);
        System.out.println(c.getTime());
        System.out.println(c.get(Calendar.YEAR));
    }
}
```

```
Tue Mar 30 04:16:28 IST 2021
Fri Oct 30 04:16:28 IST 2020
Thu Oct 30 04:16:28 IST 2014
Sat Nov 22 04:16:28 IST 2014
THU Sep 12 04:16:28 IST 2002
2002
```

3. StringBuffBuild.java

```
package naveen;
public class StringBuffBuild
{
   public static void main(String[] args)
   {
     StringBuffer sb=new StringBuffer("welcome");
     System.out.println(sb);
     sb.append(" to all");
     System.out.println(sb);
    sb.insert(2,"KLEF");
    System.out.println(sb);
    sb.replace(3,6,"java");
    System.out.println(sb);
    sb.reverse();
     System.out.println(sb);
     sb.delete(2,6);
     System.out.println(sb);
     System.out.println(sb.charAt(5));
     System.out.println(sb.length());
     System.out.println(sb.substring(4));
     System.out.println(sb.substring(4,9));
   }
}
```

welcome
welcome to all
weKLEFcome to all
weKJavalcome to all
lla ot emoclavaJkew
o
is
moclavaJkew
mocla.

4. StringTokenDemo.java

```
package naveen;
imopt
import java.util.StringTokenizer;
public class StringTokenDemo
{
  public static void main(String[] args)
  {
      String s = "welcome!to!all";
      StringTokenizer st=new StringTokenizer(s);
      int c=st.countTokens();
      System.out.println ("count ="+c);
      while (st.hasMoreTokens()).
        System.out.println(st.nextToken("!");
  }
}
```

```
count=1
welcome
to
all
```

5. StringDemo.java

```
package naveen;
public class StringDemo
{
  public static void main(String[] args)
  {
      String s1 = "welcome";
      String s2 = "welcome";
      String s3 = new String("welcome");
      if (s1==s2)
        System.out.println ("true");
      else
        System.out.println ("false");
```

```
    if(s1==s3)
      System.out.println ("false");
    else
      System.out.println ("true");

    System.out.println("s1 equals s2 :"+s1.equals(s2));
    System.out.println ("s1 equals s3 :"+s1.equals (s3));
    String s4="abc";
    String s5= "bac";
    System.out.println("s4 campareTo s5: "+s4.compareTo(s5));
  }
}
```

Output:

```
true
false
s1 equals s2:true
s1 equals s3: true
s4 compare to s5 :-1
```

Output of StringHanding.java: (next page)

```
C
7
come
lco
true
false
welcome to all
true
welcome
WELCOME
welcome  KLEFall
1
```

StringHanding.java

```
package naveen;
public class StringHanding
{
   public static void main(String[] args)
   {
      String s1="welcome";
      String s2=" to all";
      String s4="welcome";
      String s3="WELCOME";
      System.out.println(s1.charAt(3));
      System.out.println(s1.length());
      System.out.println(s1.substring(3));
      System.out.println(s1.substring(2,5));
      System.out.println(s1.contains("com"));
      System.out.println(s1.isEmpty());
      s1=s1.concat(s2); //s1=s1+s2
      System.out.println(s1);
      System.out.println(s4.equalsIgnoreCase(s3));
      s3=s3.toLowerCase();
      System.out.println(s3);
      s4=s4.toUpperCase();
      System.out.println(s4);
      s1=s1.replace("to","KLEF");
      System.out.println(s1);
      System.out.println(s1.indexOf("e1"));
   }
}
```

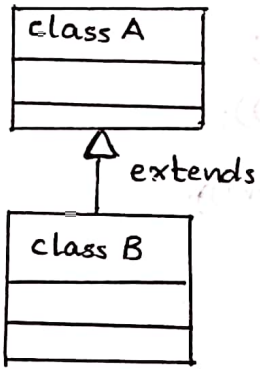
CO3 and CO4

Inheritance: It is a relation between parent and child. class can be defined from another class. we can use the keyword extends for inheritance. symbol $\xrightarrow{\text{E}}_{\text{P}}$

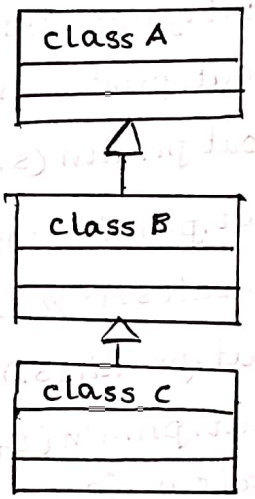
Types of inheritance:

1. Simple inheritance
2. Multilevel inheritance
3. Hierarchy inheritance
4. multiple inheritance (interface)
5. Hybrid inheritance.

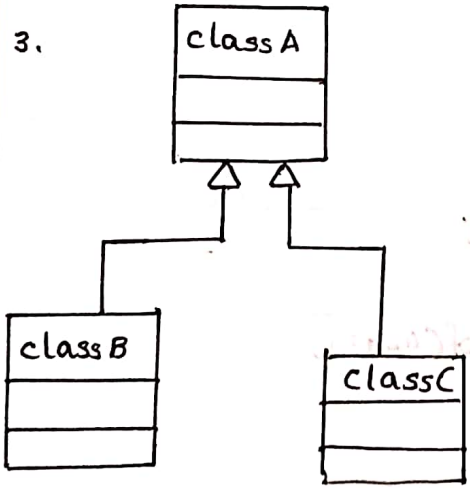
1.



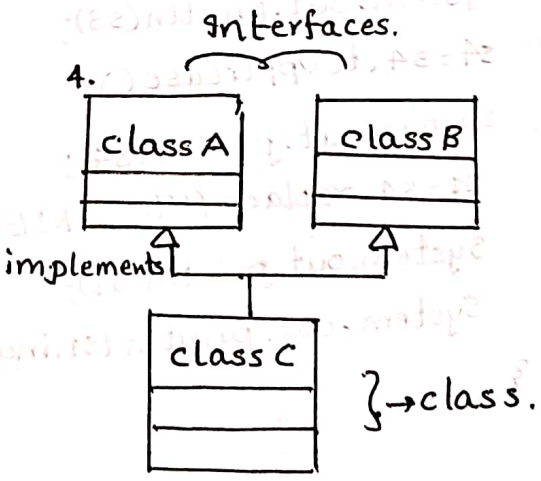
2.



3.



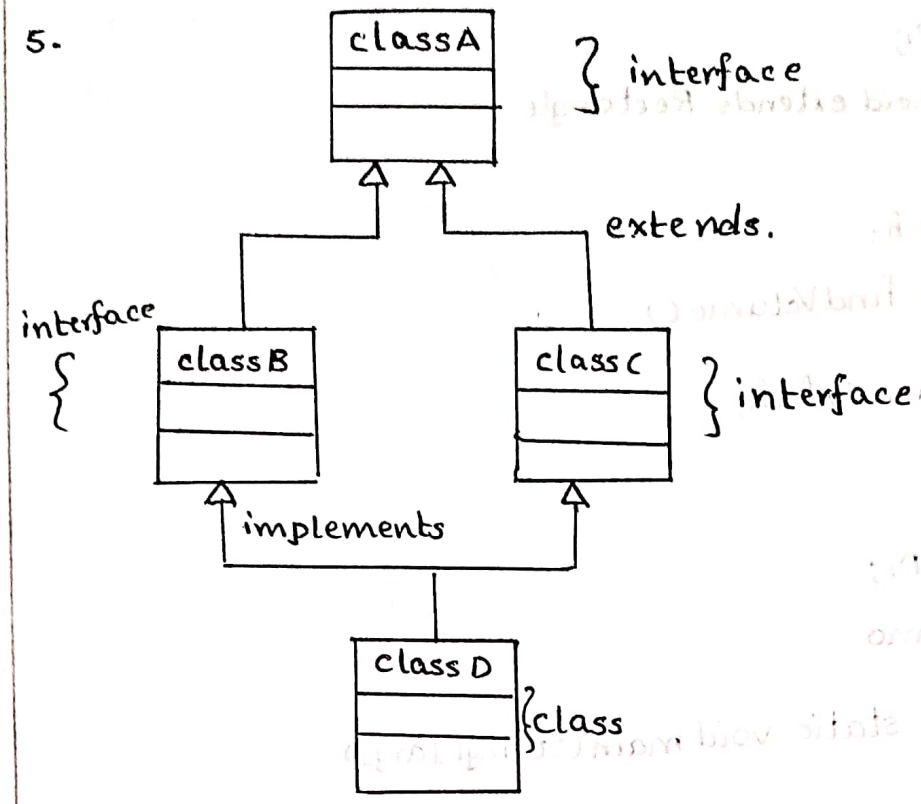
4.



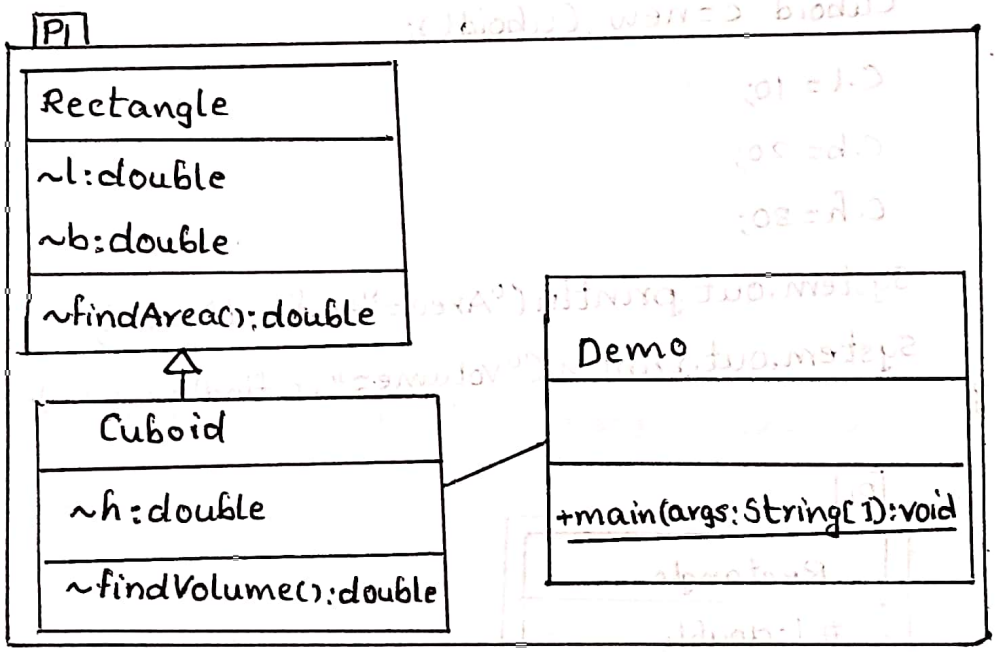
Note: *If we use implement, then it is interfaces

* class extends to another class only.

5.



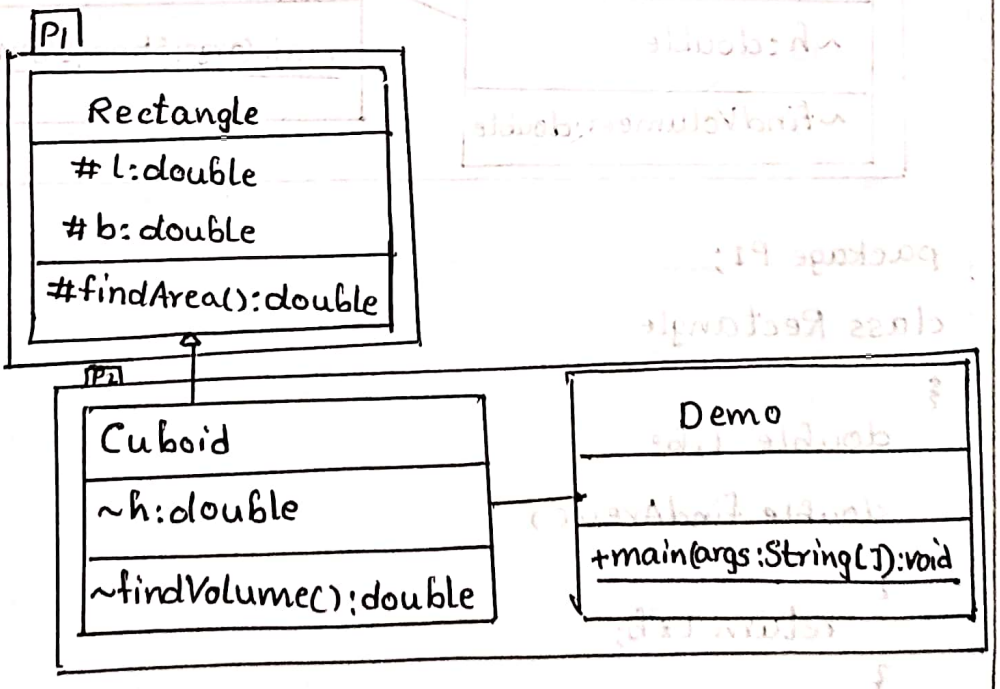
1.




```
package P1;
class Rectangle
{
   double l,b;
   double findAreac()
   {
      return l*b;
   }
}
```

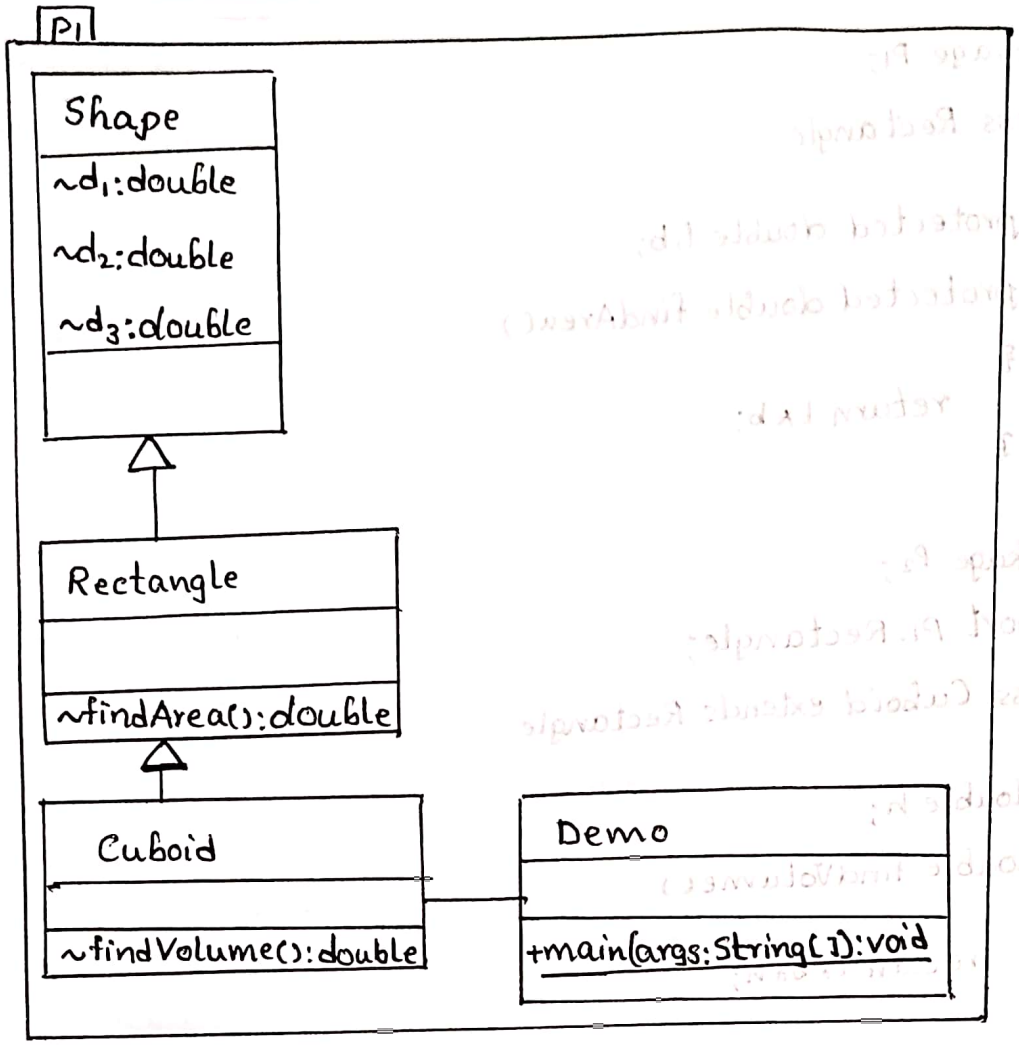
```
package P1;
class Cuboid extends Rectangle
{
    double h;
    double findVolume()
    {
      return l*b*h;
    }
}
package P1;
class Demo
{
    public static void main(String[]args)
    {
        Cuboid c=new Cuboid();
        c.l= 10;
        c.b= 20;
        c.h=30;
      System.out.println("Area="+c.findArea());
      System.out.println("Volume="+c.findVolume());
    }
}
```

2.



```
package P1;
class Rectangle
{
   protected double l,b;
   protected double findArea()
   {
      return l*b;
   }
}
package P2;
import P1.Rectangle;
class Cuboid extends Rectangle
{
   double h;
   double findVolume()
   {
      return l*b*h;
   }
}
package P2;
class Demo
{
  public static void main(String[] args)
  {
     Cuboid c=new Cuboid();
        c.l=10;
        c.b=20;
        c.h=30;
     System.out.println("Area="+c.findArea());
     System.out.println("Volume="+c.findVolume());
  }
}
```

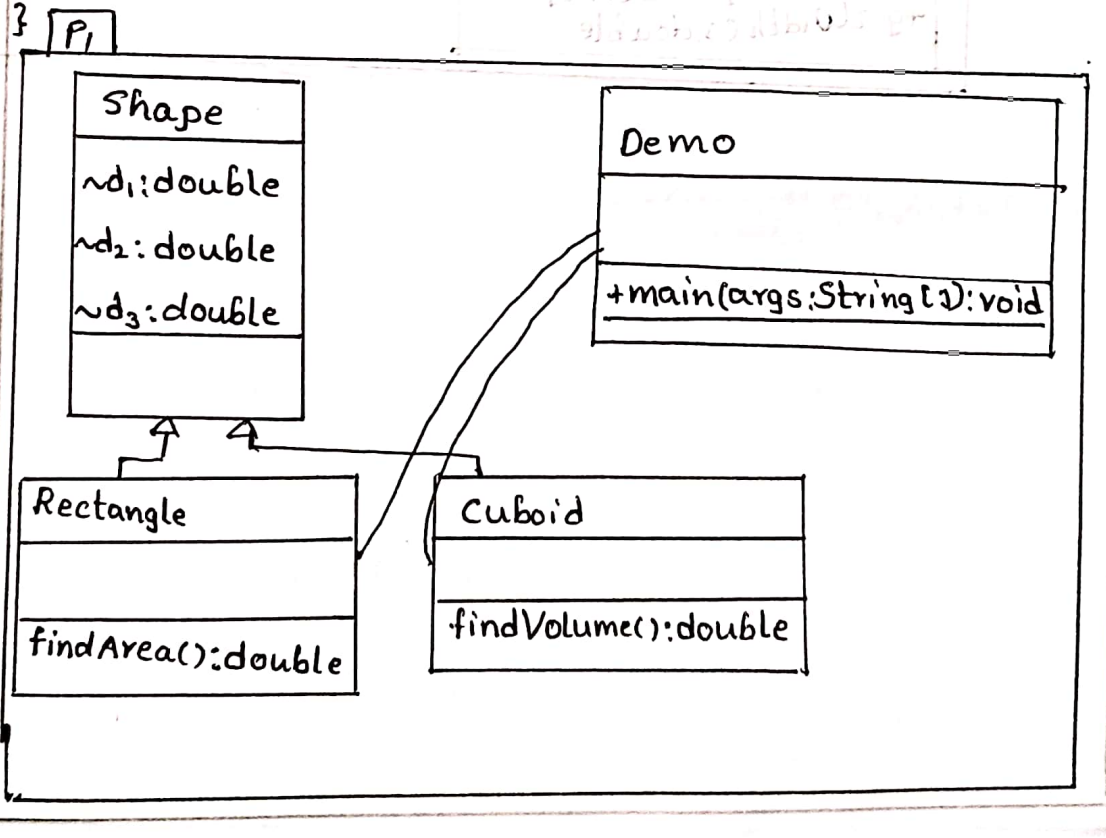
3.

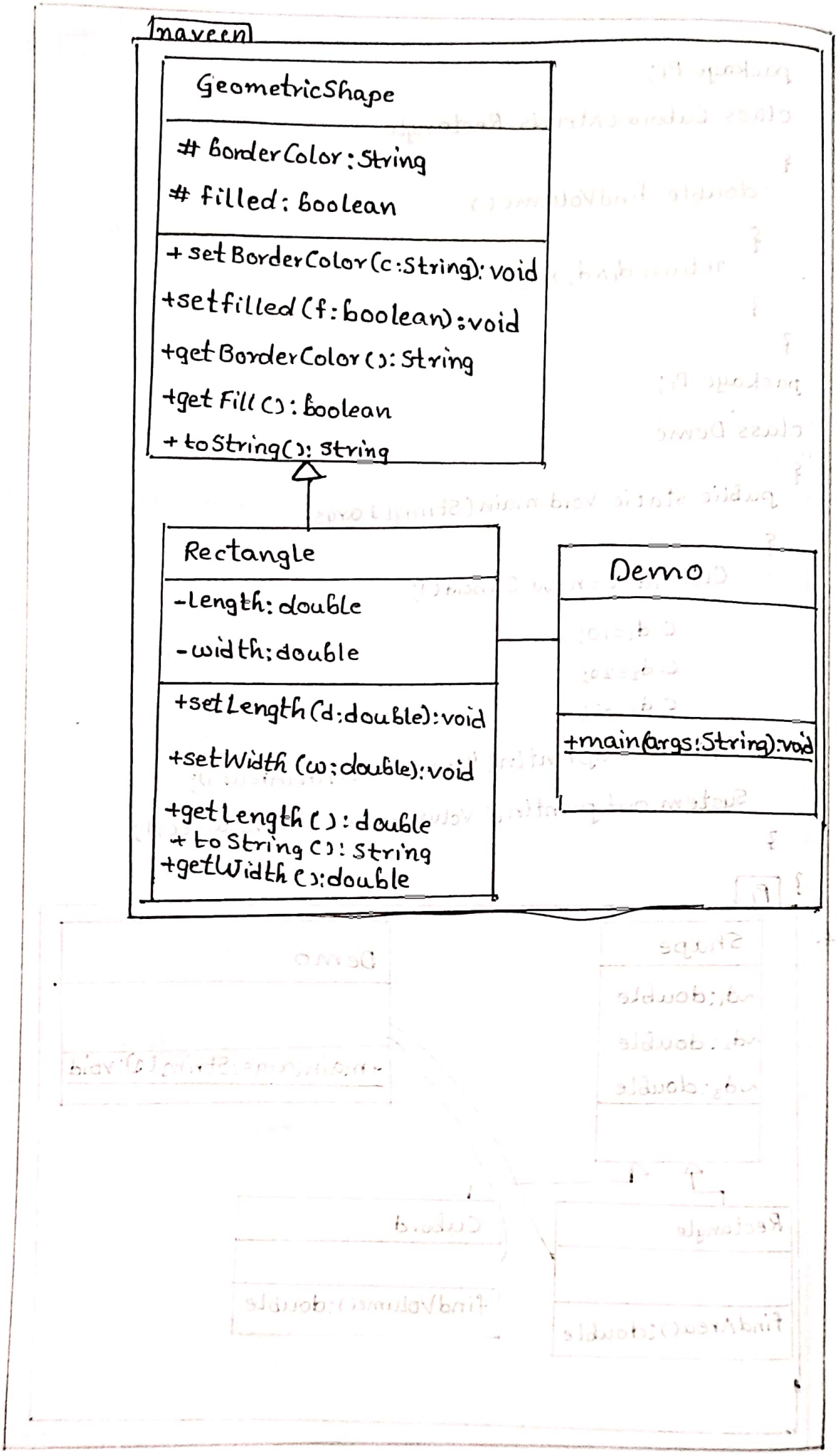



```
package P1;
class Shape
{
  double d1,d2,d3;
}
package P1;
class Rectangle extends Shape
{
   double findArea()
   {
      return d1 x d2;
   }
}
```

```
package P1;
class Cuboid extends Rectangle
{
   double findVolume()
   {
      return d1 x d2 x d3;
   }
}
package P1;
class Demo
{
  public static void main(String[] args)
  {
      Cuboid c=new Cuboid();
          c.d1=10;
          c.d2=20;
          c.d3=30;
     System.out.println("Area="+c.findArea());
     System.out.println("Volume="+c.findVolume());
  }
}
```

4.





```
package naveen;
Public class GeometricShape
{
   protected String borderColor;
   protected boolean filled;
   public String getBorderColor()
   {
      return borderColor;
   }
   public void setBorderColor(String borderColor)
   {
      this.borderColor= borderColor;
   }
   public boolean getFilled()
    {
       return filled;
    }
    public void setFilled(boolean filled)
    {
        this.filled= filled;
    }
    public String toString(){
       return "borderColor="+ borderColor +", Filled=" +filled;
    }
}
 package naveen;
 public class Rectangle extends GeometricShape
 {
    private length,width;
    public double getLength()
    {
        return length;
    }
```

```
    public void setLength(double length)
    {
        this.length = length;
    }
    public double getWidth()
    {
        return width;
    }
    public void setWidth(double width)
    {
        this.width= width;
    }
    public string toString(){
       return "Length ="+length+",width="+width+"\n"+super.
                                                    toString();
} }
package naveen;
public class Demo
{
   public static void main(String[] args)
   {
       Rectangle r1 = new Rectangle();
       r1.borderColor = "Red";
       r1.filled = true;
       r1.setLength(10);
       r1.setWidth(20);
       System.out.println(r1);
   }
}
```

```
package naveen;

public class Cuboid extends Rectangle
{
    private double height;
    public double getHeight()
    {
        return height;
    }
    public void setHeight(double height)
    {
        this.height = height;
    }
    public String toString()
    {
        return "height =" + height + "\n" + super.toString();
    }
    public static void main(String[] args)
    {
        Cuboid c1 = new Cuboid();
        c1.setBoredColor("Blue");
        c1.setFilled(False);
        c1.setLength(10);
        c1.setWidth(20);
        c1.setHeight(30);
        System.out.println(c1);
    }
}
```

```
package naveen;
public class Circle extends GeometricShape
{
    private double radius;
    public double getRadius()
    {
       return radius;
    }
    public void setRadius(double radius)
    {
        this.radius= radius;
    }
    public String toString()
    {
       return "radius=" +radius+"\n"+ super.toString();
    }

    public static void main(String[] args)
    {
        Circle c1=new Circle();
        c1.radius =10;
       c1.setBorder Color ("Green");
       c1.set Filled (false);
       System.out.println(c1);
    }
}
```

Protected

It is a access specifier which allow access within same package and also subclasses of other package. sub classes means child class.

	Private	defaut	protected	Public.
sameclass	Yes	Yes	Yes	Yes
samepackage subclass	No	Yes	Yes	Yes
samepackage Non-Subclass	No	Yes	Yes	Yes
Other package subclass	No	No	Yes	Yes
Other package Non-subclass	No	No	No	Yes

1.
```
package naveen;

Public class GeomtricShape
{
    protected String borderColor;
    protected boolean filled;
    - - - - -
    - - - - -
    - - - - - .
}
package P1;
import naveen.GeometricShape;
public class Rectangle extends GeometricShape{
    private L,b,h;
    . . . . . .
    public static void main(String[] args)
    {
        Rectangle r1 = new Rectangle();
        r1.borderColor = "Red";
```

```
        r1.filled = true;
        r1.setLength(10);
        r1.setWidth(20);
        System.out.println(r1);
    }
}
```

Other package nonsubclass.

2.
```
package p1;
import naveen.Rectangle;
public class Demo
{
    public static st void main(String[]args)
    {
        Rectangle r1 = new Rectangle();
        // r1.borderColor = "Red" (we can't access).
        r1.setBorderColor("Red");
        r1.setFilled(true);
        r1.setLength(10);
        r1.setWidth(20);
        System.out.println(r1);
    }
}
```

Same package non-subclass

3.
```
package naveen;
public class Demo
{
    public static void main(String[] args)
    {
        Rectangle r1 = new Rectangle();
        r1.borderColor = "Red";
        r1.filled = true;
```

```
        r1.setLength(10);
        r1.setWidth(20);
        System.out.println(r1);
    }
}
```

Super keyword

1. To differentiate parent class variable & Childclass variable.
2. To differentiate parent class methods & childclass methods.
3. To Call parent class constructor from a child class constructor

1. example for 1:

```
public class Test
{
   public String color="Red";
}
public class Test2 extends Test
{
    public String color="Yellow";
    public void display()
    {
      System.out.println(color);
      System.out.println(super.color);
    }
    public static void main(String[]args)
    {
       Test2 t = new Test2();
         t.display();
    }
}
```

2. Example for 2:

Note: we can't call obj.super.display() [X].

```
package naveen;
public class Test
{
    public void display()
    {
        System.out.println("Parent display()");
    }
}
package naveen;
public class Test2 extends Test
{
    public String color = "yellow";
    public void display()
    {
        System.out.println("Child display");
    }
    public void run()
    {
        display();
        super.display();
    }
    public static void main(String[] args)
    {
        Test2 t = new Test2();
        t.run();
    }
}
```

3 Example for 3:

```
package naveen;
public class TestConst
{
    int a;
    TestConst()
    {
        system.out.println("Parent No arg Con");
         a=5;
    }
    TestConst(int a)
    {
        System.out.println("Parent arg Const");
        this.a=a;
    }
}
```

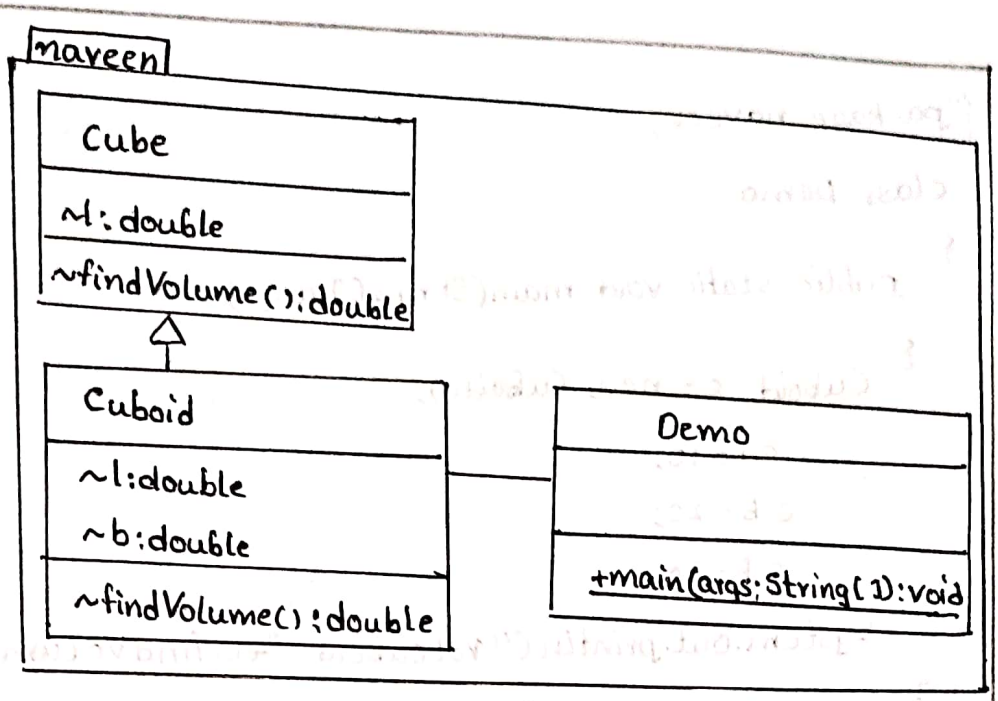
```
package naveen;
public class TestConstChild extends TestConst
{
   public TestConstChild()
   {
       super(20);
       System.
```

(i) To call a super class constructor from a child class, the child class must contains the first statement as super() or super(argument).

(ii). We can't place super() or super(argument) as second or third statement.

(iii) we can call only one super constructor.

1.

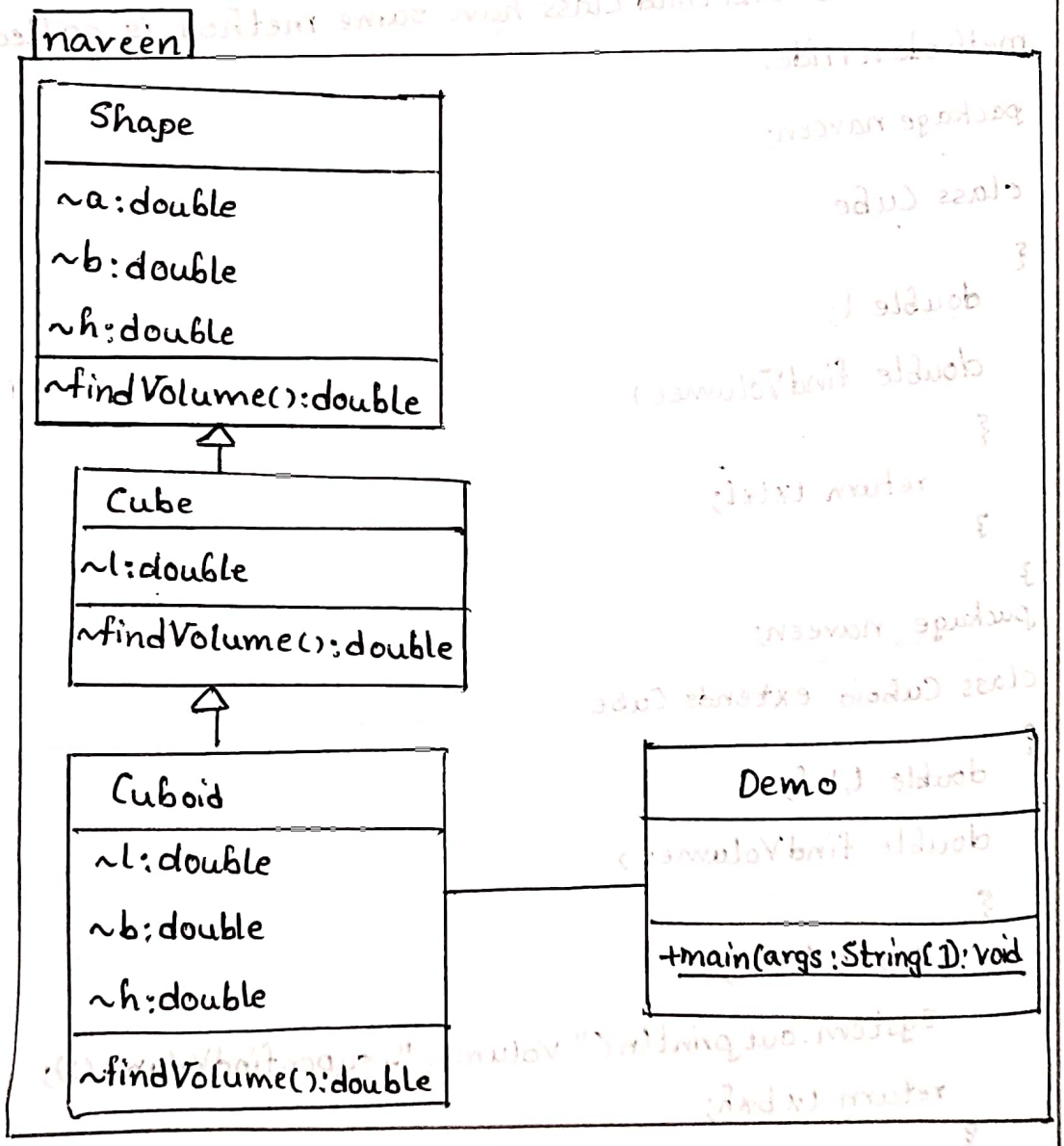


parent class and child class have same method is called methodoverride.

```
package naveen;
class Cube
{
   double l;
   double findVolume()
   {
      return l*l*l;
   }
}
package naveen;
class Cuboid extends Cube
{
   double l,b,h;
   double findVolume()
   {
      super.l=25;
      System.out.println("Volume="+super.findVolume());
      return l*b*h;
   }
}
```

```
package naveen;
class Demo
{
   public static void main(String[] args)
   {
      Cuboid c = new Cuboid();
            c.l = 10;
            c.b = 20;
            c.h = 30;
      System.out.println("Vol cuboid = "+c.findVolume());
   }
}
```

2.



```
package naveen;
class Shape
{
    double l,b,h;
    double findVolume()
    {
        return l*b*h;
    }
}
```

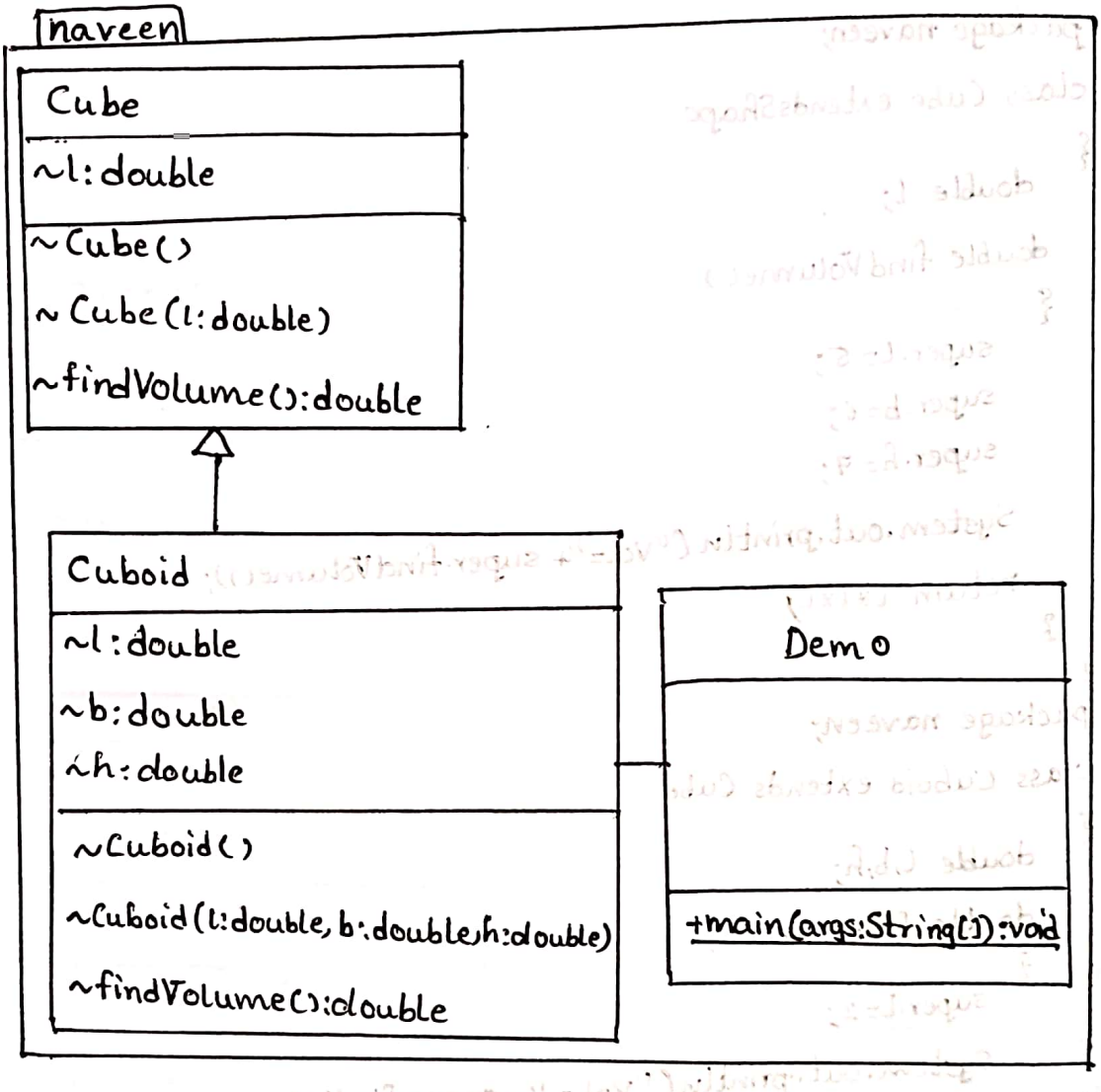
```
package naveen;
class Cube extends Shape
{
    double l;
    double findVolume()
    {
        super.l=5;
        super.b=6;
        super.h=7;
        System.out.println("Vol="+ super.findVolume());
        return l*l*l;
    }
}
```

```
package naveen;
class Cuboid extends Cube
{
    double l,b,h;
    double findVolume();
    {
        super.l=2;
        System.out.println("Vol = "+super.findVolume());
        return l*b*h;
    }
}
```

```
package naveen;
Class Demo
{
    public static void main(String[] args) {
```

```
Cuboid c=new Cuboid();
        c.l=10;
        c.b=20;
        c.h=30;
    System.out.println("Vol= "+c.findVolume());
  }
}
```

3.

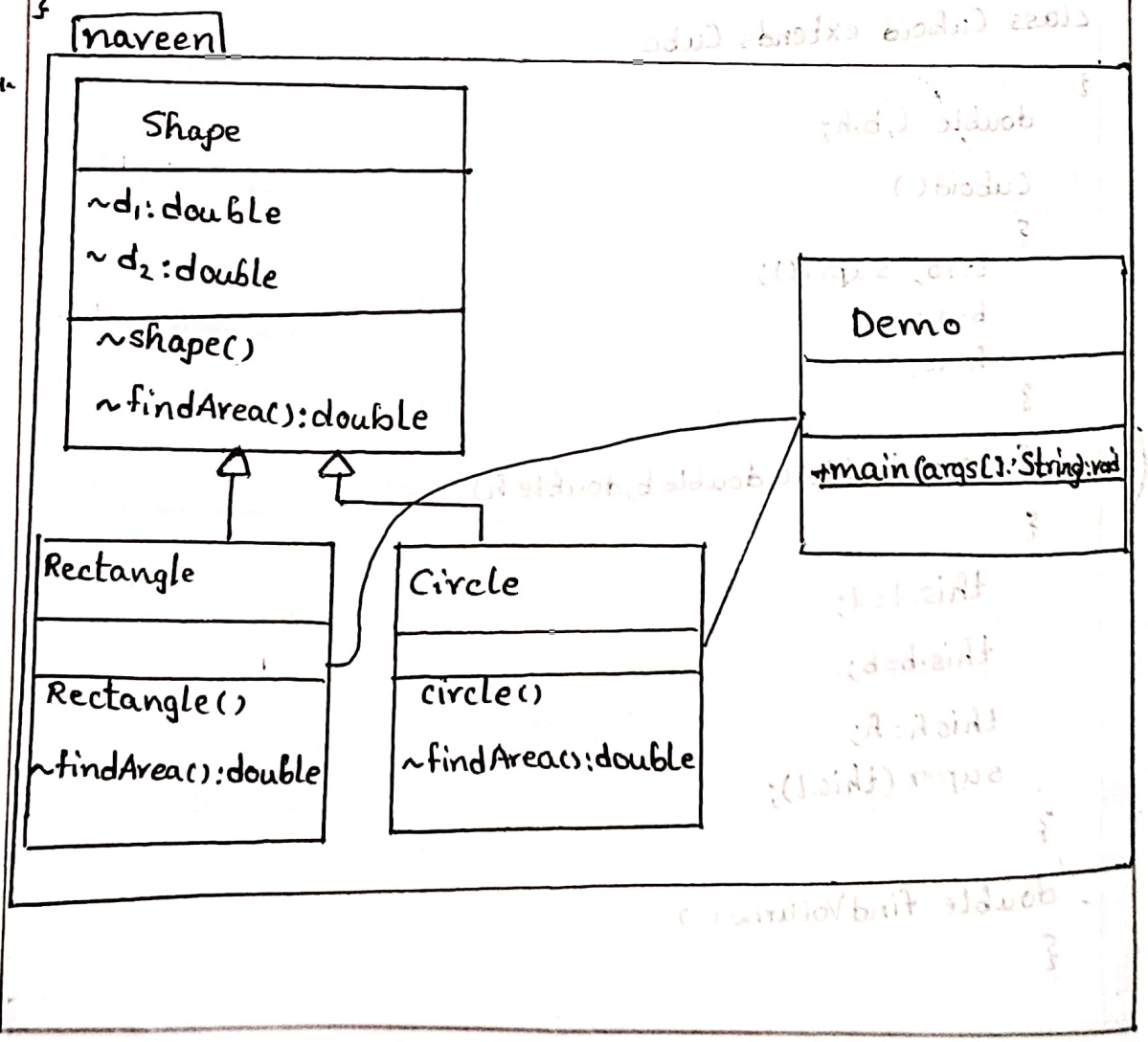


Note: Every child class constrctor have the default parent class super Constructor. we should call the parentclass constructor manually.

```
package naveen;
class Cube
{
   double l;
   Cube()
   {
      l=12;
   }
   Cube(double l)
   {
      this.l=l;
   }
   double findVolume()
   {
      return l*l*l;
   }
}
package naveen;
class Cuboid extends Cube
{
    double l,b,h;
    Cuboid()
    {
       l=10; super();
       b=11;
       h=12;
    }
    Cuboid(double l,double b,double h)
    {
       this.l=l;
       this.b=b;
       this.h=h;
       super(this.l);
    }
    double findVolume()
    {
```

```
        System.out.println("Vol=" +super.findVolume());
        return l*b*h;
    }
}
package naveen;
class Demo
{
    public static void main(String[] args)
    {
        Cuboid c = new Cuboid();
        Cuboid d = new Cuboid(5,6,7);
        System.out.println("Vol=" + c.findVolume());
        System.out.println("Vol=" +d.findVolume());
    }
}
```

4.



To assign the default shape dimension 10,20 and default area is 0.

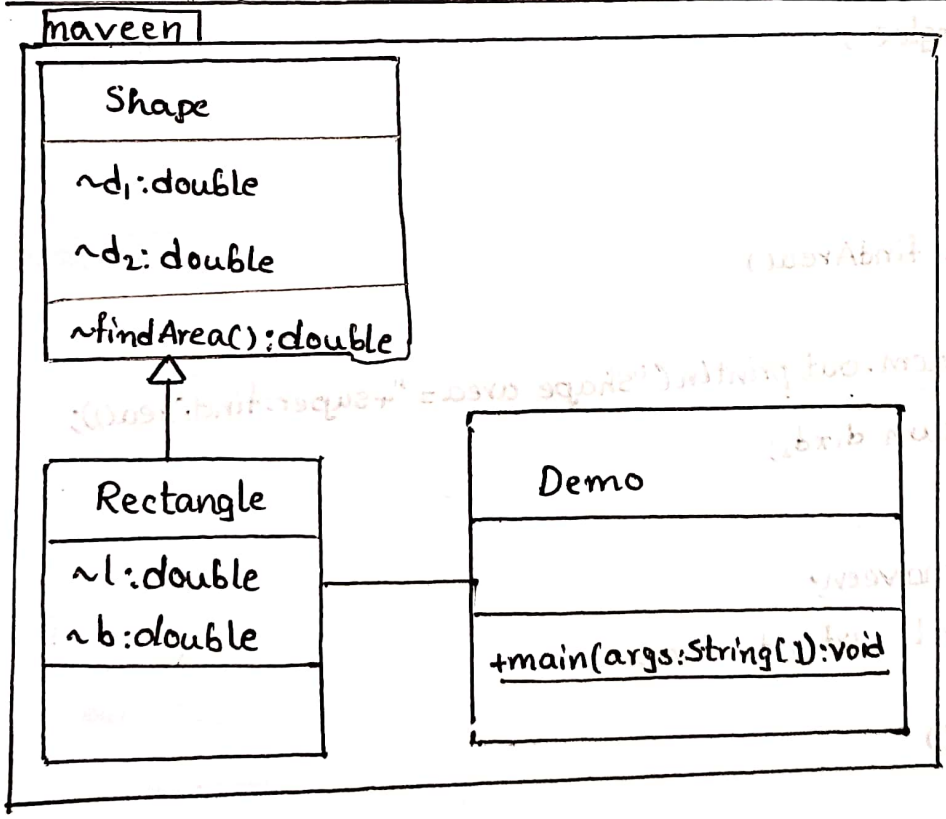
To create the object for rectangle and circle to find

```
package naveen;
class Shape
{
   double d1,d2;
   shape()
   {
      d1=10;
      d2=20;
   }
   double findArea()
   {
      return 0;
   }
}
package naveen;
class Rectangle extends Shape
{
   Rectangle()
   {

   }
   double findArea()
   {
      System.out.println("Shape area= "+super.findArea());
      return d1*d2;
   }
}
package naveen;
class Circle extends Shape
{
   Circle()
   {

   }
   double findArea(){
```

```
        System.out.println("Shape area="+super.findArea());
        return 3.14*d1*d2;
    }
}
package naveen;
class Demo
{
    public static void main(String[] args)
    {
        Rectangle r= new Rectangle();
        Circle  c = new Circle();
        System.out.println("area="+r.findArea());
        System.out.println("area="+c.findArea());
    }
}
```

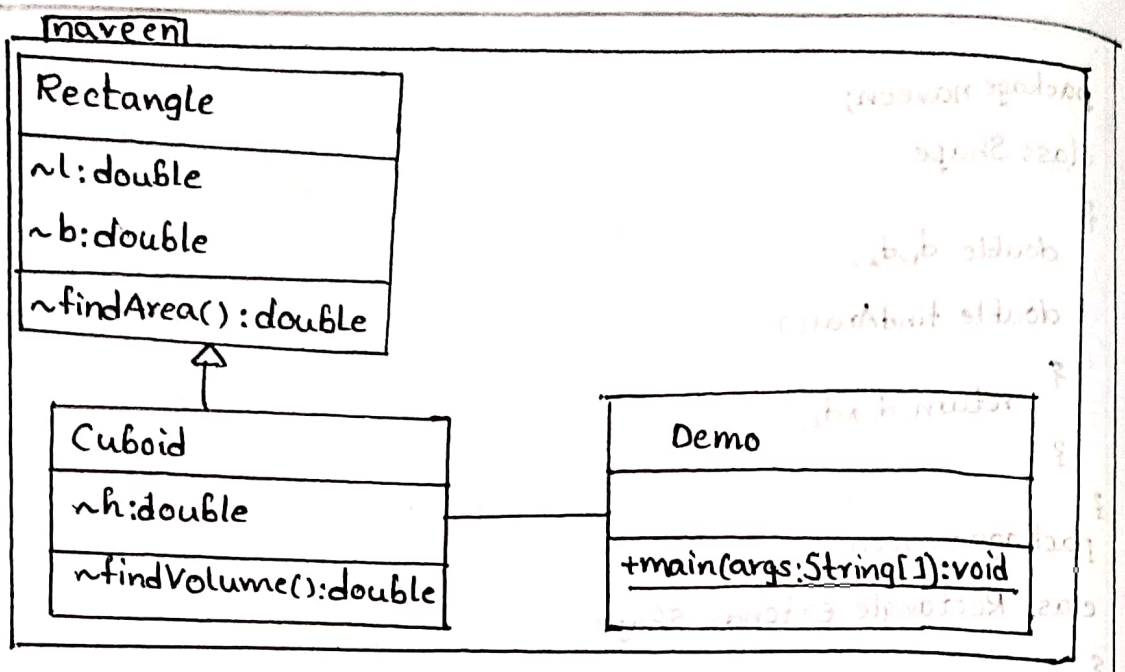
Super class object refrence to the subclass refrence.

1.



```
package naveen;
class Shape
{
   double d1,d2;
   double findArea()
   {
      return d1*d2;
   }
}
package naveen;
class Rectangle extends Shape
{
    double bb;
}
package naveen;
class Demo
{
   public static void main(String[] args)
   {
      Shape s = new Rectangle();
       s.d1 = 10;
       s.d2 = 20;
      System.out.println("area = "+s.findArea());
   }
}
```

2.




```
package naveen;
class Rectangle
{
    double l,b;
    double findArea()
    {
    return l*b;
    }
}
package naveen;
class cuboid extends Rectangle
{
    double h;
    double findVolume()
    {
        System.out.println("Rectangle Area="+ findArea());  // not there this line
        return l*b*h;
    }
}
```

```
package naveen;
public class Demo
{
    public static void main(String[] args)
    {
        Cuboid c = new Cuboid();
                c.l=10;
                c.b=11;
                c.h=12;
        System.out.println("Coboid Vol "+c.findVolume()+
                "Rect Area="+c.findArea());
    }
}
```

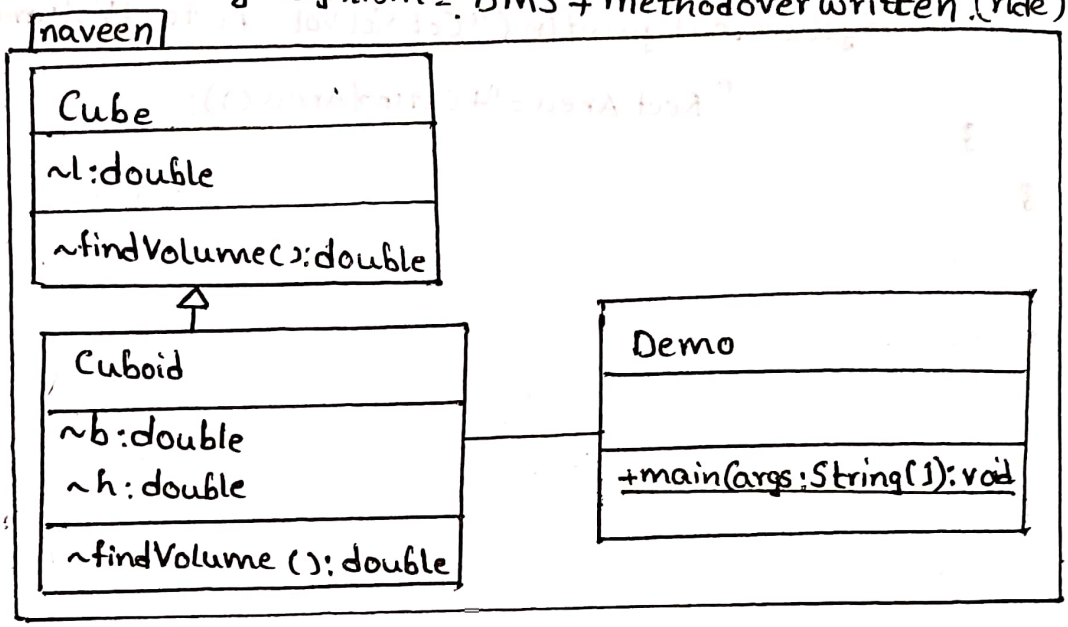
Dynamic method Dispath (runtime polymorphism):

if it create superclass object with subclass refrence, the super class object can access only overwritten method. (methodoverride)

```
className c = new SubClassName(); } Dynamic method dispath
```

runtime polymorphism = DMS + methodoverwritten (ride)

1.

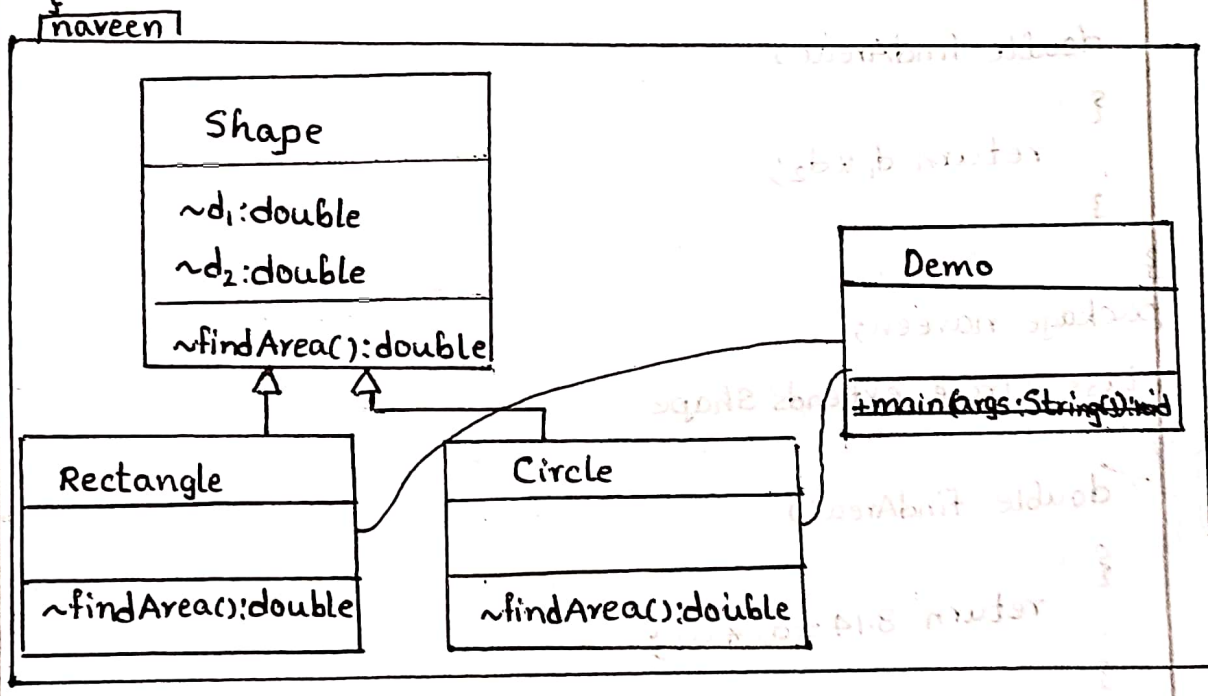



```
package naveen;
class Cube
{
   double l;
   double findVolume()
   {
      return l*l*l;
   }
}
package naveen;
class Cuboid extends Cube
{
    double b,h;
```

```
    double findVolume()
    {
        b=20;
        h=30;
        return l*b*h;
    }
}
```

```
package naveen;
class Demo
{
    public static void main(String[] args)
    {
        Cube c = new Cuboid();
            c.l = 10; // c.b,c.h x Can't access
        System.out.println("vol=" +c.findVolume()); //subclass()
    }
}
```

2.



To create superclass object with subclass refrence to access findArea method of rectangle and Cube.

```
package naveen;
class Shape
{
  double d1,d2;
  double findArea()
  {
    return d1*d2;
  }
}
package naveen;
class Rectangle extends Shape
{
  double findArea()
  {
    return d1*d2;
  }
}
package naveen;
class Circle extends Shape
{
  double findArea()
  {
    return 3.14*d2*d1;
  }
}
package naveen;
class Demo
{
```

```
Public static void main(String[] args)
{
    Shape s = new Rectangle();
        s.d1 = 10;
        s.d2 = 20;
    System.out.println("Area=" + s.findArea());
    Shape s1 = new Circle();
        s1.d1 = 10;
        s1.d2 = 10;
    System.out.println("Area=" + s1.findArea());
}
}
```

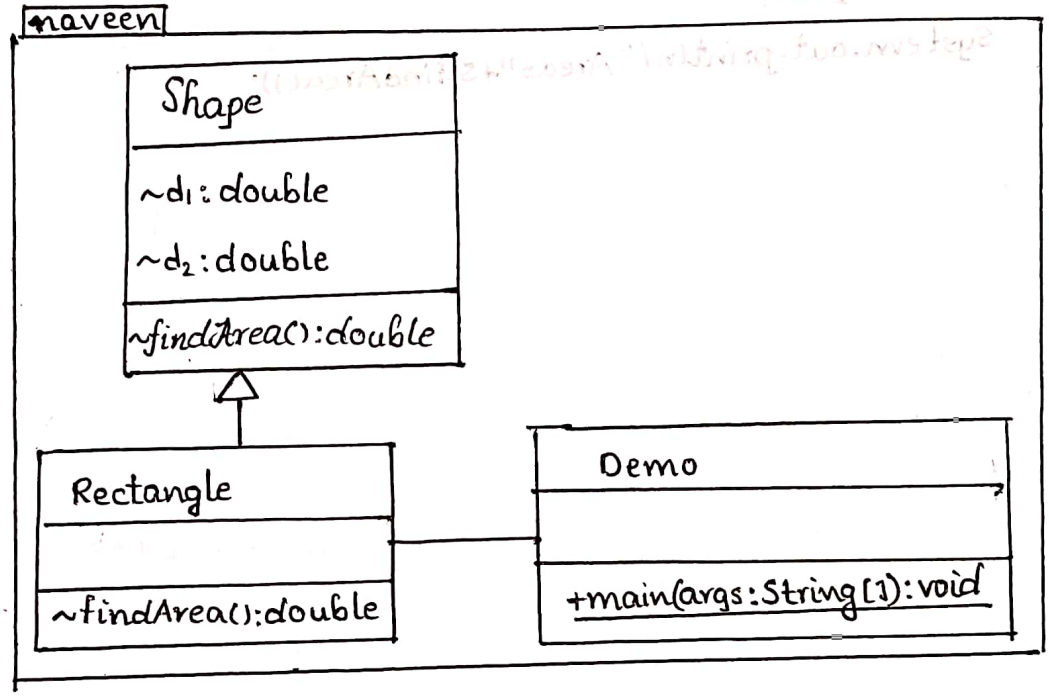
Exceptional Handling:

exception is a run-time error which arises during the execution of java program. The term exception in java stands for an "exceptional event".

So exceptions are nothing but some abnormal and typ

Abstract Class

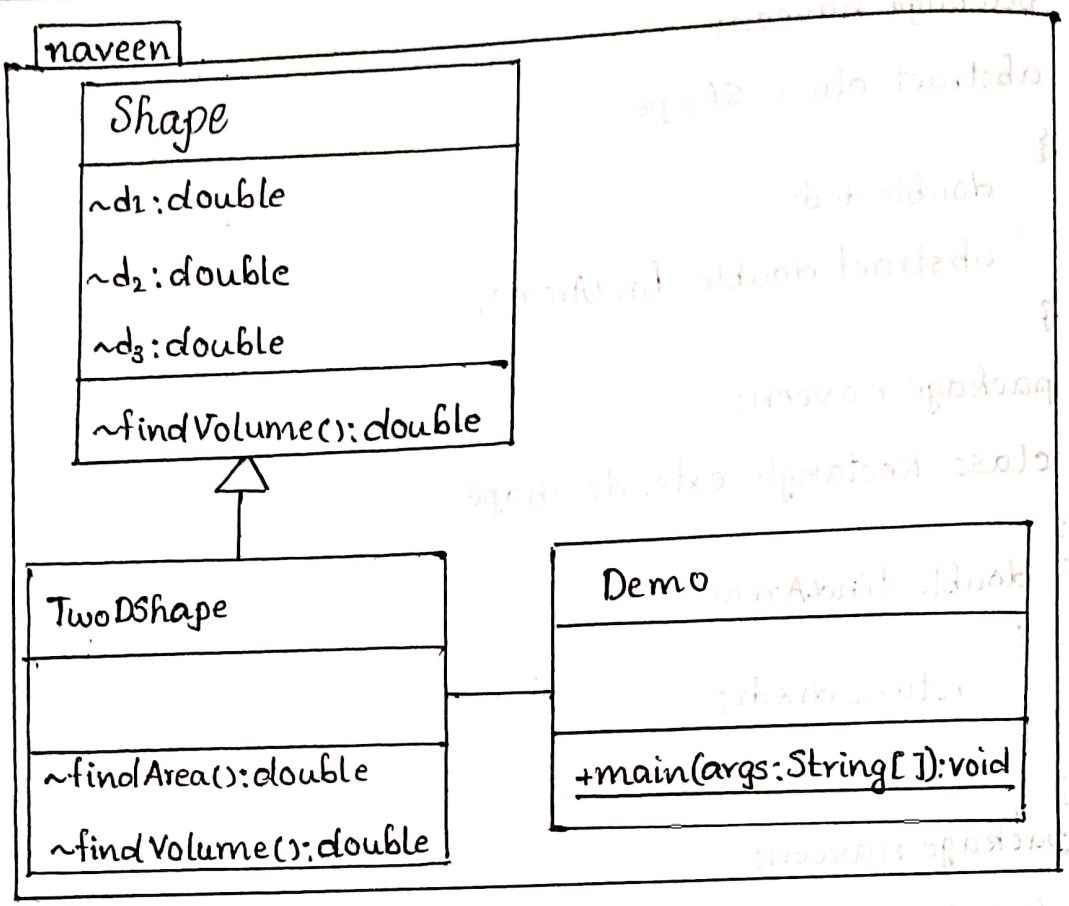
1.



In abstract class, classname and method we have to write in italic style. For abstract class, parent class and child class have same method i.e methodoverride.

```
package naveen;
abstract class Shape
{
    double d1,d2;
    abstract double findArea();
}
package naveen;
class Rectangle extends Shape
{
  double findArea()
  {
     return d1*d2;
  }
}
package naveen;
class Demo
{
  public static void main(String args[])
  {
     Shape s=new Rect
        s.d1=12;
        s.d2=14;
      System.out.println("Area="+s.findArea());
  }
}
```

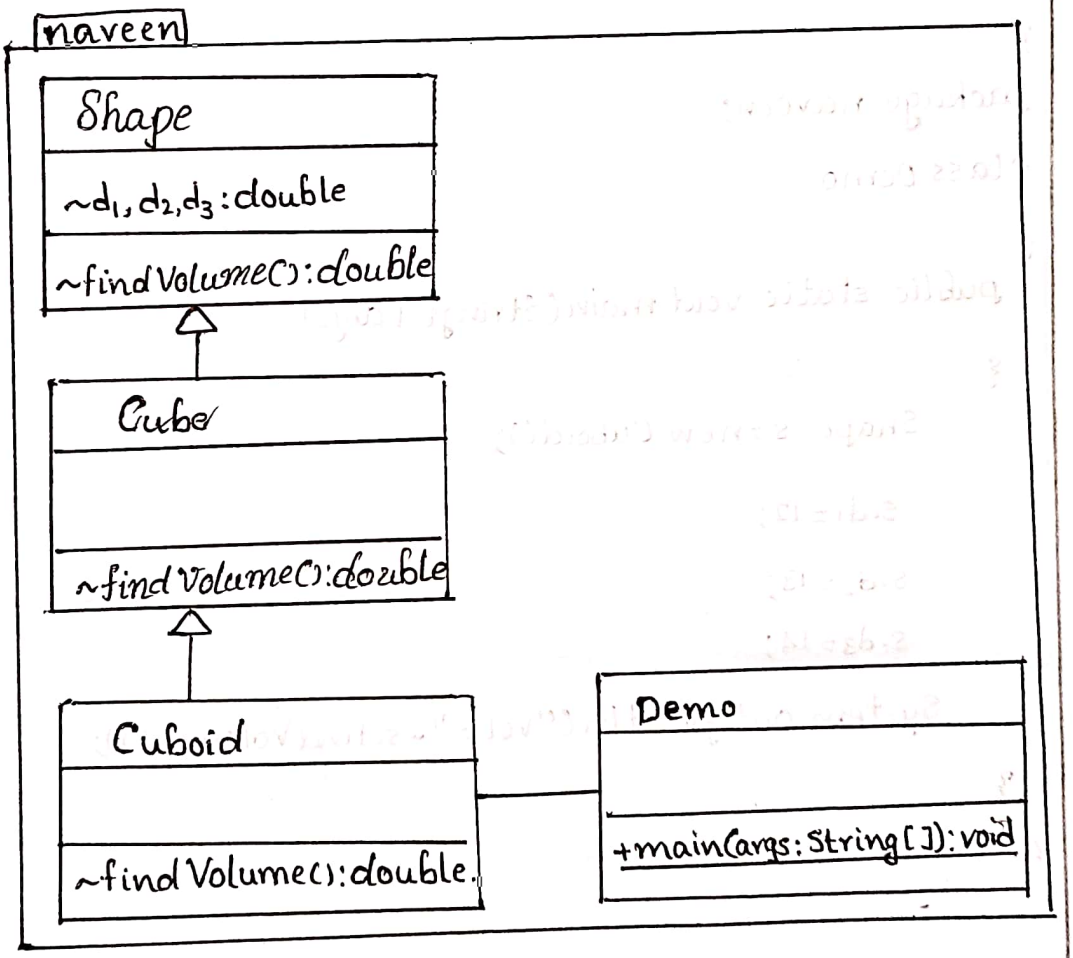
2.




```
package naveen;
abstract class Shape
{
   double d1, d2, d3;
   abstract double findVolume();
}
package naveen;
class TwoDShape extends Shape
{  double findArea()
   {
      return find d1 * d2;
   }
   double findVolume()
   {
      return d1*d2*d3;
   }
}
```

```
package naveen;
class Demo
{
  public static void main(String[] args)
  {
      Shape s=new TwoDShape();
       s.d1= 12;
       s.d2 = 13;
       s. d3 = 15;
      System.out.println("vol= " +s.findVolume());
  }
}
```

3.




```
package naveen;
abstract class Shape
{
  double d1,d2,d3;
  abstract double findVolume();
}
```

```
package naveen;
abstract class Cube extends Shape
{
   abstract double findVolume();
}
package naveen;
class Cuboid extends Cube
{
   double findVolume()
   {
      return d1*d2*d3;
   }
}
package naveen;
class Demo
{
  public static void main(String[] args)
  {
       Shape s=new Cuboid();
        s.d1=12;
        s.d2=13;
        s.d3=14;
      System.out.println("vol="+s.findVolume());
  }
}
```

Final keyword Constant.

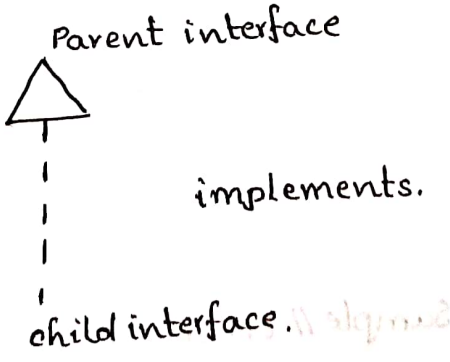
1. Member constant member
2. Method can't override
3. Class can't inherit

1.
```
package naveen;
class Sample
{
   final int MIN=12;
   void display()
   {
      MIN=26;//error
   }
}
package naveen;
class Demo
{
  public static void main()
   {
       Sample s=new Sample();
          s.MIN=15;//error
       System.out.println("value="+s.display());
   }
}
```

3.
```
package naveen;
final class Sample
{

}
package naveen;
class Demo extends Sample // error
{

}
```

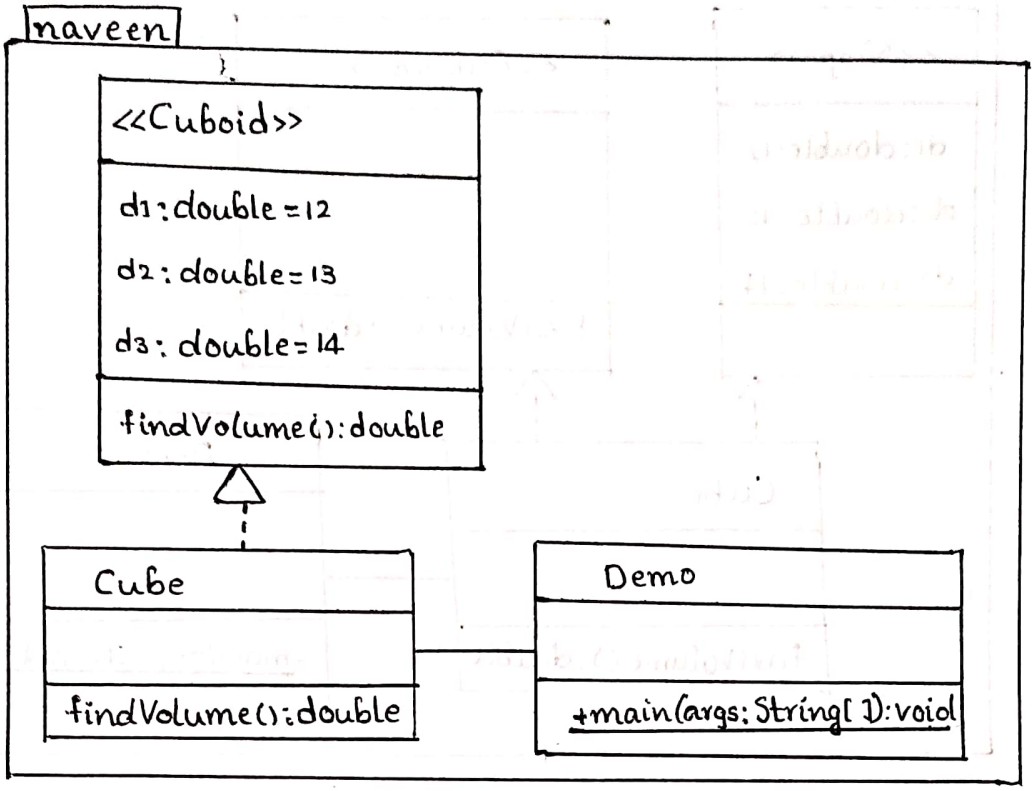
2.
```
package naveen;
class Sample
{
    int m=12;
    final void display()
    {

    }
}
package naveen;
class Demo extends Sample
{
    void display()
    {
      //can't override
    }
}
```

Interface

1. multiple inheritance - interface.
2. 100% abstraction
3. all the members are default final static
4. all the methods are default ~~are~~ (== +) public.
5. 

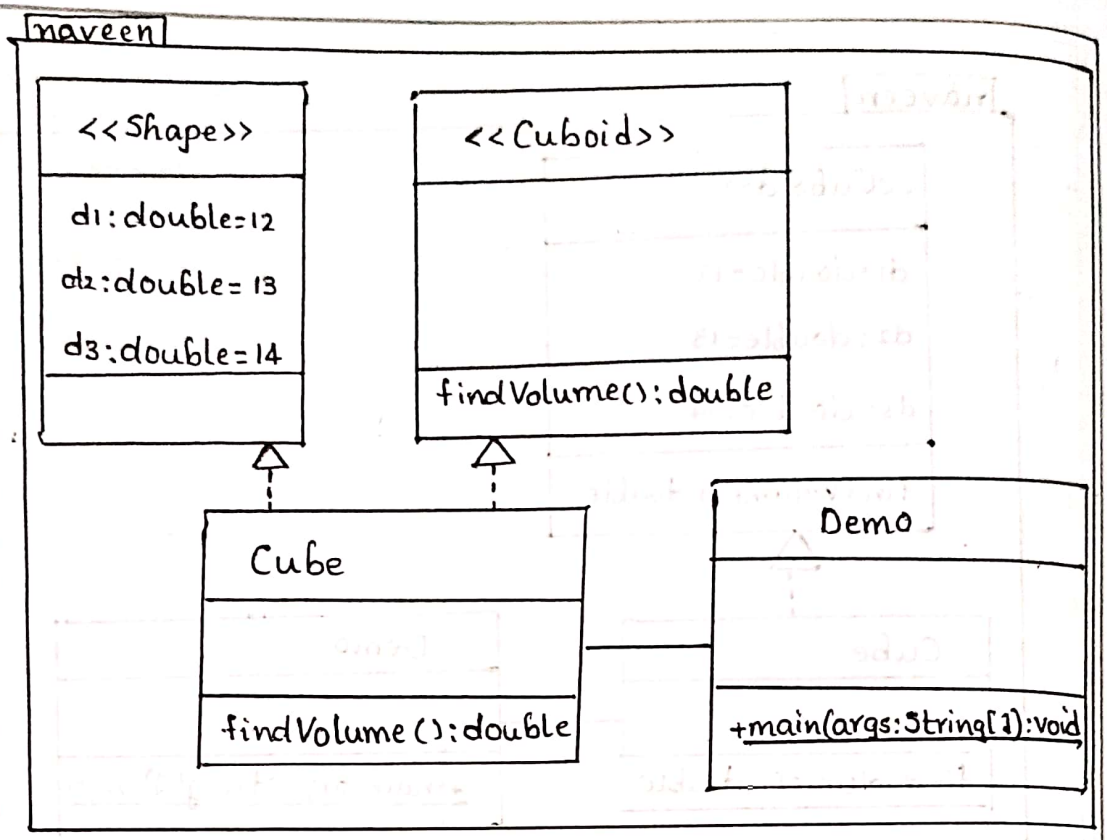
6. ~~clas~~ to denote interface by << >>

1.




```
package naveen;
interface Cuboid
{
    double d1=12; // final static double d1=12 (default);
    double d2=13,d3=14;
    double findVolume(); //public abstract double findVolume();
}                                                    (default)
package naveen;
class cube implements Cuboid{
   double findVolume(){
    return d1*d1*d1;
}
}
package naveen;
class Demo{
  public static void main(String[] args){
     Cuboid c=new Cube(); // DMD-right ans
   //Cube cb=new Cube(); //right ans
   // Cuboid c=new Cuboid(); //wor(error)
  System.out.println("Vol="+c.findVolume());
 }
}
```

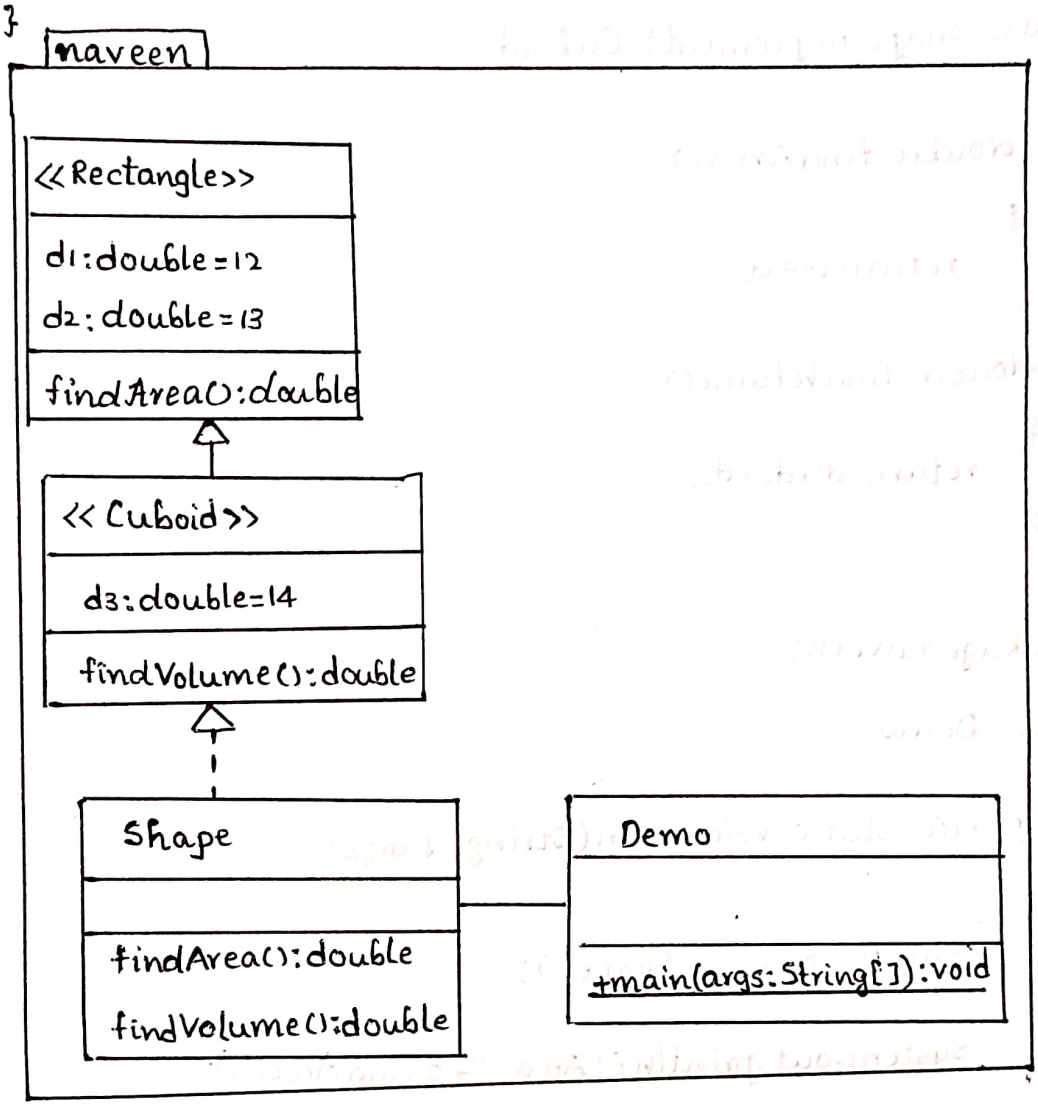
2.




```
package naveen;
interface Shape
{
   double d1=12, d2=13, d3=14;
}
package naveen;
interface Cuboid
{
  double findVolume();
}
package naveen;
class Cube implements Shape, Cuboid
{
    double findVolume()
    {
        return d1*d1*d1;
    }
}
```

```
package naveen;
class Demo
{
   public static void main(String args[])
   {
      Cuboid c = new Cube();
      System.out.println("Vol=" + c.findVolume());
   }
}
```

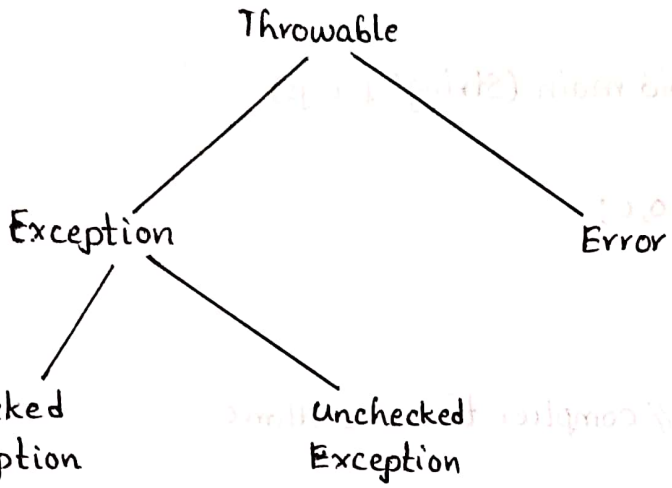
3.




```
package naveen;
public interface Rectangle
{
   final static double d1=12, d2=13;
   public abstract findArea();
}
```

```
package naveen;
interface Cuboid extends Rectangle
{
    double d3=14;
    double findVolume();
}
package naveen;
class Shape implements Cuboid
{
    double findArea()
    {
        return d1*d2;
    }
    double findVolume()
    {
        return d1*d2*d3;
    }
}
package naveen;
class Demo
{
    public static void main(String[] args)
    {
        Shape s=new Shape();
        System.out.println("Area="+s.findArea());
        System.out.println("Vol="+s.findVolume());
    }
}
```

Exception Handling



IOException
FileNotException
classNotFound Exception
NosuchField Exception

ArithmeticException
IndexOutofBoundsException
NullPointerException
NumberFormatException

Keywords:

try ---> risk of code.

catch---> deal of Error

throw---> unchecked Excep manually --obj

throws---> checkedexcep--method

finally ---> always finally will exc

1.
```
package naveen;
class Sample
{
  public static void main (String[ ] args)
  {
     int a=12, b=0, c;
    try
    {
       c = a/b; // complier throw arithme
    }
    catch (ArithmeticException e)
    {
       e.printStackTrace(); // System.out.println("error");
                            // System.out.println(e);
  }
}
```

2.
```
package naveen;
class Sample
{
  public static void main (String[ ] args)
  {
    try
    {
       int a[ ]=new int[5];
        a[8]=12;
    }
    catch(ArrayIndexOutofBoundsException e)
    {
      e.printStackTrace(); //System.out.println("Math error");
                           //System.out.println(e);
    }
  }
}
```

3.
```
package naveen;
class Sample
{
  public static void main(String[] args)
  {
     try
     {
        String s=null;
        int l=s.length;
     }
     catch(NullPointerException e)
     {
        System.out.print(e);
     }
  }
}
```

4.
```
package naveen;
class Sample
{
  public static void main(String[]args)
  {
     try
     {
        String s="abc";
        int x = integer.parseInt(s);
     }
     catch(NumberFormatExceptione)
     {
        System.out.print(e);
     }
  }
}
```