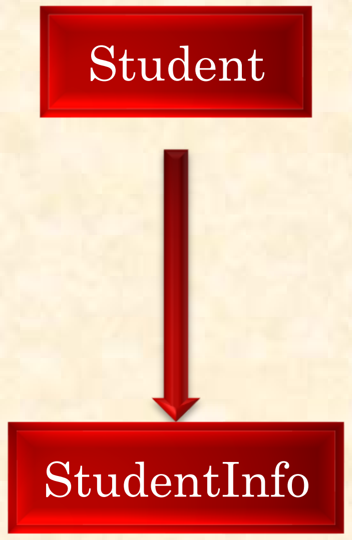
Polymorphism & Inheritance

Nihar Ranjan Roy

What is Inheritance?

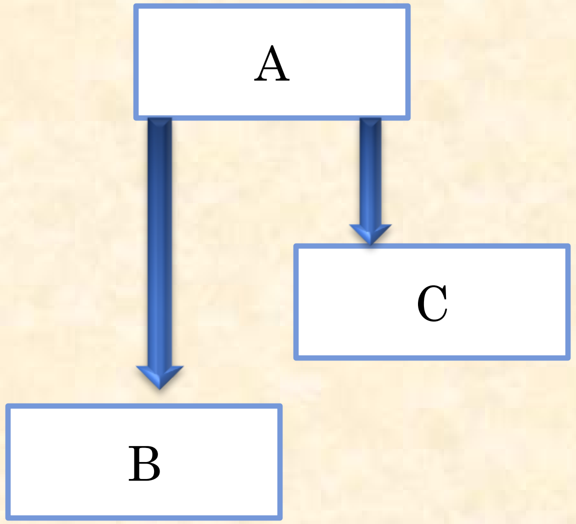
```
class Student
{
String name;
int roll;
}
```

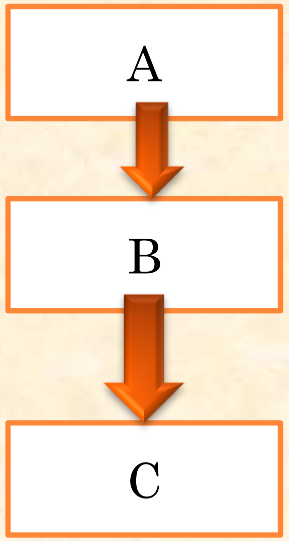
```
class Student
{
String name;
int roll;
String city;
String  Phone_no
}
```

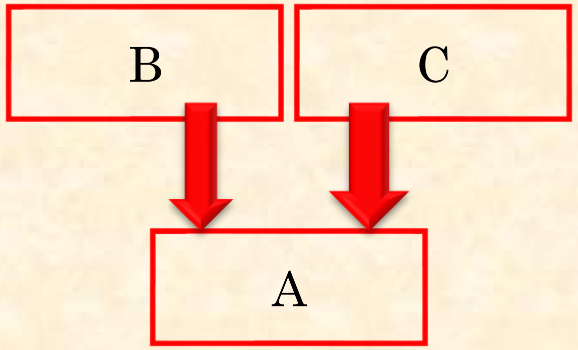


```
class StudentInfo extends Student
{
String City;
String Phone_No;
}
```

Multi-Level inheritance







Multiple Inheritance

Simple Inheritance

Multi Level Inheritance

Problem

You have a student class with members name and roll but your client has asked for some more information to be associated with the student i.e city and phone number. How inheritance can help u in this? Write the program for accepting the info and displaying it.

```
class Student
{
String name;
int roll;
Student()
{ }
Student(String n,int r)
{name=n;
roll=r;
}
void disp()
{
System.out.println("\nName
is==>"+name+"\nRoll no
is==>"+roll);
}
}
```

```
class StudentInheri
{
public static void main(String args[])
{StudentInfo st;
st=new StudentInfo("nihar",1,"Gtr
noida","123456789");
st.disp();
}
}
```

```
class StudentInfo extends Student
{
String city;
String phone;
StudentInfo()
{}

StudentInfo(String n,int r,String c,String ph)
        {
        name=n;
        roll=r;
        city=c;
        phone=ph;
        }

void disp()
        {
        System.out.println("name=>"+name);
        System.out.println("roll=>"+roll);
        System.out.println("city=>"+city);
        System.out.println("phone=>"+phone);
}       }
```

Access Specifiers?

Use of super key word

A sub class can call the constructor of its super class with the super key word

super(<parameter list>)

Note➔

1) super must be the first statement
2) super refers to the immediate super class

```
class Student
{
String name;
int roll;
Student()
{ }
Student(String n,int r)
{name=n;
roll=r;
}
void disp()
{
System.out.println("\nName
is==>"+name+"\nRoll no
is==>"+roll);
}
}
```

```
class StudentInheri
{
public static void main(String
args[])
{StudentInfo st=new
StudentInfo("nihar",1,"Gtr
Noida","123456789");
st.disp();
}
}
```

```
class StudentInfo extends Student
{
String city;
String phone;
StudentInfo()
{}

StudentInfo(String n,int r,String c,String ph)
        {
        super(n,r);
        city=c;
        phone=ph;
        }

void disp()
        {
        System.out.println("name=>"+name);
        System.out.println("roll=>"+roll);
        System.out.println("city=>"+city);
        System.out.println("phone=>"+phone);
}       }
```

Multiple inheritance in java?

```
 class A extends B,C
{
….
}

Or
class A extends B extends C
{
…..
}
```

INTERFACES

Syntax

```
[access-specifier] interface <interface-name>
{
return_type method_name(Parameters);
return_type method_name1(Parameters);
…..
final data_type var_name=value;
}
```

USE

```
 class A implements B
{//Definition of methods of interface B
}
```

1. Methods in a interface are abstract
2. varibles are implicitly final and static
3. Interface method cannot be static,final, native ,private or protected

What is the difference between a class and an interface?

Problem

```
interface shape
{
double pi=3.14;
double volume();
}
```

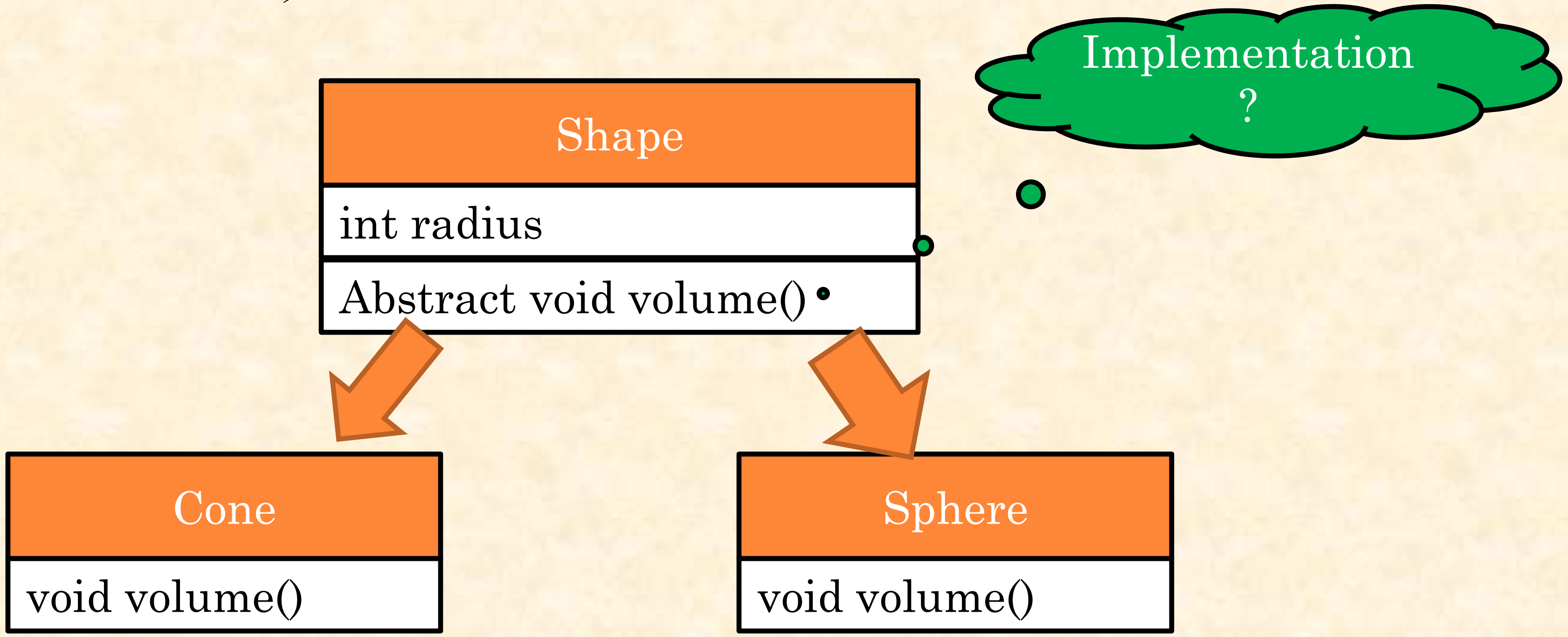
USE THIS INTERFACE TO FIND THE VOLUME OF A SPHERE CLASS AND A CONE CLASS

```
interface Shape
{
double pi=3.14;
public void volume();
}

class Cone implements Shape
{
int r,h;//radius and height
Cone(){}
Cone(int rad,int height){r=rad;h=height;}

public  void volume()
{
System.out.println("Volume  is "+(pi*r*r*h)/3);
}
public static void main(String args[])
{
Cone obj=new Cone(2,3);
obj.volume();
}
}
```

What is a abstract class



Abstract class are incomplete so we cannot create objects out of it. The inheriting class has to implement the abstract methods before using it

Rules for abstract class

- Abstract method➔ is a method without implementation

  Example➔ abstract void disp();
- Abstract class can have concrete methods also
- A class with at least one abstract method is a abstract class
- A class without an abstract method can also be declared abstract
- Cannot declare abstract constructors
- Cannot declare abstract static methods

```
abstract class Shape
{ int radius;

Shape(int r){radius=r;}
 void disp(){System.out.println("Radius of shape"+radius);}
 abstract void volume();

 }

  class Cone extends Shape
  {
  int height;
  Cone(int r,int h)
  {
  super(r);
  height=h;
  }
  void volume()
  {
  System.out.println((3.14*radius*radius*height)/3);

   }
  }
```

```
class AbstractClass
 {

  public static void main(String
args[])
 {
 Cone obj=new Cone(2,3);
 obj.disp();
 obj.volume();
 }
}
```

Abstract vs concrete class

Abstract class	Concrete class
Cannot have objects	Can have objects
Not necessary that all methods are implemented	Implements all of its methods
Use less without sub class	May have sub class

What is polymorphism?

The ability to appear in many forms.

In object-oriented programming, *polymorphism* refers to a programming language's ability to process objects differently depending on their data type or class. More specifically, it is the ability to redefine *methods* for *derived classes*

What is method overloading?

```
class SUM
{……
    Add (int I,int j)
    {….}
    Add (float f, float g)
    {….}
    Add (float f, int a)
    {…}
…
}
```

Nihar Ranjan Roy

What is constructor overloading?

```
class Student
{
String name;
int roll;

Student()
{ }

Student(String n, int r)
{name=n;
roll=r;
}

void disp()
{
………
……..
}
}
```

?

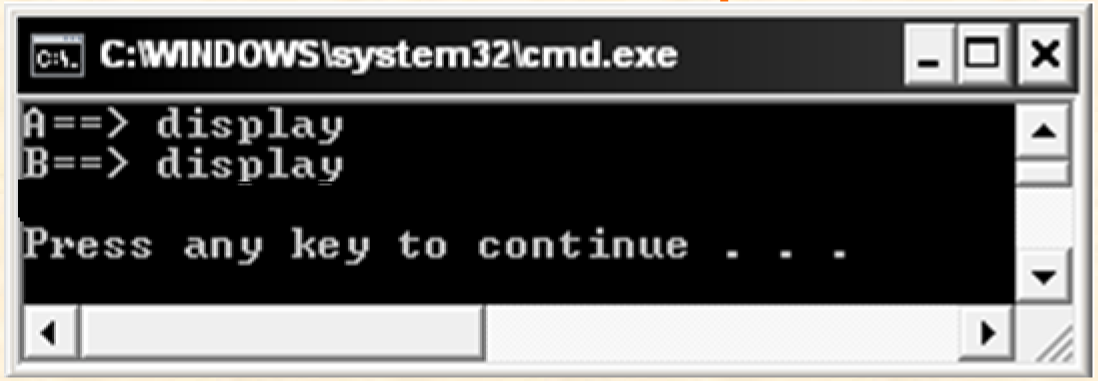
Method over riding

```
class A
{
void disp()
{System.out.println("A==> display");
}
}
class B extends A
{
void disp()
{System.out.println("B==> display");
}
}
```

```
class DyanicBinding
{
public static void main(String
args[])
{
A objA=new A();
objA.disp();

B objB=new B();
objB.disp();

}
}
```



```
A==> display
B==> display

Press any key to continue . . .
```

Overloading vs overriding

Over loading	Over ridding
Two of more methods use the same name with different parameters	Super class and sub class use the same method with same signature/parameters
Methods belong to same class	Methods belong to super class and sub class
Signature varies	Signature is same
Static binding	Dynamic binding

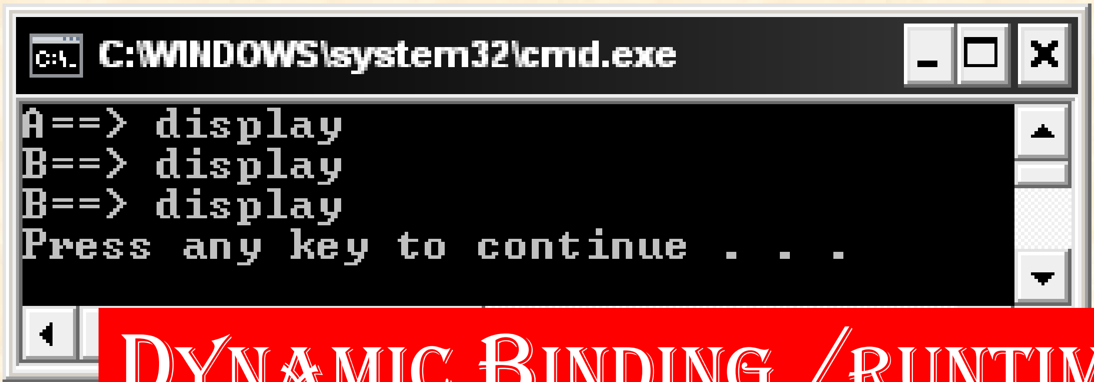
Dynamic Binding /runtime polymorphism

```
class A
{
void disp()
{System.out.println("A==>
   display");
}
}
class B extends A
{
void disp()
{System.out.println("B==>
   display");
}
}
```

```
class DyanicBinding
{
public static void main(String
args[])
{
A objA=new A();
objA.disp();

B objB=new B();
objB.disp();

objA=new B();
objA.disp();
}
}
```


C:\WINDOWS\system32\cmd.exe

```
A==> display
B==> display
B==> display
Press any key to continue . . .
```

Dynamic Binding /runtime polymorphism

Nihar Ranjan Roy

Problem

I want that my method or my class should not be inherited. How can I prevent it from being inherited by others?

There is one special class in java that is inherited by all other classes. which is that class?