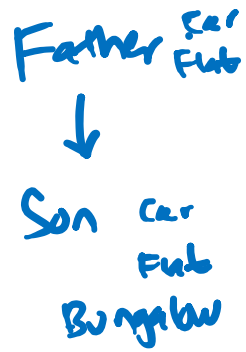
Inheritance

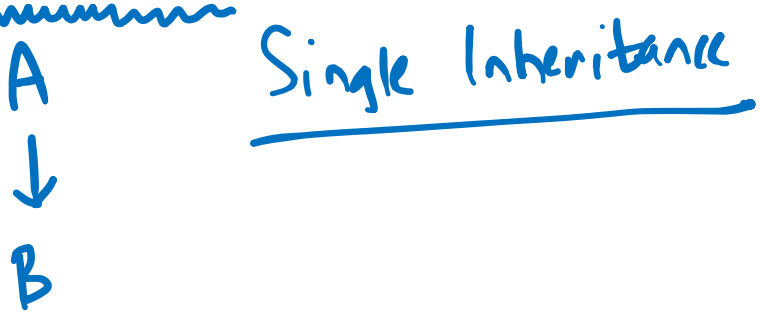
In the terminology of Java, a class that is inherited is called a *superclass. The class that does the inheriting is called a subclass.*

Therefore, a subclass is a specialized version of a superclass.

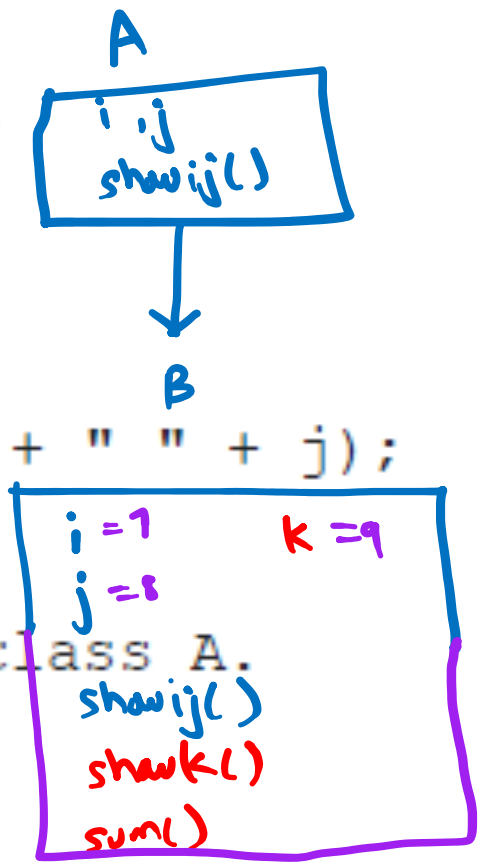
It inherits all of the instance variables and methods defined by the superclass and adds its own, unique elements.



Inheritance Basics To inherit a class, you simply incorporate the definition of one class into another by using the **extends keyword. To see how, let's begin with a short example. The following program creates a superclass called A and a subclass called B. Notice how the keyword extends is used to create a subclass of A.**



```
// A simple example of inheritance.
// Create a superclass.
class A {
int i, j; //instance
void showij() {
System.out.println("i and j: " + i + " " + j);
}
}
// Create a subclass by extending class A.
class B extends A {
int k; //instance
void showk() {
System.out.println("k: " + k);
}
void sum() {
System.out.println("i+j+k: " + (i+j+k));
}
}
```



```
class SimpleInheritance {
public static void main(String args[]) {
A superOb = new A();
B subOb = new B();
// The superclass may be used by itself.
superOb.i = 10;
superOb.j = 20;
System.out.println("Contents of superOb: ");
superOb.showij();
System.out.println();
/* The subclass has access to all public members of
its superclass. */
subOb.i = 7;
subOb.j = 8;
subOb.k = 9;
System.out.println("Contents of subOb: ");
subOb.showij();
subOb.showk();
System.out.println();
System.out.println("Sum of i, j and k in subOb:");
subOb.sum();
}
}
```

The output from this program is shown here:

```
Contents of superOb:
i and j: 10 20
Contents of subOb:
i and j: 7 8
k: 9
Sum of i, j and k in subOb:
i+j+k: 24
```

As you can see, the subclass **B** includes all of the members of its superclass, **A**. This is why **subOb** can access **i** and **j** and call **showij( )**. Also, inside **sum( )**, **i** and **j** can be referred to directly, as if they were part of **B**.

You can only specify one superclass for any subclass that you create. Java does not support the inheritance of multiple superclasses into a single subclass.

(not supported in Java)

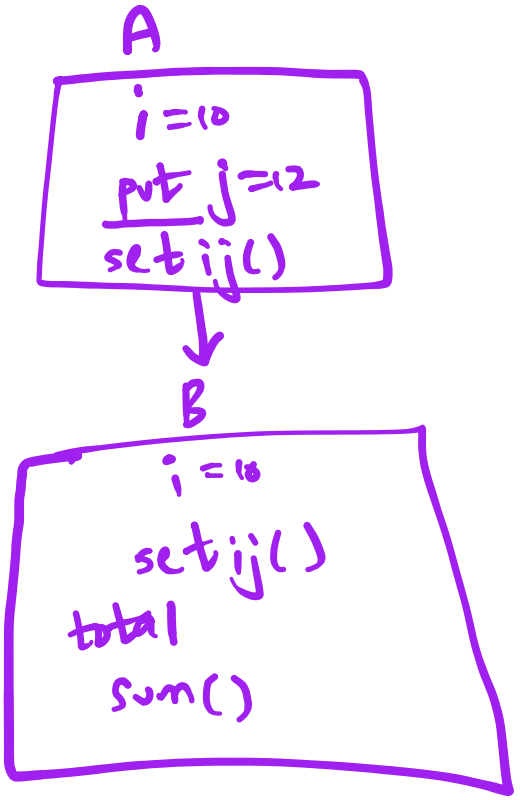
(This differs from C++, in which you can inherit multiple base classes.) You can, as stated, create a hierarchy of inheritance in which a subclass becomes a superclass of another subclass. However, no class can be a superclass of itself.

Member Access and Inheritance

Although a subclass includes all of the members of its superclass, it cannot access those members of the superclass that have been declared as **private.**

For example, consider the following simple class hierarchy:

```
/* In a class hierarchy, private members remain
private to their class.
This program contains an error and will not
compile.
*/
// Create a superclass.
class A {
int i; // public by default
private int j; // private to A
void setij(int x, int y) {
i = x; // this.i = i;
j = y; // this.j = j;
}
}
// A's j is not accessible here.
class B extends A {
int total;
void sum() {
total = i + j; // ERROR, j is not accessible here
}
}
```



```
class Access {
public static void main(String args[]) {
B subOb = new B();
subOb.setij(10, 12);
subOb.sum();
System.out.println("Total is " + subOb.total);
}
}
```

This program will not compile because the reference to **j inside the sum( ) method of B causes an access violation. Since j is declared as private, it is only accessible by other members of its own class. Subclasses have no access to it.**

Note: A class member that has been declared as private will remain private to its class. It is not accessible by any code outside its class, including subclasses.

A More Practical Example

Let‘s look at a more practical example that will help illustrate the power of inheritance.
Here, the final version of the **Box class developed in the preceding chapter will be extended to incl**ude a fourth component called **weight.**

Thus, the new class will contain a box‘s width, height, depth, and weight.

```
// This program uses inheritance to extend Box.
class Box {
double width;
double height;
double depth;
// construct clone of an object
Box(Box ob) { // pass object to constructor
width = ob.width;
height = ob.height;
depth = ob.depth;
}
// constructor used when all dimensions specified
Box(double w, double h, double d) {
width = w;
height = h;
depth = d;
}
// constructor used when no dimensions specified
Box() {
width = -1; // use -1 to indicate
height = -1; // an uninitialized
depth = -1; // box
}
```

```
// constructor used when cube is created
Box(double len) {
width = height = depth = len;
}
// compute and return volume
double volume() {
return width * height * depth;
}
}
// Here, Box is extended to include weight.
class BoxWeight extends Box {
double weight; // weight of box
// constructor for BoxWeight
BoxWeight(double w, double h, double d, double m) {
width = w;
height = h;
depth = d;
weight = m;
}
```

```
}class DemoBoxWeight {
public static void main(String args[]) {
BoxWeight mybox1 = new BoxWeight(10, 20, 15, 34.3);
BoxWeight mybox2 = new BoxWeight(2, 3, 4, 0.076);
double vol;
vol = mybox1.volume();
System.out.println("Volume of mybox1 is " + vol);
System.out.println("Weight of mybox1 is " + mybox1.weight);
System.out.println();

vol = mybox2.volume();
System.out.println("Volume of mybox2 is " + vol);
System.out.println("Weight of mybox2 is " + mybox2.weight);
}
}
```

The output from this program is shown here:

```
Volume of mybox1 is 3000.0
Weight of mybox1 is 34.3
Volume of mybox2 is 24.0
Weight of mybox2 is 0.076
```

BoxWeight inherits all of the characteristics of **Box** and adds to them the **weight** component. It is not necessary for **BoxWeight** to re-create all of the features found in **Box**. It can simply extend **Box** to meet its own purposes.

A major advantage of inheritance is that once you have created a superclass that defines the attributes common to a set of objects, it can be used to create any number of more specific subclasses.

Each subclass can precisely tailor its own classification. For example, the following class inherits **Box and adds a color attribute:**

```
// Here, Box is extended to include color.
class ColorBox extends Box {
int color; // color of box
ColorBox(double w, double h, double d, int c) {
width = w;
height = h;
depth = d;
color = c;
}
}
```

There are various types of inheritance namely:

1. Single inheritance A→B
2. Multilevel inheritance A→B→C
3. Hierarchial inheritance (show later)

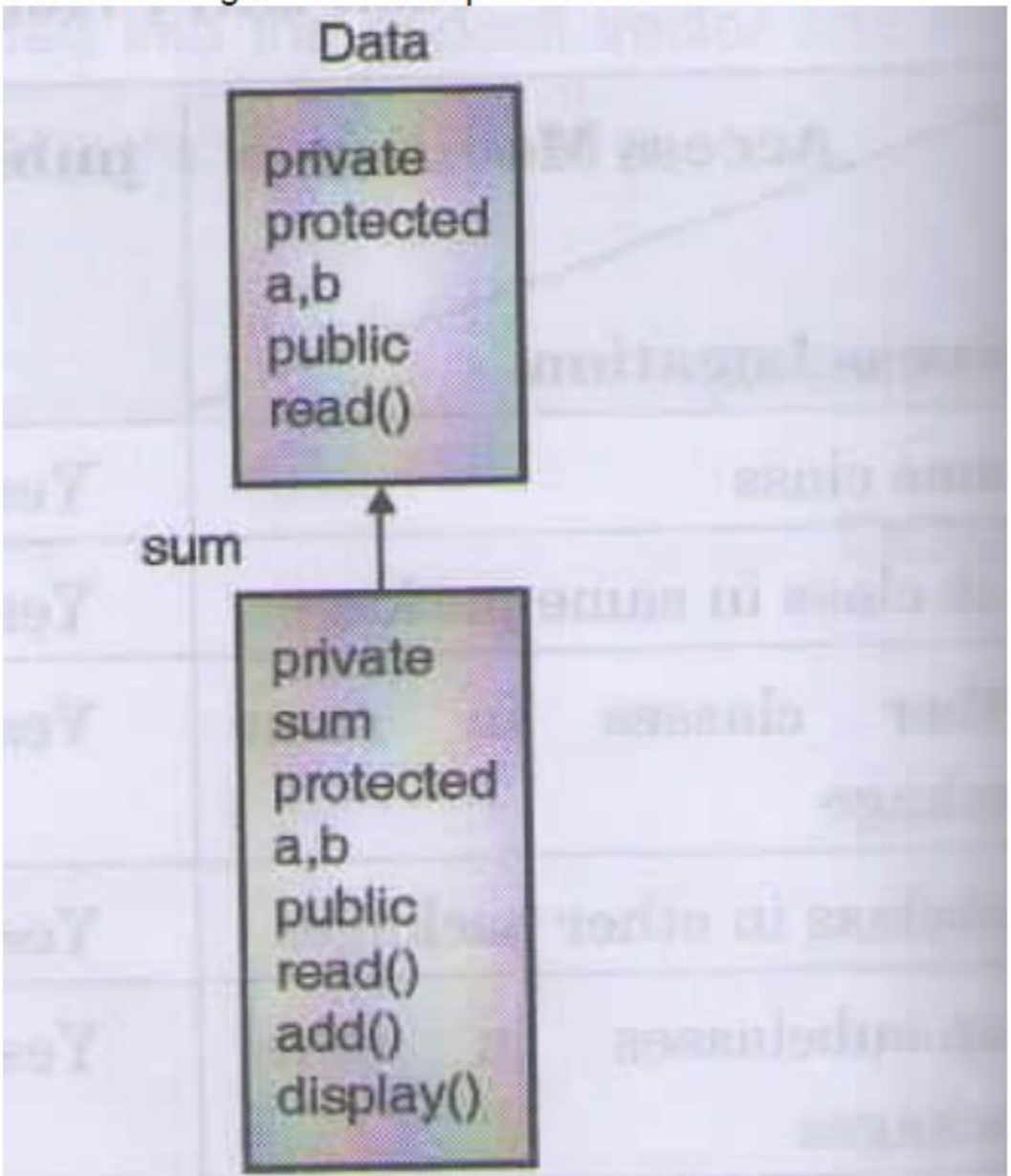
Java does not support Multiple and hence Hybrid inheritance. These can be slightly made possible using interfaces seen in the further sections of this chapter.

Single Inheritance

In this case only one class is derived from another class.

//WAP to add two numbers using single inheritance such that the base class method must accept the two //numbers from the user and the derived class method must add these numbers and display the sum.

The class diagram of this requirement is as shown:



Class Diagram of Single Inheritance

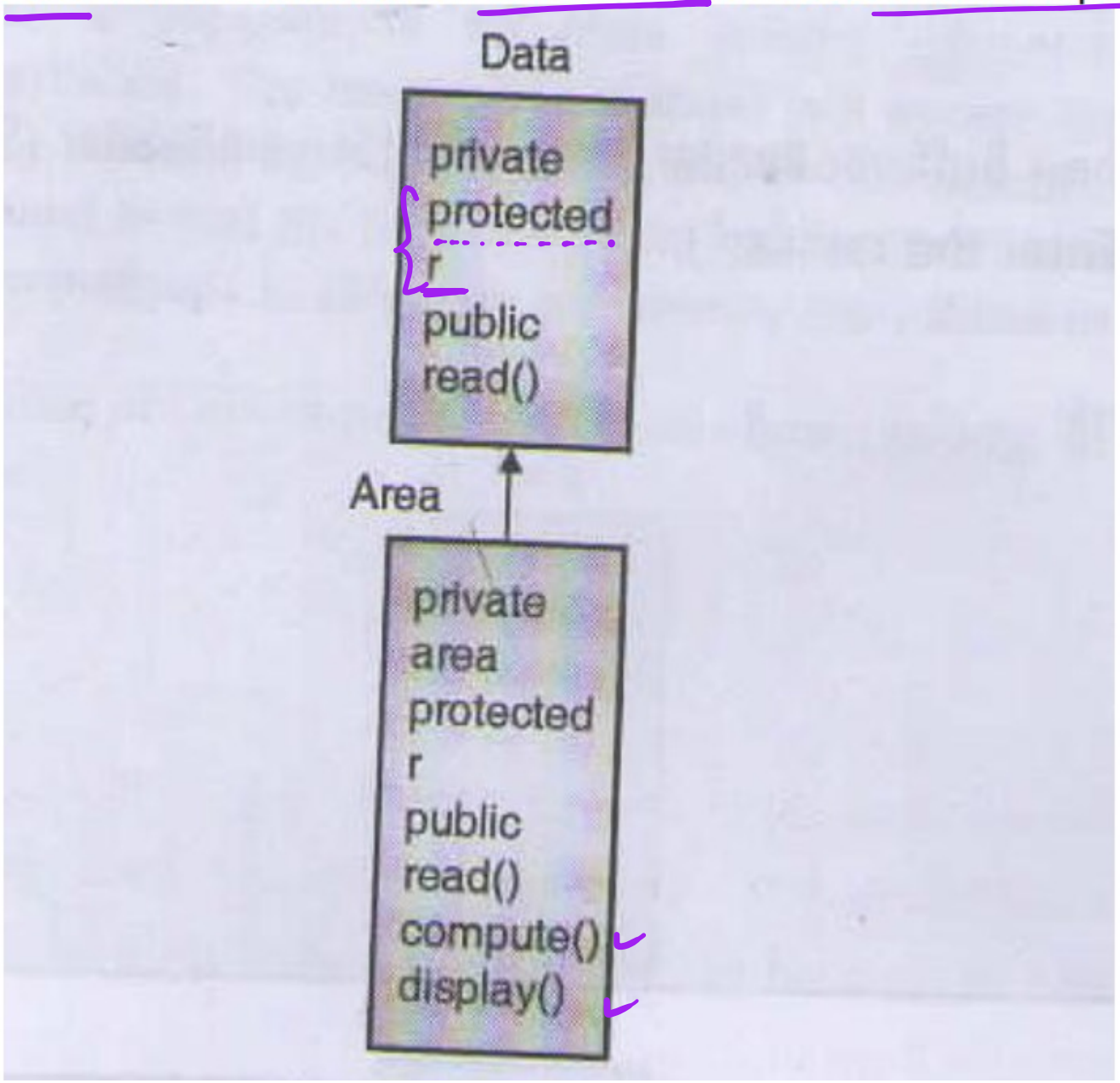
```
import java.io.*;
class Data{
protected int a, b;
public void read(int x, int y)
{
  a=x;

  b=y;
}
}
class Sum extends Data
{
private int sum;
public void add()
{
   sum=a+b;
}
public void display()
{
   System.out.println("Sum="+sum);
}
}
```

```
class Main{
public static void main (String args[]) throws IOException{
int x,y;
String str;
BufferedReader br = new BufferedReader (new InputStreamReader (System.in));
System.out.println("Enter two numbers");
str=br.readLine();
x=Integer.parseInt(str);
str=br.readLine();
y=Integer.parseInt(str);
Sum s=new Sum();
s.read(x,y);
s.add();
s.display();
}
}
```

Output:

```
Enter two numbers:
3
4
Sum=7
```

//WAP to find the area of circle using single inheritance such that the base class method must accept the radius //from the user and the derived class method must calculate and display the area.



```
import java.io.*;
class Data{
protected float r;
public void read(float x)
{
  r=x;
}
}
class Area extends Data
{

private float area;
public void calculate()
{
  area=3.14f*r*r;
}
public void display()
{
  System.out.println("Area="+area);
}
}
```

9.2

Data

protected r

read()

Area

a.

r = 9.2

read()

area = 314.0

calculate()

display()

3.14f * 9.2 * 9.2

Area = 314.0

```
class Main{
public static void main (String args[]) throws IOException{
float x;
String str;
BufferedReader br = new BufferedReader (new InputStreamReader (System.in));
System.out.println("Enter the radius:");
str=br.readLine();
x=Float.parseFloat(str);
Area a=new Area();
a.read(x);
a.calculate();
a.display();
}
}
```

// x=9.2

Output:

```
Enter the radius:
10
Area=314.0
```

Multi Level Inheritance

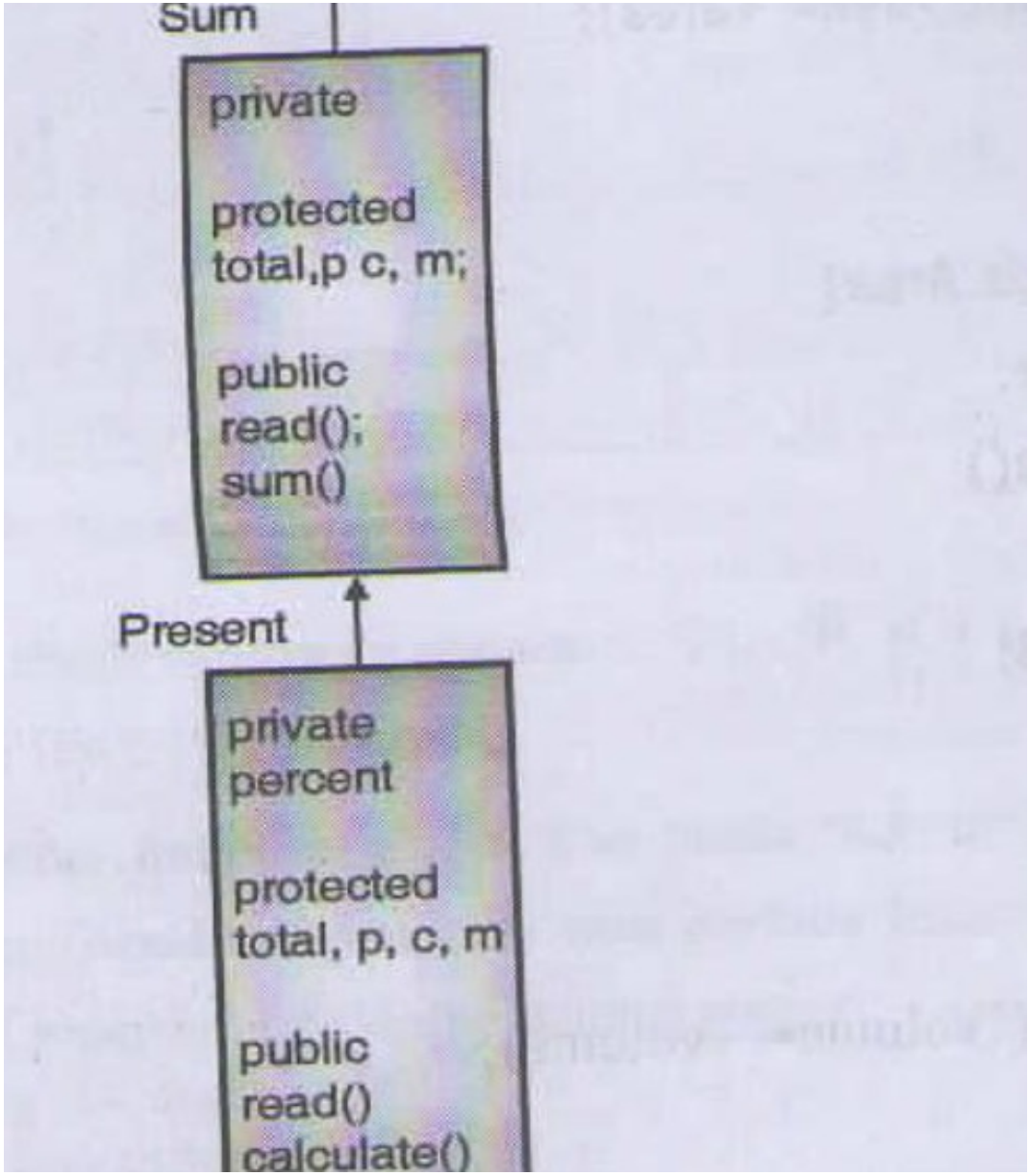
In this case, one class is derived from a class which is derived from another class.

area $= \pi r^2$

//WAP to calculate the volume of the sphere using multilevel inheritance. The base class method will accept the //radius from the user. A class will be derived from the above mentioned class that will have a method to find the area of the circle and another class derived from this will have methods to calculate and display the volume of the sphere.

vol $= \frac{4}{3}\pi r^3$

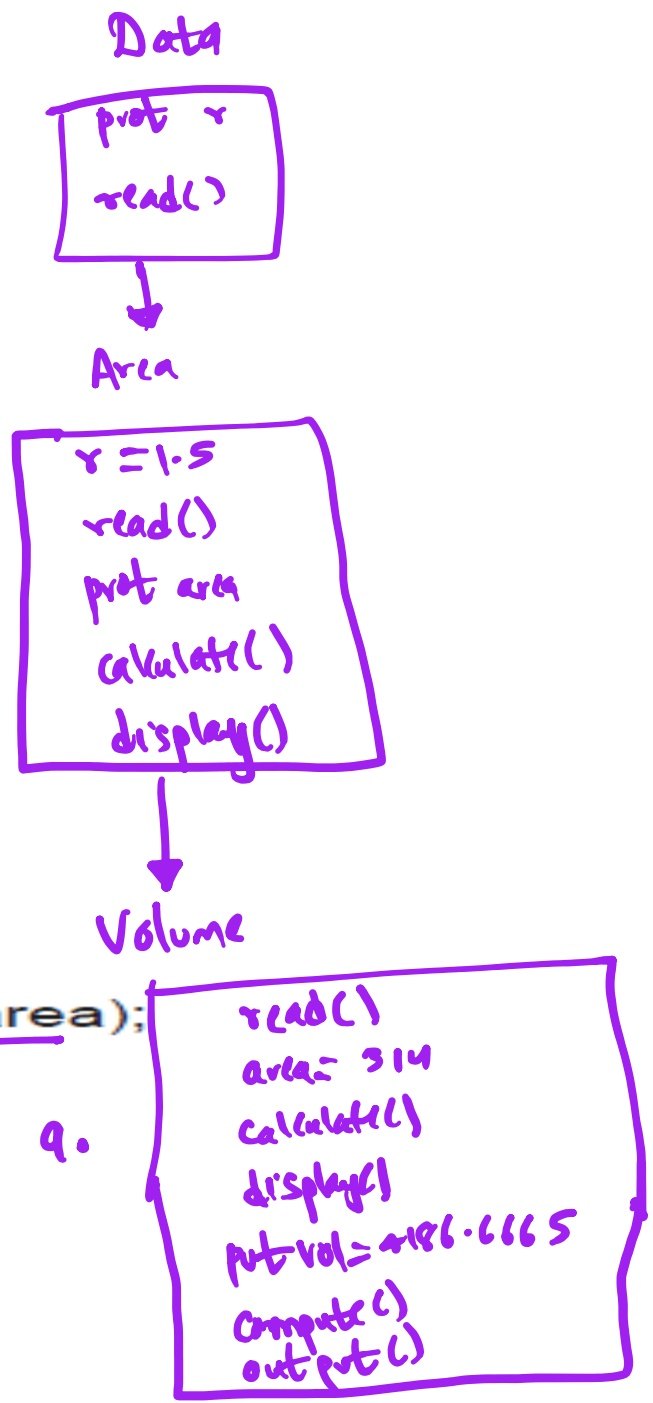
$= \frac{4}{3} r \cdot$ area .....


Sum
private
protected
total,p c, m;
public
read();
sum()
Present
private
percent
protected
total, p, c, m
public
read()
calculate()

```
import java.io.*;
class Data{
protected float r;
public void read(float x)
{
  r=x;
}
}
class Area extends Data
{
protected float area;
public void calculate()
{
   area=3.14f*r*r;
}
public void display()
{
   System.out.println("Area="+area);
}
}
```

1.5

//3.14x1.5x1.5



```
class Volume extends Area{
private float volume;
public void compute()
{
  volume=area*r*4/3;
}
public void output()
{
   System.out.println("Volume="+volume);
}
}
```

```
class Main{
public static void main (String args[]) throws IOException{
float x;
String str;
BufferedReader br = new BufferedReader (new InputStreamReader (System.in));
System.out.println("Enter the radius:");
str=br.readLine();
x=Float.parseFloat(str);
Volume a=new Volume();
a.read(x);
a.calculate();
a.display();
a.compute();
a.output();
}
}
```

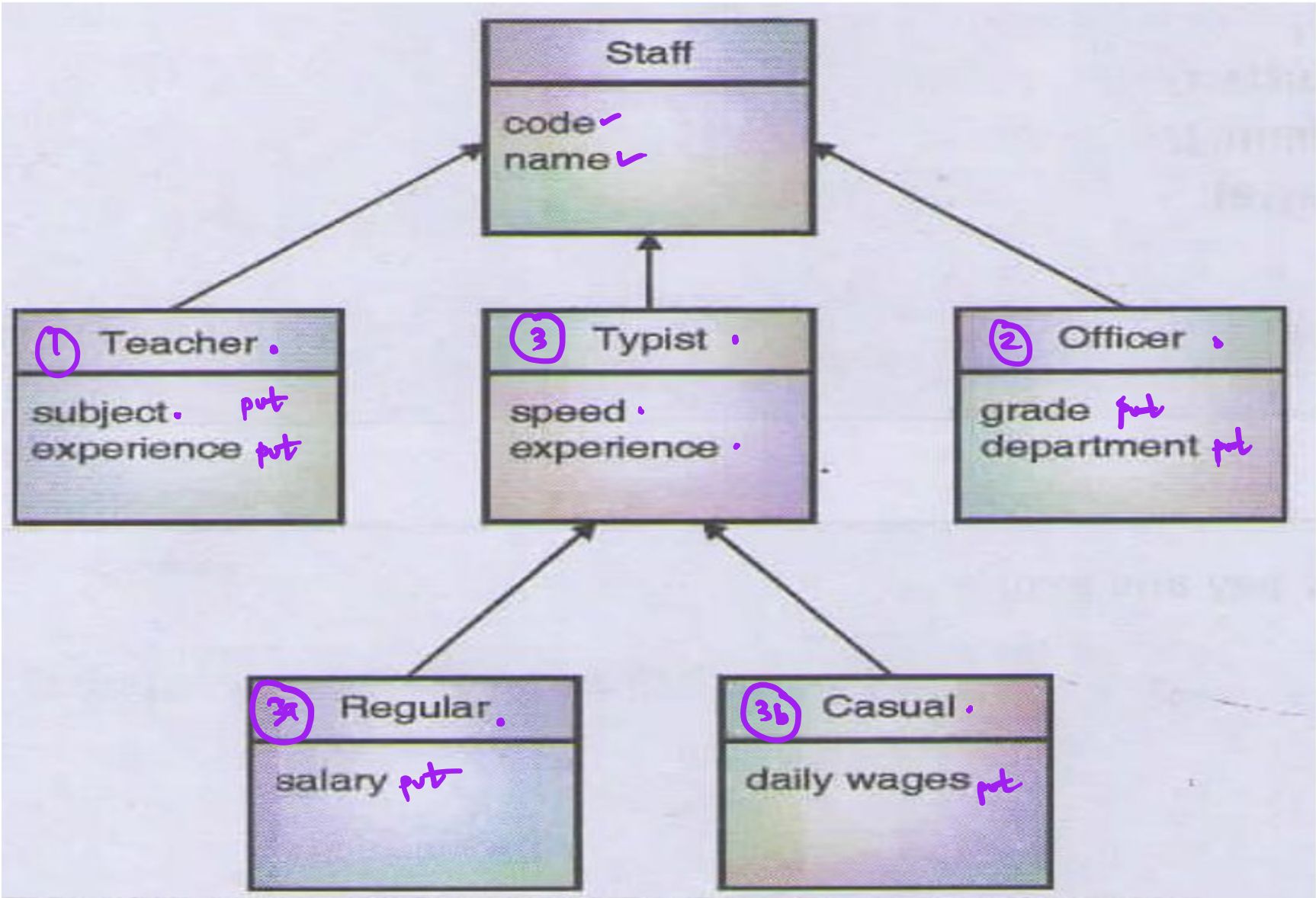
Output:

```
Enter the radius:
10
Area= 314.0
Volume= 4186.6665
```

Hierarchial Inheritance

When multiple classes are derived from a class and further more classes are derived from this derived class, it is called hierarchial inheritance.

//WAP to define following inheritance relationship.



Class Diagram of Hierarchical Inheritance for Program

```
import java.io.*;
class Staff
{
        protected String name;
        protected int code;
}
```

```
class Teacher extends Staff
{
        private String subject;
        private int experience;
        public void read() throws IOException
        {
                BufferedReader br=new BufferedReader(new InputStreamReader(System.in));
                String str;
                System.out.println("Enter name, code, subject and experience of the teacher:");
                name=br.readLine();
                str=br.readLine();
                code=Integer.parseInt(str);
                subject=br.readLine();
                str=br.readLine();
                experience=Integer.parseInt(str);
        }
        public void display()
        {
                System.out.println("Teacher
Details:\nName:"+name+"\nCode:"+code+"\nSubject:"+subject+"\nExperience:"+experience);
        }
}
```

```
class Officer extends Staff
{
        private   String dept;
        private int grade;
        public    void read()throws IOException
        {
                BufferedReader br=new BufferedReader(new InputStreamReader(System.in));
                String str;
                System.out.println("Enter name, code, department and grade of the officer:");
                name=br.readLine();
                str=br.readLine();
                code=Integer.parseInt(str);
                dept=br.readLine();
                str=br.readLine();
                grade=Integer.parseInt(str);
        }
```

```
        public void display()
        {
                System.out.println("Officer
Details:\nName:"+name+"\nCode:"+code+"\nDepartment:"+dept+"\nGrade:"+grade);
        }
}
```

```
class Typist extends Staff
{
        protected int speed,experience;
}
```

```
class Regular extends Typist
{
        private   float salary;
        public void read()throws IOException
        {
                BufferedReader br=new BufferedReader(new InputStreamReader(System.in));
                String str;
                System.out.println("Enter name, code, speed, experience and salary of the regular typist:");
                name=br.readLine();
                str=br.readLine();
                code=Integer.parseInt(str);
                str=br.readLine();
                speed=Integer.parseInt(str);
                str=br.readLine();
                experience=Integer.parseInt(str);
                str=br.readLine();
                salary=Float.parseFloat(str);
        }
        public void display()
        {
                System.out.println("Regular                                                                Typist
Details:\nName:"+name+"\nCode:"+code+"\nSpeed:"+speed+"\nExperience:"+experience+"\nSalary:"+salary);
        }
}
```

```
class Casual extends Typist
{
        private float dailywages;
        public void read()throws IOException
        {
                BufferedReader br=new BufferedReader(new InputStreamReader(System.in));
                String str;
                System.out.println("Enter name, code, speed, experience and dailywages of the Casual
typist:");
                name=br.readLine();
                str=br.readLine();
                code=Integer.parseInt(str);
                str=br.readLine();
                speed=Integer.parseInt(str);
                str=br.readLine();
                experience=Integer.parseInt(str);
                str=br.readLine();
                dailywages=Float.parseFloat(str);
        }
        public void display()
        {
                System.out.println("Casual                                                    Typist
Details:\nName:"+name+"\nCode:"+code+"\nSpeed:"+speed+"\nExperience:"+experience+"\nDaily
Wages:"+dailywages);
        }
}
```

```
class Main
{
        public static void main(String args[])throws IOException
        {
                BufferedReader br=new BufferedReader(new InputStreamReader(System.in));
                String str;
                int choice;
                System.out.println("1. Teacher\n2. Officer\n3. Regular Typist\n4. Casual Typist\nEnter the
choice, whose details you want to enter:");
                str=br.readLine();
                choice=Integer.parseInt(str);
                switch(choice)
                {
                        case 1:Teacher t=new Teacher();
                        t.read();
                        t.display();
                        break;
                        case 2:Officer o=new Officer();
                        o.read();
                        o.display();
                        break;
                        case 3:Regular r=new Regular();
                        r.read();
```

```
                r.display();
                break;
                case 4:Casual c=new Casual();
                c.read();
                c.display();
                break;
                default:System.out.println("Invalid choice");
            }
        }
}
```

Output:
1 .Teacher
2. Officer
3. Regular Typist
4. Casual Typist
Enter the choice, whose details you want to enter:
2
Enter name, code, department and grade of the officer:
Ajay
325
Accounts
1
Officer Details:
Name: Ajay
Code: 325
Department: Accounts
Grade: 1