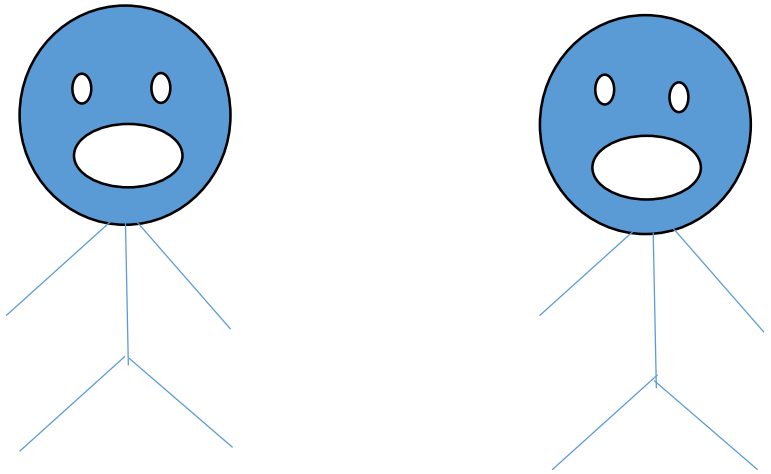
C++
CLASS

P.P.Krishnaraj
RSET

CLASS

Class---Collection of related data and functions

Object----Can be real world entity or variable



Objects : Real world entity

Common properties		
	Sleep()	Eye color
	Eat()	Hair color
	Walk()	Height
	Run()	Weight

Class: we can keep this in class

Describes common properties and common actions of an object

Object is a variable of type class

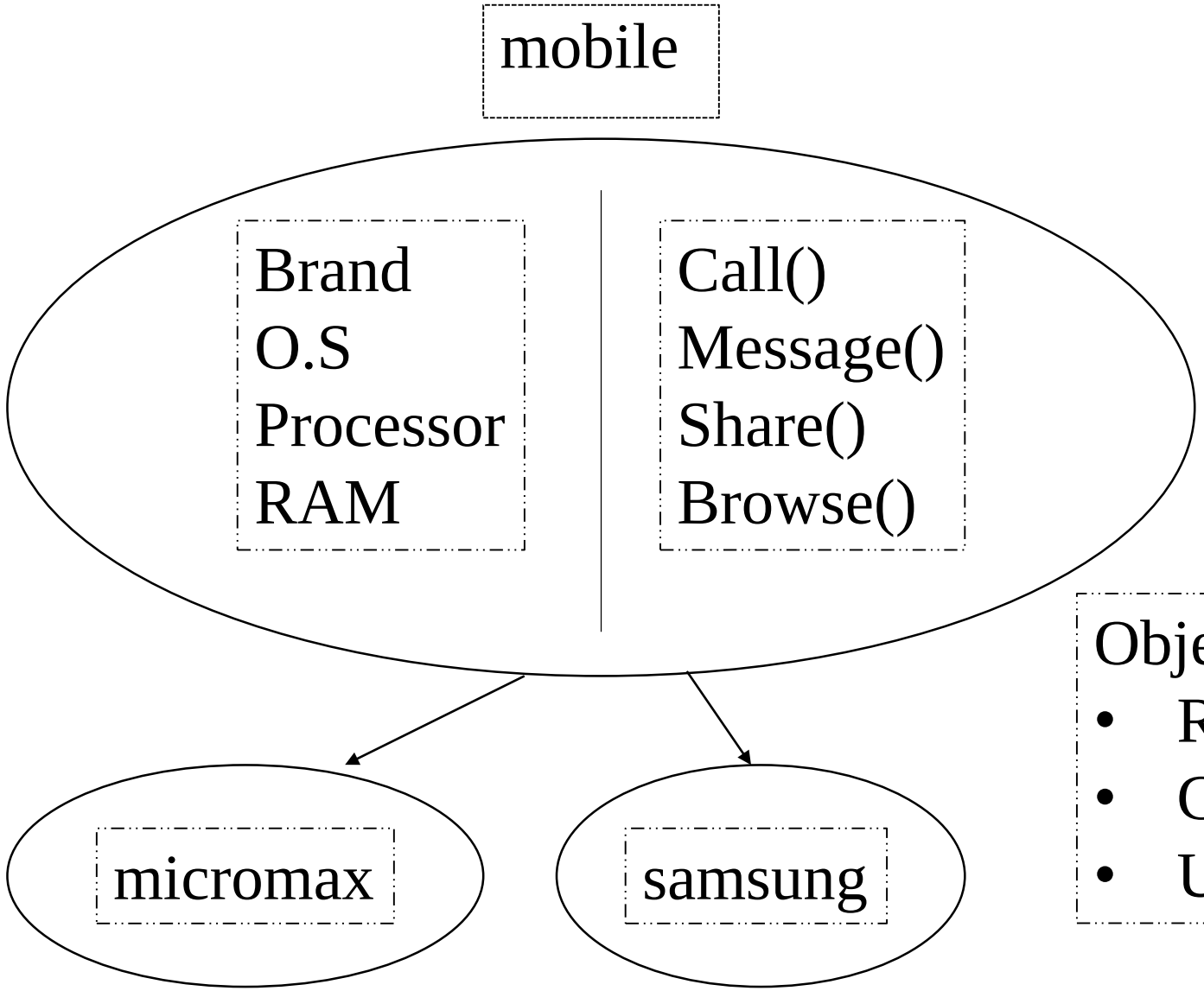
To make a class , we use keyword class

```
Class person
{
----------;
----------;
----------;
}
```

Properties i.e datas + functions

type

```
person p;
int a;
```

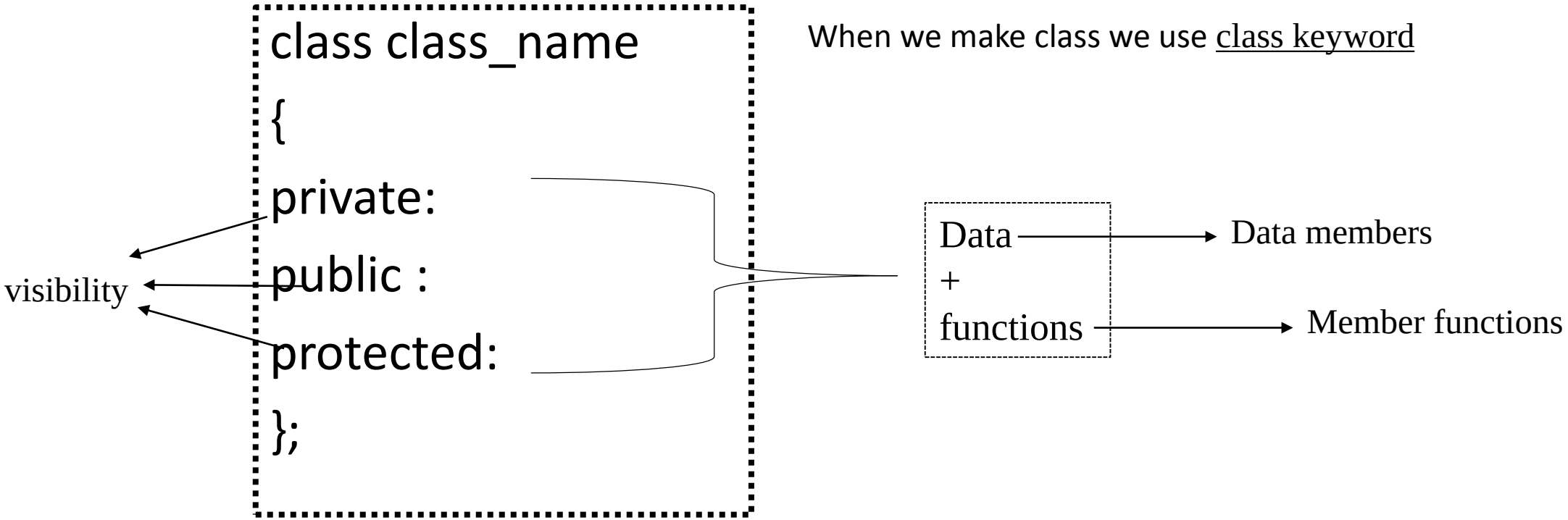


CLASS
This is a common list for all objects

Objects
- Real world entity
- Class type variable
- Unique characteristics

NOTE: Variables can be different, so class describes objects properties and behaivour

How to define class and create objects



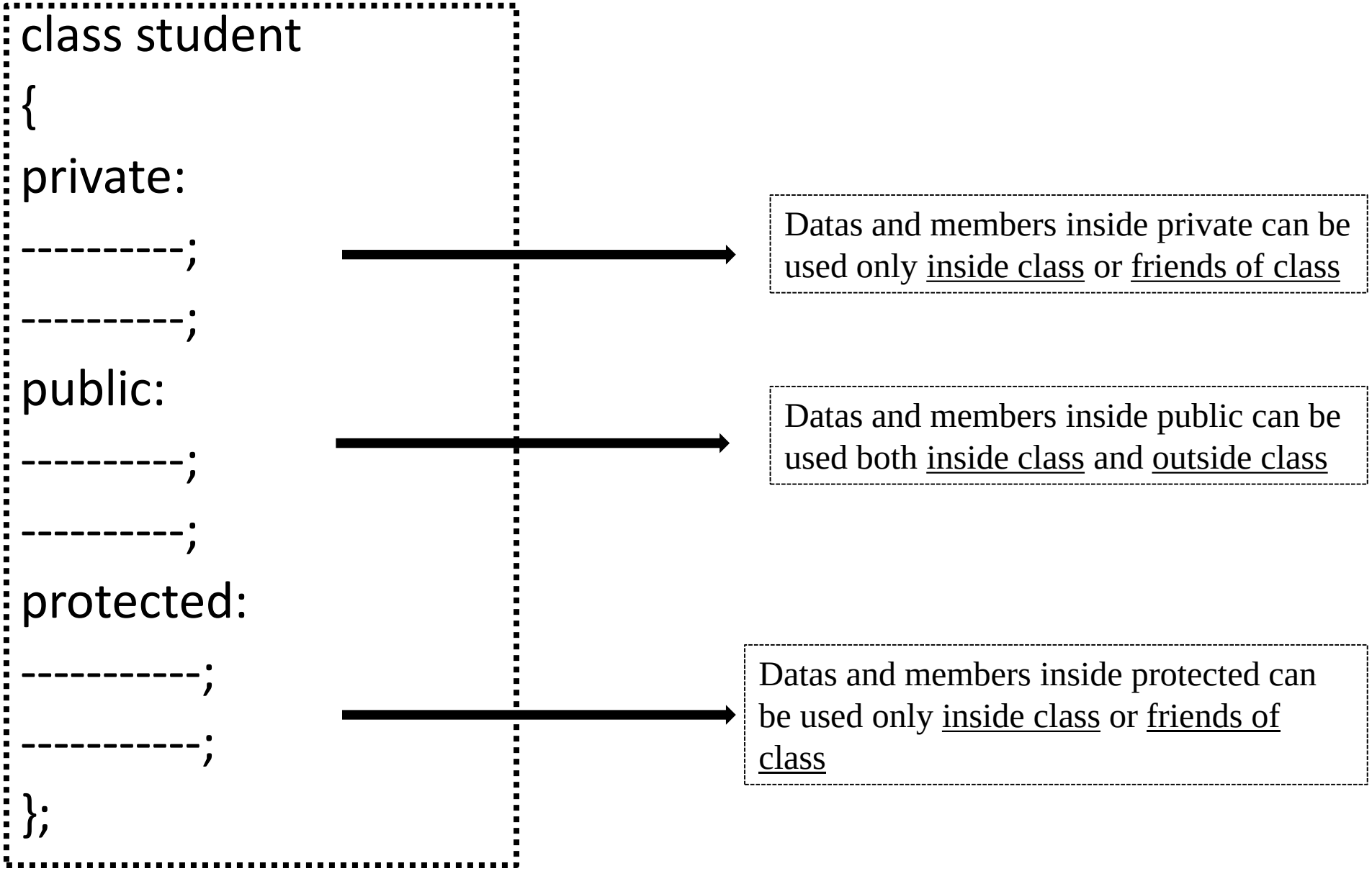
NOTE: usually data is private , functions is written in public

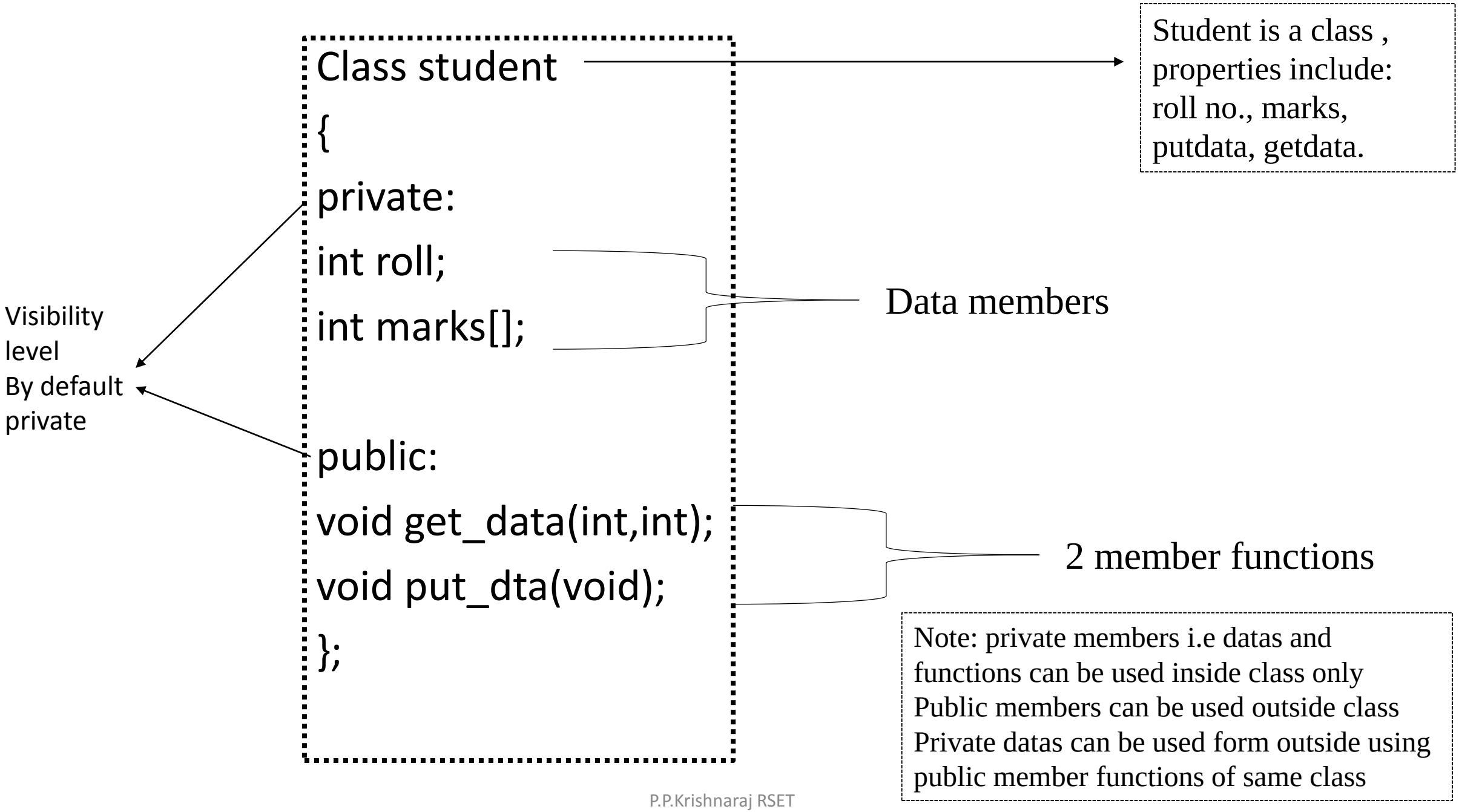
Access specifiers

Private:
Public:
Protected:

```
class class_name
{
private:
public:
protected:
};
```

Used for access control or set boundaries for class members i.e upto where these members are accessible(inside or outside class)


class student
{
private:
---------;
---------;
public:
---------;
---------;
protected:
----------;
----------;
};
Datas and members inside private can be used only inside class or friends of class
Datas and members inside public can be used both inside class and outside class
Datas and members inside protected can be used only inside class or friends of class


Class student
{
private:
int roll;
int marks[];
public:
void get_data(int,int);
void put_dta(void);
};
Student is a class , properties include: roll no., marks, putdata, getdata.
Data members
2 member functions
Visibility level By default private
Note: private members i.e datas and functions can be used inside class only
Public members can be used outside class
Private datas can be used form outside using public member functions of same class

Member function definition

1.Inside class definition

```
Class student
{
private:
int roll;
int marks[];

public:
void get_data(int,int);

void put_data(void);
{
cout<<“roll represents roll no.”;
cout<<“cut off mark is 15”;
}
};
```

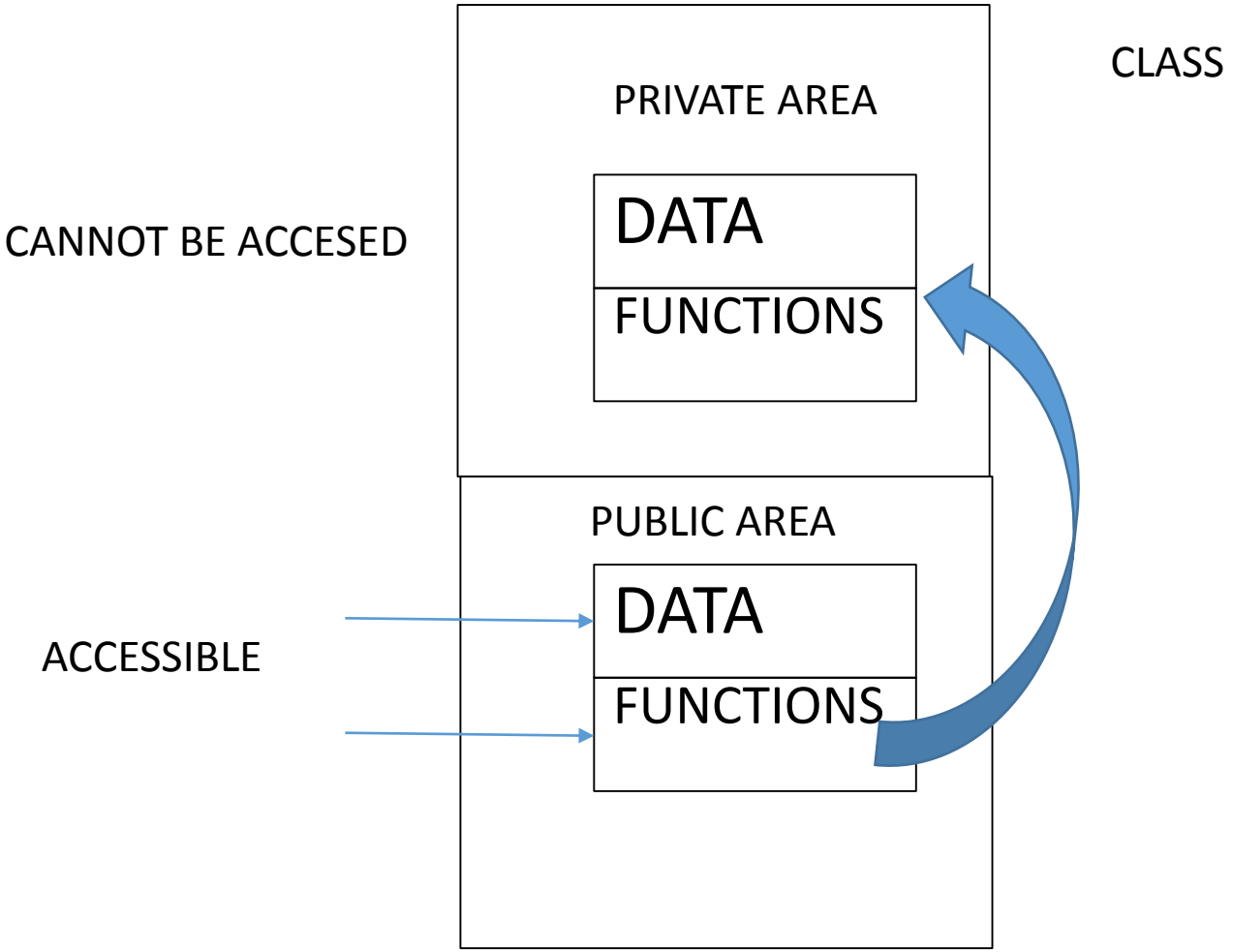
Declared not defined

Defined

2. Outside class definition

```
Return_type class_name :: function_name(parameters);
{
Body;
}
```

```
Class student
{
private:
int roll;
int marks;

public:
void get_data(int,int);
};
void student :: get_data(int x,int y);
{
roll=x;
marks=y;
}
```


CLASS
PRIVATE AREA
DATA
FUNCTIONS
CANNOT BE ACCESED
PUBLIC AREA
DATA
FUNCTIONS
ACCESSIBLE

OBJECT

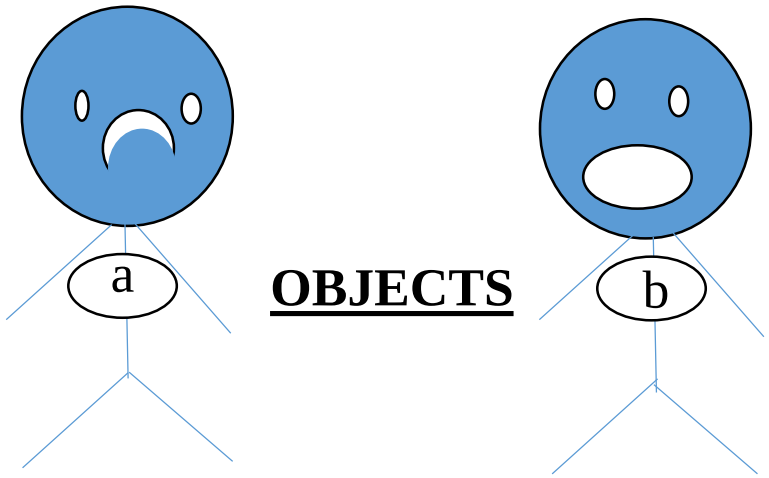
The declaration of object is same as declaration of variables of basic data types. Defining object of class data type is known as class installation.

class_name object_name;

student p,q,*r; //declaration of object or class type variable.

An object is an abstract having following properties.

- It is individual
- It points to a thing, either physical or logical identifiable by user.
- It holds data as well as operation methods that holds data.
- Its scope is limited to the block in which it is defined.




```
void main()
{
student a,b;
a.marks=25;
b.marks=75;
a.get_data();
}
```

CLASS

marks	get_data()
percentage	put_data()
cuttoff	percentage()

```
Class student
{
-----;
-----;
-----;
-----;
}
```

ACCESSING CLASS MEMBERS

The object can access the public member variables and functions of a class by using operator dot (.) and arrow (->).

The syntax is:

```
a.get_data();
```

Where a is an object and get_data is a member function.

```
c->get_data();
```

*c is a pointer to class item, therefore (->) operator is used.

```
class student      //class declaration
{
int marks;
float percentage;
int cutoff;
};

void main()
{
student a,b,c;
clrscr()
a.marks=25;
a.percentage=75;
a.cuttoff=33.3;
}
```
NOTE:the above program will generated error message

The public keyword

The member variables and the function declared followed by the keyword public can be accessed directly by the object.

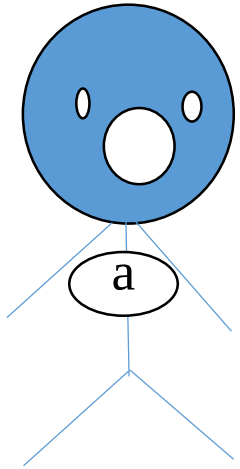
The declaration can be done as below:

```
class student                    //class declaration
{
public:                                  //public section begins
int marks;
float percentage;
int cuttoff;
};
```

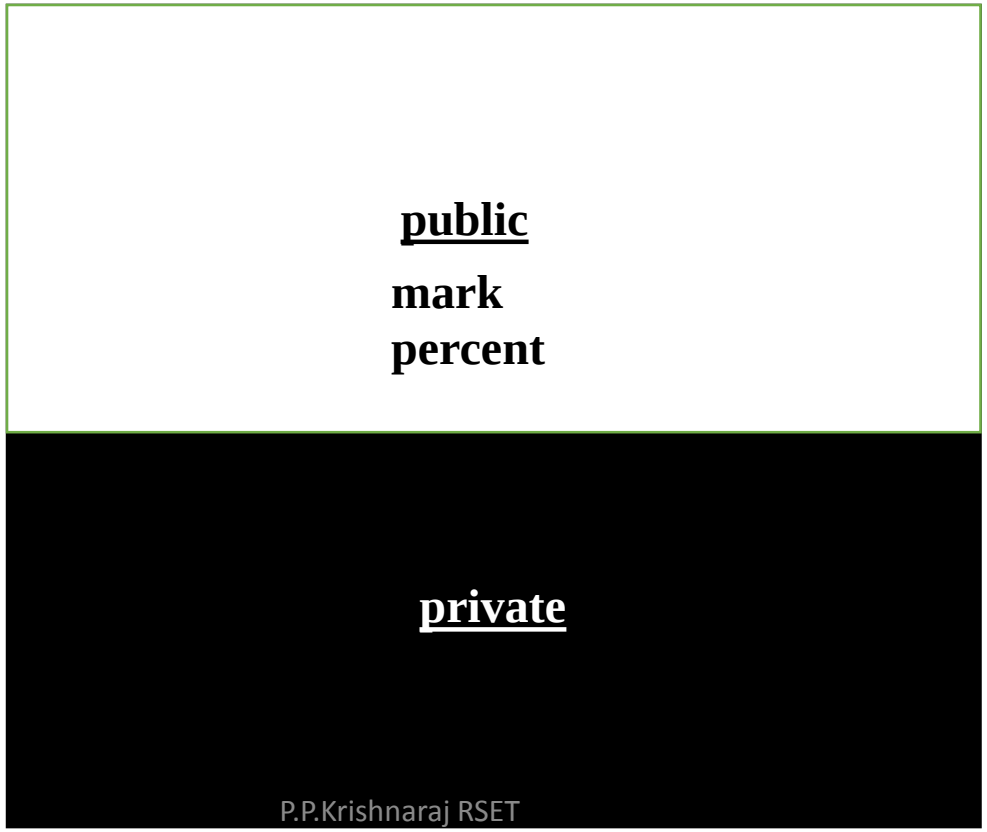
```
#include<iostream.h>
#include<conio.h>
class student                    //class declaration
{
public:                          //public section begins
int marks;
float percentage;
int cuttoff;
};
int main()
{clrscr();
student a;                       //object declaration
a.marks=75;                      //data member initialisation
a.percentage=75;
a.cuttoff=33;
cout<<“marks=“<<a.marks;
cout<<“percentage=“<<a.percentage;
cout<<“cuttoff=“<<a.cuttoff;
}
```

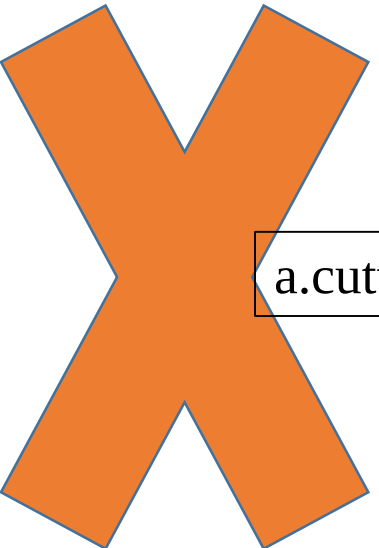
NOTE: The object a of the class student accesses the member variables directly. The member variables are initialised and the variables are initialised on the screen.

a.mark=25;
a.percent=75;



OBJECTS





a.cuttoff=25; //not possible

```
Class student
{
public :
int mark;
int percent;

private :
cuttoff;
}
```

The private keyword

The private keyword is use to prevent any direct access to member variables or function by the object.

```
class student
{
private:
int marks;
float percentage;
int cuttoff;
};
int main()
{
clrscr();
student a;                    //object declaration
a.marks=75;        // not accessible (member initialisation)
a.percentage=75; // not accessible
}
```

ERROR: ‘student::marks’ is not accessible
‘student::percentage’ is not accessible

NOTE: private members are not accessible by the object directly. To access the private members of the class ,member functions of same class are used. The member functions must be declared in public section in class.

Defining member functions

- The member function must be declared inside the class. They can be defined in a)private or public section b) inside or outside class
- The member function defined inside class are treated as inline functions. if the member function is small then it should be defined inside class otherwise it should be defined outside class.
- If the member function is defined outside class its prototype declaration must be done inside the class. while defining the function ,scope access operator and class name should precede the function name.

(1) Member function inside the class in public

Program to access private data members of a class using member functions

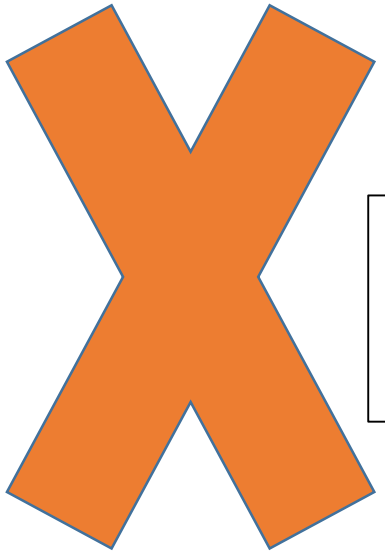
```
class student
{
private:
int marks;
float percentage;
int cuttoff;

public:
void get_data()
{
marks=75;
percentage=75;
cuttoff=33;
cout<<“marks=“<<marks;
}
};
```

private data members

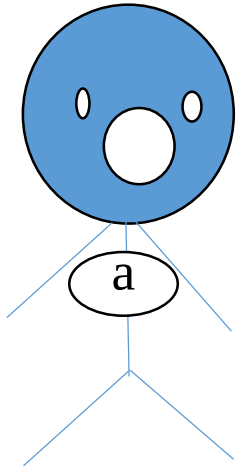
Member function body

```
int main()
{
clrscr();
student a;          //object declaration
a.get_data();       //function call
return 0;
}
```

(2) Member function inside the class in private



~~a.marks=25;~~
~~a.data();~~



OBJECTS

a.details();

public

details()

private

mark
data()

Private data cant be accessed directly

```
class student
{
private :
int mark;
void data()
{
marks=25;
}
public :
void details()
{
data();
}
```

Program to access private member function of a class using public member functions

```
class student
{
private:
int marks;
float percentage;
int cuttoff;
void get_data()
{
marks=75;
percentage=75;
cuttoff=33;
}
public:
void put_data();          //public member f(n)
{
get_data();               //call to private member f(n)
cout<<“marks=“<<marks;
}
};
int main()
{
clrscr();
student a;                //object declaration
                          /*a.get_data(); not accessible*/
a.put_data();             //call to public member f(n)
return 0;
}
```

private data members

private member function body

(3) Member function outside the class

To define a function outside the class:

- **Prototype declaration must be declared inside the class**
- **Function name must be preceded by class name and its return type separated by scope access operator**

```
#include<iostream.h>
#include<conio.h>
class sum
{
int a,b,c;
public:
void getdata(void);
void putdata(void);
};
void sum::getdata(void)
{
cout<<"enter the value of a and b";
cin>>a>>b;
}
void sum::putdata(void)
{
c=a+b;
cout<<"addition of values"<<a<<"and"<<b<<"="<<c;
}
```

Private data members

Function prototype declaration

Function definition outside class

Function definition outside class

```
int main()
{
sum obj;   //object declaration
obj.getdata();//call
obj.putdata();
getch();
return 0;
}
```

Q) Declaring outside function inline.

Write a program to declare outside function inline

```
#include<iostream.h>
#include<conio.h>
class sum
{
int a,b,c;
public:
void getdata(void);
};
inline void sum::get_data(void)
{
cout<<“enter the value of a and b”;
cin>>a>>b;
}
```

Function definition outside

```
int main()
{
sum obj;   //object declaration
obj.get_data();//call
getch();
return 0;
}
```

Q) Write a program to declare class with private, public and protected sections. Declare object and access data elements of these different sections.

```
#include<stdio.h>
#include<conio.h>
class access
{
private:
int p;
void getp()
{
cout<<“enter the value of p”;
cin>>p;
}
/*continued*/
```

```
protected :
int m;
void getm()
{
cout<<“enter the value of m”;
cin>>m;
}
public :
int h;
void geth()
{
getp();
getm();
}
};
```

```
void main()
{
clrscr();
access a;//object declaration
        //a.p=2; not possible
        //a.getp() not possible
        //a.m=5; not possible
        //a.getm(); not possible
a.h=4;//public data member
a.geth();//public member function
}
```

DATA HIDING AND ABSTRACTION

- Data hiding is also known as encapsulation. An encapsulated object is often called as abstract data type.
- Why encapsulation????
- Programmer often makes various mistakes and data gets changed accidently.
- So to protect data we need to create a secure and impassable wall, data hiding is nothing but making data variables of class private.
- Private data cannot be accessed by object directly. Object using public member function of same class can access the private data of the class.

Q) Write a program to calculate simple interest. Hide the data elements of the class using private keyword

```
#include<stdio.h>
#include<conio.h>
class interest
{
private :
float p;
float r;
float t;
float I;
float A
public :
void calculate()
{
cout<<“principal=“;
cin>> p;
cout<<“rate=“;
cin>>r;
cout<<“no. of years”;
cin>>t;
I=(p*r*t)/100;
A=I+p;
cout<<“total amount=“<<A;
}
};
int main()
{
clrscr();
interest r;
r.calculate();
return 0;
}
```

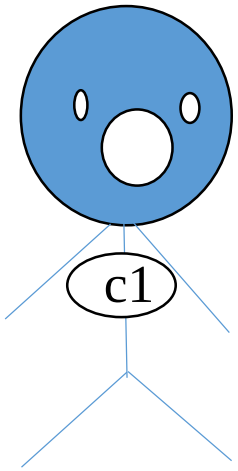
Passing object as function argument in C++

$$\text{Comp1} = 2 + 4i$$
$$\text{Comp2} = 3 + 7i$$

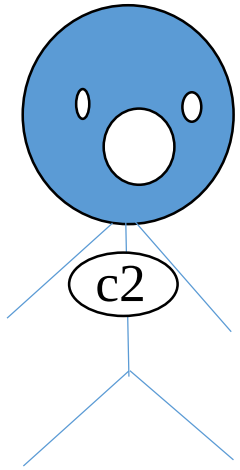
$$\text{Total} = 5 + 11i$$

Without using classes and objects

```
int main()
{
int real1,imag1, real2,imag2,total_real,total_imag;
cout<<“enter the real and imaginary part of comp1”;
cin>>real1>>imag1;
cout<<“enter the real and imaginary part of comp1”;
cin>>real2>>imag2;
Total_real=real1+real2;
Total_imag=imag1+imag2;
cout<<“sum=”<<Total_real<<“+”<<Total_imag<<“i”;
}
```



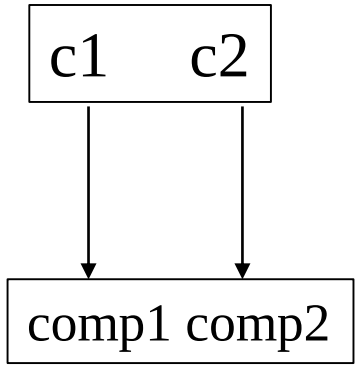
c1.read();



c2.read();

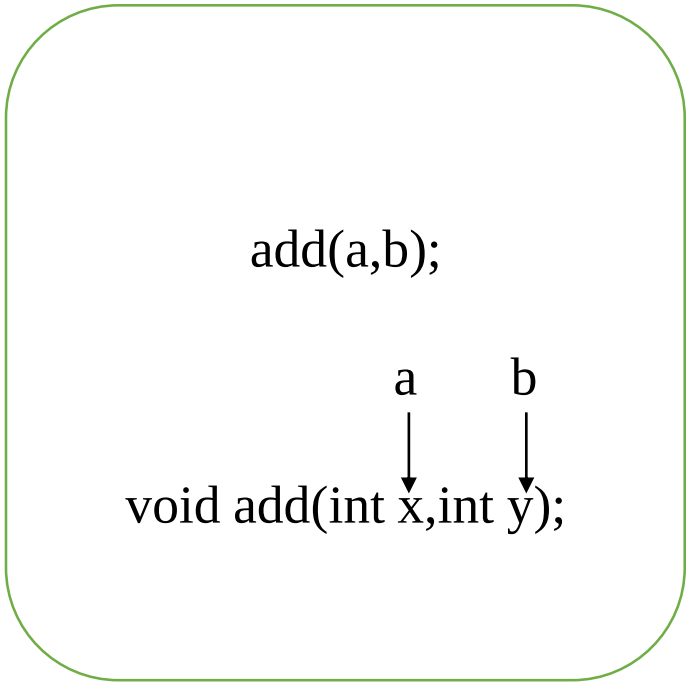
c3.sum(c1,c2);

```
class c_name
int a;
public :
void read()
{
cin>>a;
}
void sum(c_name comp1 ,c_name
comp2)
{
S=comp1.a+comp2.a;
}
```



```
class complex
{
private:
int real, imag;
public :
void readData()
{
cout << "Enter real and imaginary number respectively";
cin >> real >> imag;
 }
void addComplexNumbers(Complex comp1, Complex comp2)
{
real=comp1.real+comp2.real;
imag=comp1.imag+comp2.imag;
 }
void displaySum()
 {
 cout << "Sum = " << real<< "+" << imag << "i"; }
};
```

```
int main()
{
complex c1,c2,c3;
c1.readData();
c2.readData();
c3.addComplexNumbers(c1, c2);
c3.displaySum();
return 0;
}
```



Q) Time 1= 3 hrs 40 minutes

Time 2= 4 hrs 30 minutes

Find total time…?

```
#include<iostream.h>
#include<conio.h>
class time
{ int h,m;
public :
void get_time(int,int);
void put_time(void);
void sum(time,time);      };

void time::get_time(int X,int Y)
{  h=X;
   m=Y;       }
void time::put_time(void)
{
cout<<“hours“<<h;
cout<<“minutes=“<<m;   }
void time::sum(time ti1,time ti2)
{
m=ti1.m+ti2.m;
h=m/60;
h=ti1.h+ti2.h;
}
int main()
{
time t1,t2,t3;
t1.get_time(3,40);
t2.get_time(4,30);
t3.sum(t1,t2);
t1.put_time();
t2.put_time();
t3.put_time();
getch();
return 0;
}
```

```
#include<iostream.h>
#include<conio.h>
class weight
{
int kilogram, gram;
public:
void getdata();
void sum_weight(weight,weight);
};
void weight::getdata()
{
cout<<“kilogram”:
cin>>kilogram;
cout<<“gram”;
cin>>gram;
}
void weight::putdata()
{
cout<<kilogram<<“kgs”<<“and”<<gram<<“gs”;
}
void::weight::sum_weight(weight w1,weight w2)
{
gram=w1.gram+w2.gram;
kilogram=gram/1000;
gram=gram%1000;
kilogram+=w1.kilogram+w2.kilogram;
}

int main()
{
weight w1,w2,w3;
cout<<“enter weight in kilograms and grams”;
cout<<“enter weight 1”;
w1.getdata();
cout<<“enter weight 2”;
w2.getdata();
w3.sum_weight(w1,w2);
cout<<“weight 1=“<<w1.putdata();
cout<<“weight 2=“<<w2.putdata();
cout<<“total weight =“<<w3.putdata();
return 0;
}
```