Section 2: Lecture 5

Introduction

- Memory Allocation
- Arrays with in the class
- Static Data Members
- Friend Functions

Class implementation

```
class item{
int number;
float cost;
public :
void getdata();
void putdata()
{
 cout<<“number:”<<number<<“\n;
  cout<<“cost:”<<cost<<“\n;
}
};
void item::getdata(int a,float b)
{number=a;
cost=b;
}
```

```
void main()
{
Item x;

cout<<“object  x” <<\n;
x.getdata(100,299.95);
x.putdata();
item y;

cout<<object y”<<\n;
y.getdata(200,175.50);
y.putdata();
}
```

```
        Output:-object x
                number:100
                cost:299.95

                object y
                number:200
                cost:175.5
```

Private member fuction

```
class sample
{
int m;
void read(void);
public:
void update(void);
void write (void);
};
```

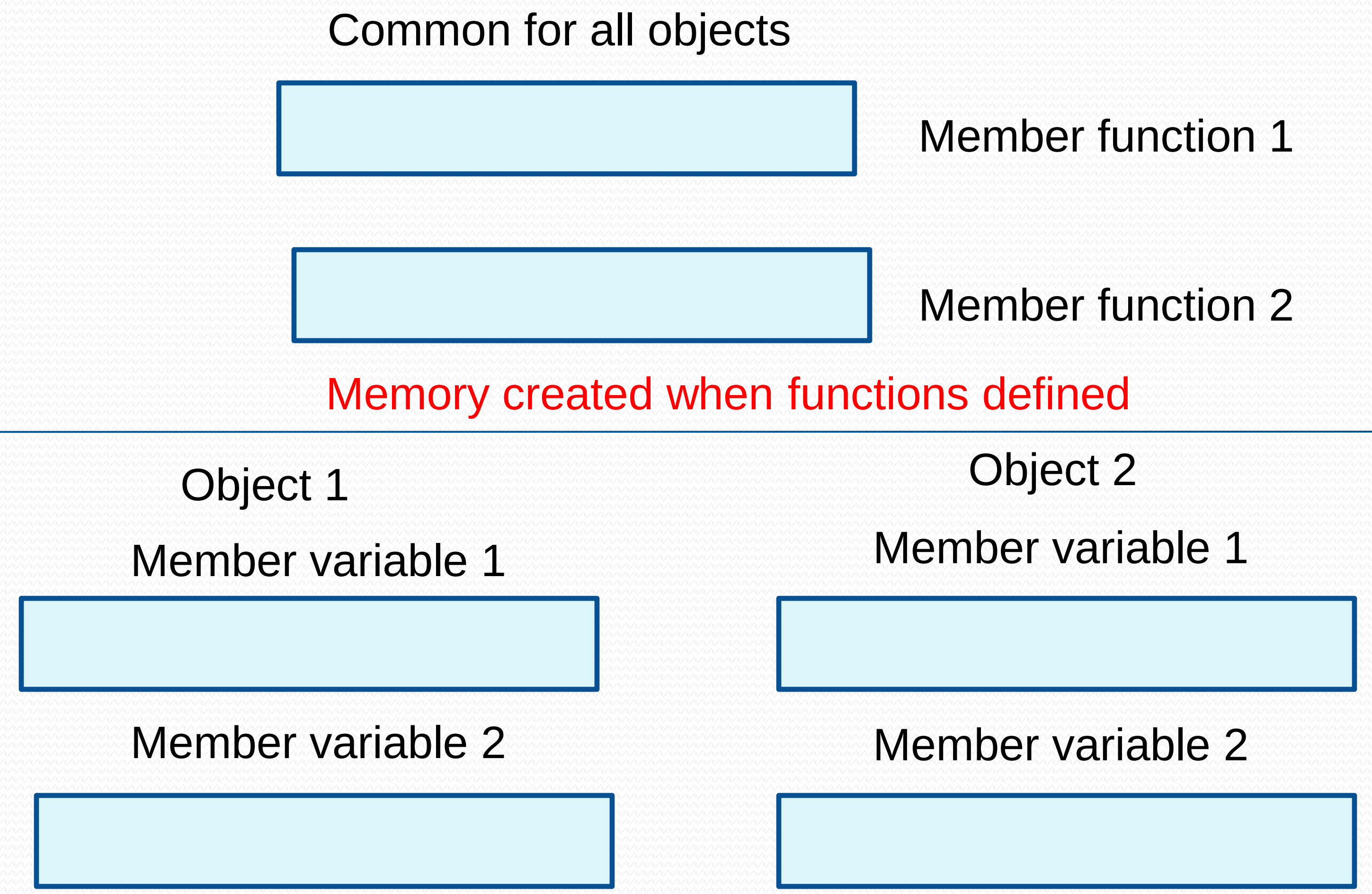
```
s1.read();    //won't work;
               object cannot access private member


void sample ::update(void)
{
read();                      //simple call ;
                            no object used
}
```

Array within a class

```
const int size=10;
class array
{
  int a [size];     //a is int type array
  public:
  void setval(void);
  void display(void);
};
```

Memory allocation for object



Memory created when object defined

Fig: Object of memory

Static data members

Special properties:

1. It is initialized to 0 when the first object of its class is created.
2. Only one copy of that member is created for the entire class and is shared by all the objects of that class ,no matter how many objects are created.
3. It is visible only within the class ,but its lifetime is the entire program.

Static Data Member

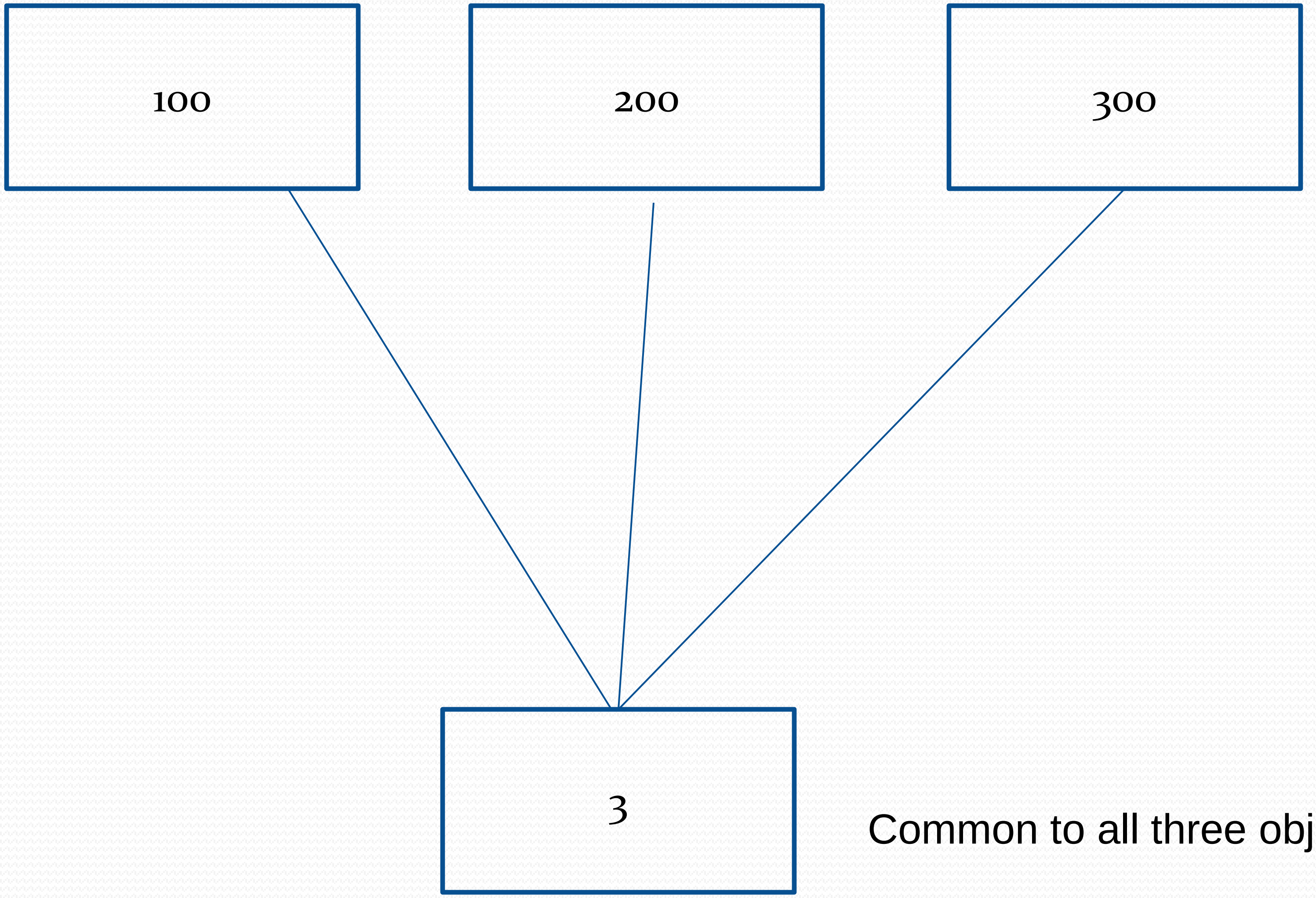
```
class item
{
static int count;
int number;

public:
void getdata(int a)
{
number=a;
count++;
}
void getcount(void)
{cout<<“count: “;
cout<<count<<“\n”;}
};
int item::count;  //defination of static
                         member
```

```
 void  main()
{
Item a,b,c;
a.getcount();
b.getcount();
c.getcount();

a.getdata(100);
b.getdata(200);
c.getdata(300);
Cout<<“after reading
data”;
a.getcount();
b.getcount();
c.getcount();
}
```

Sharing of a static data member



Common to all three object

Static Member Function

Properties:-

1 a static function can have access to only static members (function and variables) declared in the same class.

2 a static member function can be called using the class name (instead of its objects)

As follows:

```
class-name::function-name;
```

Static Member Function

```
class test
{
 int code
 static int count;
 public:
 void setcode(void) {
code= ++count;}
void showcode(void){
cout<<“object number:”<<code<<“\n”;}
static void showcount(void){
 cout<< “cout”<<cout<<“\n”;}
};
```

```
void  main()
test t1,t2;
t1.setcode();
t1.setcode();

test::showcount();
test t3;
t3.testcode();
test::showcount();
t1.showcode();
t2.showcode();
t3.showcode();
}
```

Out put-

```
Count:2
Count:3
Object number:1
Object number:2
Object number:3
```

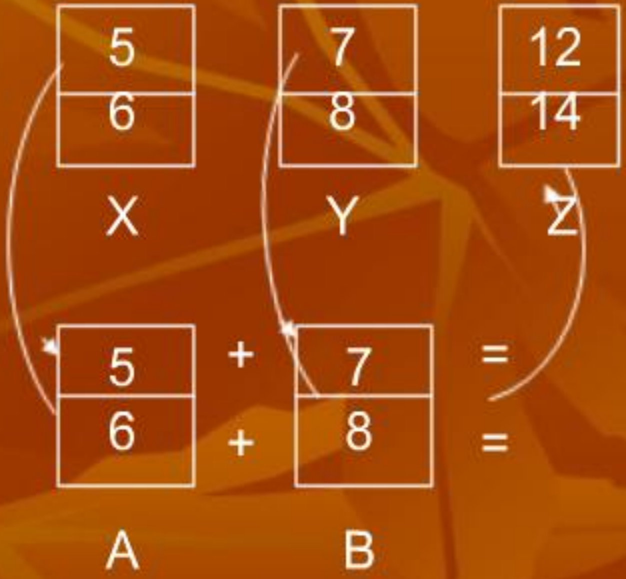
Passing Object

```
#include<iostream.h>
class Complex
{
float real, imag;
public:
void getdata( );
void putdata( );
void sum (Complex A, Complex B);
};
void Complex : : getdata( )
{
cout<<"enter real part:";
cin>>real;
cout<<"enter imaginary part:";
cin>>imag;
}
void Complex : : putdata( )
{
if (imag>=0)
cout<<real<<"+"<<imag<<"i";
else
cout<<real<<imag<<"i";
}
```

```
void complex : : sum ( Complex A, Complex B)
{
real = A.real + B.real;
imag= A.imag + B.imag;
}

void main( )
{
Complex X,Y,Z;
X.getdata( );
Y.getdata( );
Z.sum(X,Y);
Z.putdata( );
}
```

```
12 + 14 i
```



Returning Object

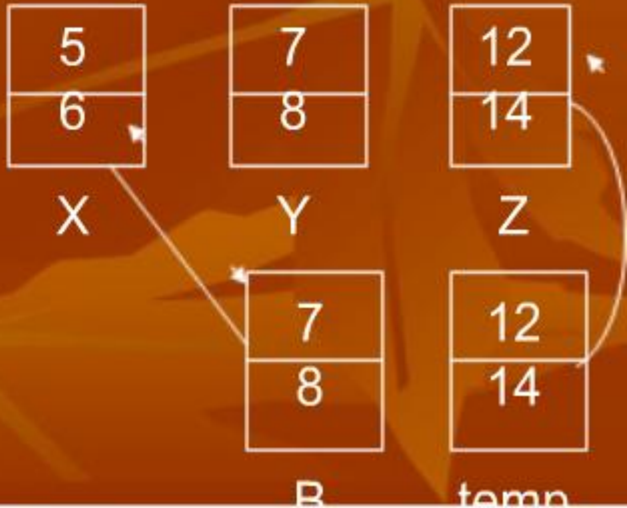
```
#include<iostream.h>
class Complex
{
float real, imag;
public:
void getdata( );
void putdata( );
Complex sum (Complex B);
};
void Complex : : getdata( )
{
cout<<"enter real part:";
cin>>real;
cout<<"enter imaginary part:";
cin>>imag;
}
void Complex : : putdata( )
{
if (imag>=0)
cout<<real<<"+"<<imag<<"i";
else
cout<<real<<imag<<"i";
}
Complex Complex : : sum (Complex B)
{
Complex temp;
temp.real=real + B.real;
temp.imag= imag + B.imag;
return temp;
}
void main ( )
{
Complex X, Y, Z;
X.Getdata( );
Y. getdata( );
Z= X.sum (Y);
Z.putdata( );
}
```

Returning Object

```
#include<iostream.h>
class Complex
{
float real, imag;
public:
void getdata( );
void putdata( );
Complex sum (Complex B);
};
void Complex : : getdata( )
{
cout<<"enter real part:";
cin>>real;
cout<<"enter imaginary part:";
cin>>imag;
}
void Complex : : putdata( )
{
if (imag>=0)
cout<<real<<"+"<<imag<<"i";
else
cout<<real<<imag<<"i";
}
```

```
Complex Complex : : sum (Complex B)
{
Complex temp;
temp.real=real + B.real;
temp.imag= imag + B.imag;
return temp;
}
void main ( )
{
Complex X, Y, Z;
X.Getdata( );
Y. getdata( );
Z= X.sum (Y);
Z.putdata( );
}
```

12 + 14 i



A member function of one class can be friend of another class

```
Class x
{
...
...
Int fun1();                    //member function of x
.....
};

Class y
{
.....
.....
Friend int x::fun1();                    //fun1() of x is friend of y
...
}
```