ABOUT CLASS AND OBJECT

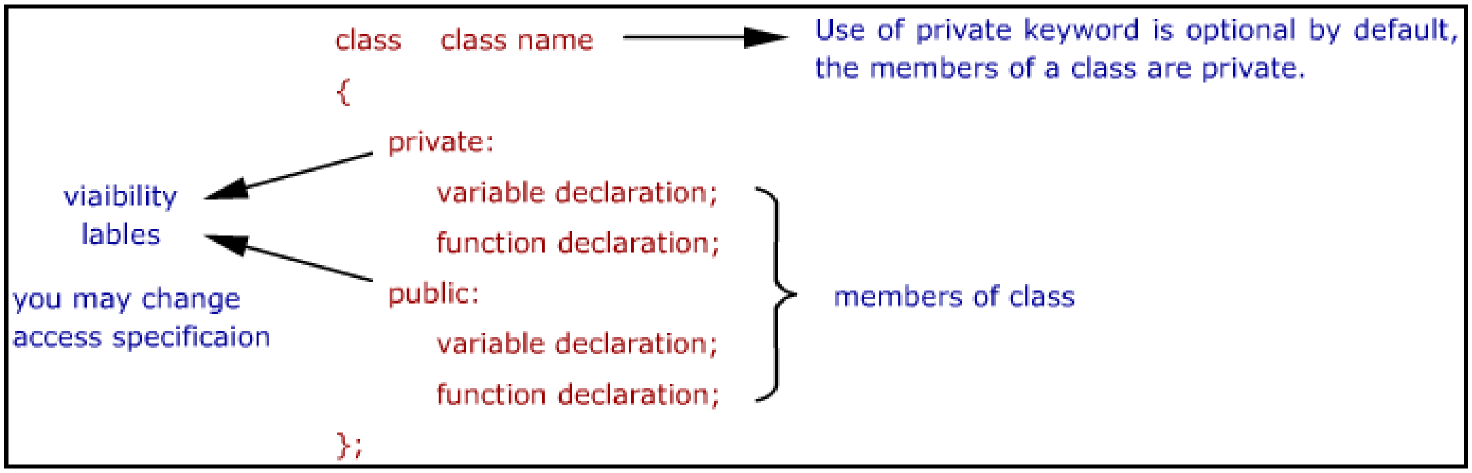
Class

A class is a way to bind the data and its associated functions together. If allows the data to be hidden, if necessary from external use. When defining a class, we are creating a new abstract data type that can be treated like any other built in data type generally, a class specification has two parts.

(a) **Class declaration:** describe the type and scope of its members.

(b) **Class function definition:** describe how the class functions are implemented.

The general from of a class declaration is



Private members: can be accessed only from with in class.

Public members: can be accessed from outside the class also

Variables declared inside the class - data members

Functions declared inside the class - member function

Only the member functions can have access to the private data members and private functions.

However, the public members can be accessed from outside the class

```
class item        ──────►  This name now becomes a new type identifier that
                           can be used to declare instances of that class type.
{
          int no.;      }  private by default
          float cost;   }
   public
          void get data (int a, float b);   }  fn declaration using prototype
          void put data (void)              }
};
```

Function prototype is a declaration statement in the program and is of the following form:

```
type fn name (arg. list);
                  │
                  └──────────►  contains the type and names of arguments that
                                must be passed to the function.

e.g. float volume (int x, float y, float z);
     float volume (int x, float y, z);          // illegal
or   float volume (int, float, float);          // acceptable at this stage, compiler only checks
                                                for type of arguments when the function is called
```

The f^n definition, names are required because the arguments must be referenced inside the function.

Creating Objects

The declaration of item does not define any objects of item but only specifies what they will contain. Once a class has been declared, we can create variables of that type by using the class name.

```
e.g. item x;
          │
          └──────►  class variable is known as object
```

We can also declare more than one object in one statement.

```
item x, y, z;
```

Objects can also be created when a class is defined by placing their names immediately after the closing brace i.e.

```
class item
{
   ...........
   ...........
} x, y;
```

Accessing Class Members

The private data of a class can be accessed only through the member functions of that class.

For calling a member function

object name f^{n} name (actual arguments);

```
x.get data (100, 75.5);          (Valid, assign 100 and 75.5 for cost)
x.put data ();
getdata (100, 75.5);             // illegal
x.number = 100;                  // illegal
```

A variable declared as public can be accessed by the objects directly e.g.

```
class xyz
{
            int x;
            int y;
    public int z;
};
      .........
      .........
    xyz p;
    p . x = 10;                          // error, x is private
    p . z = 10;                         // ok, z is public
```

But use of data in this manner defeats the idea of data hiding and therefore should be avoided.

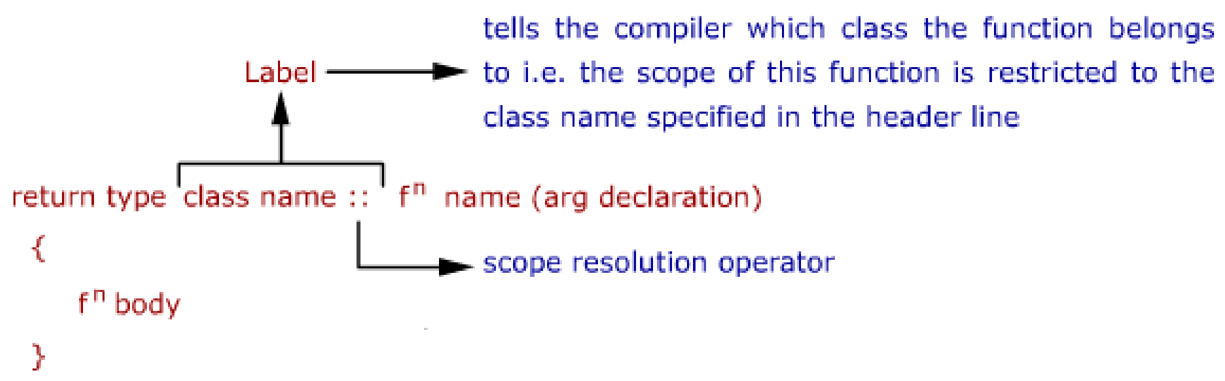
Defining Member Functions

Member functions can be defined in two places:

- Outside the class definition
- Inside the class definition

Outside the class: Member function that are declared inside a class have to be defined separately outside the class.

An importance difference between a member function and normal function is that a member function incorporates a membership "identify label" in the header.


tells the compiler which class the function belongs to i.e. the scope of this function is restricted to the class name specified in the header line
Label
scope resolution operator

```
return type class name :: fⁿ name (arg declaration)
 {
     fⁿ body
 }

void item :: get data (int a, float b)
 {
     number = a;
         cost = b;
 }
void item :: put data (void)
 {
     cout << "Number" << number << "\n";
     cout << "cost" << cost << "\n";
 }
```

The member functions have some special characteristic often used in the program development.

- Several different classes can use the same function name. The 'membership label' will resolve their scope.
- Member functions can access the private data of the class. A non-member function cannot do so (an exception to this rule is a friend function).
- A Member function can call another member function directly, without using the dot operator.

Inside the class definition

Another method of defining a members function is to replace the function declaration by the actual function definition inside the class.

```
class item
{
      int number;
      float cost;
public:
      void get data (int a, float b)
      {
          .........
          .........
      }
      void put data (void)
      {
          .........
          .........
      }
}
```

When a function is defined inside a class, it is treated as an inline function. Therefore, all the restrictions and limitations that apply to an inline function are also applicable here. Normally, only small functions are defined inside the class definition.