CLASSES AND OBJECT

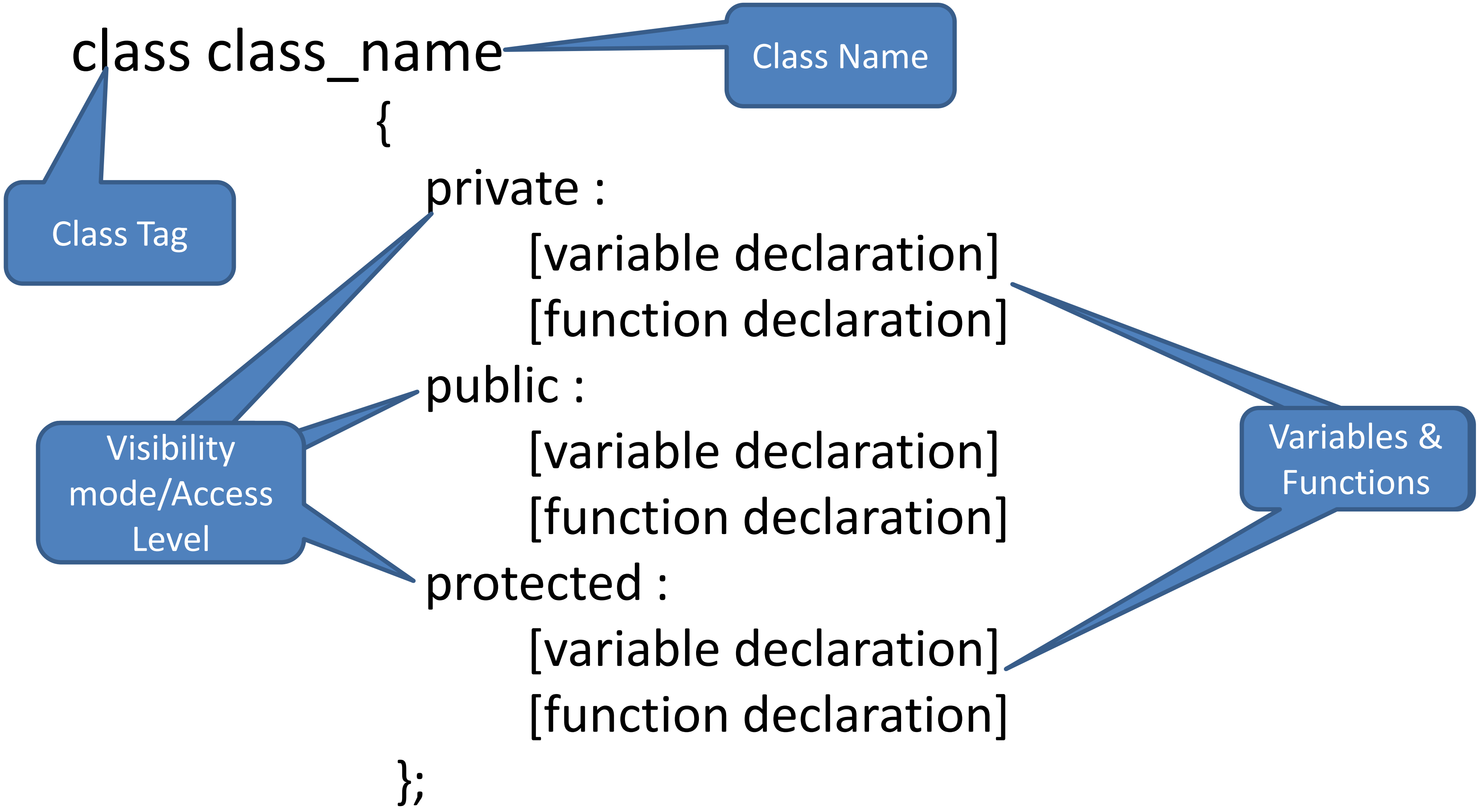
CHAPTER 04

CLASS XII

Class

- Why Class ?
  - Class is the way to represent Real-World entity that have both the Characteristics and Behaviors of an entity.

Implementation of Class



```
class class_name
            {
              private :
                   [variable declaration]
                   [function declaration]
              public :
                   [variable declaration]
                   [function declaration]
              protected :
                   [variable declaration]
                   [function declaration]
            };
```

Example

```
class account  {
                    int Account_no;
                    char Type;
                    float Balance;
               public :
                    void display();
                    float Deposit(float Amount);
                    float Withdrawl(float Amount);
               protected :
                    float cal_Interest();
               };
```

Class Function Definition

- Inside the class definition

  called **Inline definition.**

- Outside the class definition

  called **Outline definition.**

Example : Inline Definition

```
class account {
                    int Account_no;
                    char Type;
                    float Balance;
          public :
                    void display();
                    float Deposit(float Amount)
                           {
                                   Balance +=Amount;
                                   return Balance;
                           }

                    float Withdrawl(float Amount)
                           {
                                   Balance - =Amount;
                                   return Balance;
                           }
          protected :
                    float cal_Interest();
      };
```

Example : Outline Definition

```
class account   {
                    int Account_no;
                    char Type;
                    float Balance;

               public :
                    void display();
                    float Deposit(float Amount);
                    float Withdrawl(float Amount);
               protected :
                    float cal_Interest();

             };   //end of class definition


 float account :: Deposit(float Amount)
             {
                    Balance +=Amount;
                    return Balance;
             }



 float account ::  Withdrawl(float Amount)
             {
                    Balance - =Amount;
                    return Balance;
             }
```

Inline Vs Outline Function

Inline Function	Outline Function
Copy the hole function code to the called place at the time of compilation.	No copy (Jump Action followed)
More than one copy of same function	Only one copy of a Function
Fast in Execution	Slower
Wastage of Memory	Saving of Memory

Accessibility of Class Members

- **Private members** and **Protected Members** can be accessed by only the Member Functions of the class *(Accessible from only inside class).*

- **Public Members** can be accessed by *Members Functions as well as Object* directly using DOT (.) operator *(Accessible from outside class also).*

Referencing Class Members

eg.

```
account  A1,  A2;
A1.Deposit(5000);                    //correct
A1.Withdrawl(5000);                  //correct
A1.Balance;                          //incorrect
A1.Type;                             //incorrect
A1.Display();                        //correct
A1.cal_Interest();                   //incorrect
```

NOTE : Inside the Member Function no object name and DOT(.) operator required.

Scope Rule and Classes

```
#include<iostream.h>
        class stud  { int rollno;
                      float fee;
                    } ;
        stud S1;

        void main()
        {
                stud s2;
                :
        }
```

Global Object

Local Object

Nested Class

- Class within Class called nested Class.

Way 1

```
class parent
 {
  int age;
  char F_Name[30];
  char M_Name[30];
 };
```

```
class  student
    {
         int roll_no;
         char  name[30];
         int class_ ;
         parent PM;
         int age;

    };

student S1, S2;
```

Nested class

Way 2

```
class  student
    {
        int roll_no;
        char  name[30];
        int class_ ;
        class parent
         {
                 int age;
                 char F_Name[30];
                 char M_Name[30];
          } PM;
        int age;
    };

student S1, S2;
```

Nested class

Data Hiding and Encapsulation

```
class account  {
        int Account_no;
        char Type;
        float Balance;
    public :
        void display();
        float Deposit(float Amount);
        float Withdrawl(float Amount);
    protected :
        float cal_Interest();
};
```

Data Hiding implementation

FRIEND FUNCTION and FRIEND CLASS

- **FRIEND FUNCTION :** A non-Member function that can access the Private and Protected members of a class.

- **FRIEND CLASS :** A Class whose Members functions can access the Private and Protected members of another class.

Friend Function Implementation

```
class account  {
                    int Account_no;
                    char Type;
              public :
                    float Balance;
                    void display();
                    float Deposit(float Amount);
                    float Withdrawl(float Amount);
             protected :
                    friend float cal_Interest();
           } a ;


float cal_Interest()
           {
                    float interest = a.Balance * 0.03;
                    return interest;
           }



void main()
          {
           float amt =cal_Interest();
           cout<<"Your interest is : "<<amt;
          }
```

Friend Class Implementation

```
  class account  {
                    int Account_no;
                    char Type;
               public :
                    float Balance;
                    void display();
                    float Deposit(float Amount);
                    float Withdrawl(float Amount);
               protected :
                    float cal_Interest();
             } a ;

class acc_Holder  {
                public :
                    char Name[30];
                    chat Address[30];
                    friend class account;
             }  ;
```

All the Members functions of class **account** become the **friend** function of class **acc_Holder.**

Friend as Bridge

```
class accSBI
   {
        int Account_no;
        char Type;
public :
       float Balance;
      friend void disp_Tot_Bal(accSBI, accUBI);
       float Deposit(float Amount);
       float Withdrawl(float Amount);
   protected :
       float cal_Interest();

  } ;
```

```
class accUBI
   {
        int Account_no;
        char Type;
public :
       float Balance;
       friend void disp_Tot_Bal(accSBI, accUBI);
       float Deposit(float Amount);
       float Withdrawl(float Amount);
   protected :
       float cal_Interest();

  } ;
```

```
friend void disp_Tot_Bal ( accSBI   S ,    accUBI   U)
{
    cout<< S.Balance +  U.Balance);
}
```

Scope Resolution Operator (::)

```
#include<iostream.h>
int x=10;

void main()
        {
                int x=20;
                cout<< x <<" : "<< ::x <<endl;
                {
                   int x=30;
                   cout<< x <<" : "<< ::x <<endl;
                }

        }
```

```
Output :                20 : 10
                        30 : 10
```

Array in Class
and
Array of Class/Object

```
class account  {
                 int Account_no;
                 char Type;
                 float Balance;
                 char Name[30];
                public :
                 void display();
                 float Deposit(float Amount);
                 float Withdrawl(float Amount);
               protected :
                 float cal_Interest();
             } a[5] ;
```

Static Class Members

```
class JointAccount
{     int Account_no;
      char Type;
      float Balance;
      static int count_Deposit;
   public :
      void display();
      void Deposit(float Amount);
      void Withdrawl(float Amount);
      static void show()
        { cout<<"Tot Deposits "<< count_Deposit;
        }
   protected :
      float cal_Interest();
} a, b ;

int JointAccount :: count_Deposit=0;
```

Static Member declaration

Static Member Function

Static Member Definition outside class

```
void JointAccount::Deposit ( float amt)
{
    Balance += amt;
    count_Deposit++;
}


void main()
{
   a.Deposit(500);
   b.Deposit(1000);

   JointAccount :: show();
}
```

Handling static Member

Invoking of static function using class not object

Thanks………….

By : Dinesh Patel
PGT [CS]
KV IIT Powai, Mumbai