Inheritance : Extending Classes

Chapter 06

Class XII [CS]

Need of Inheritance

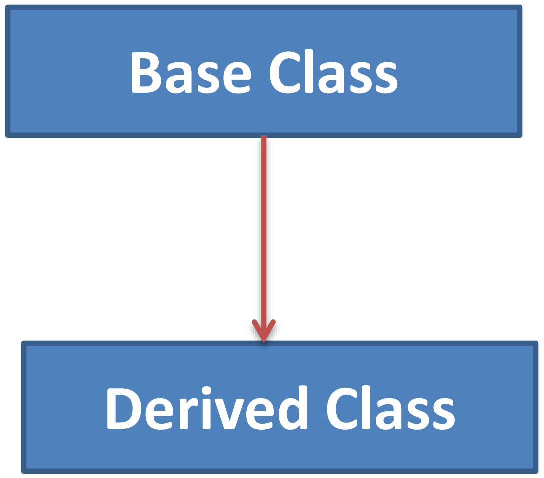
- Implement Real-World Concept
- Reusability of Code
- To have a Transitive nature

$$A \rightarrow B \rightarrow C$$

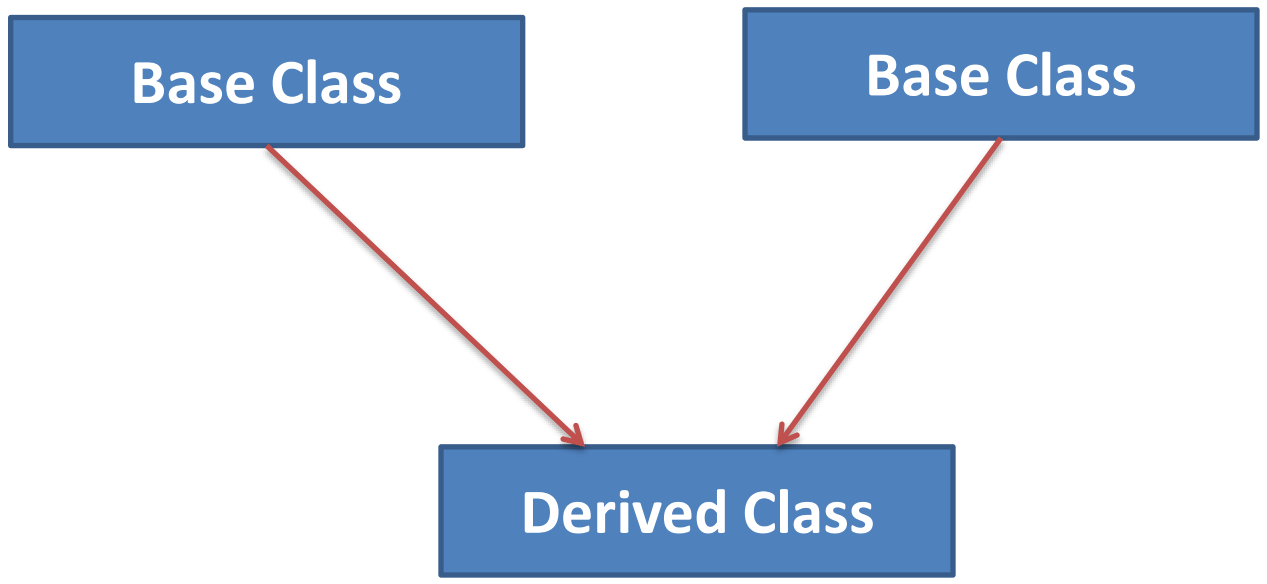
So, $A \rightarrow C$

Types of Inheritance

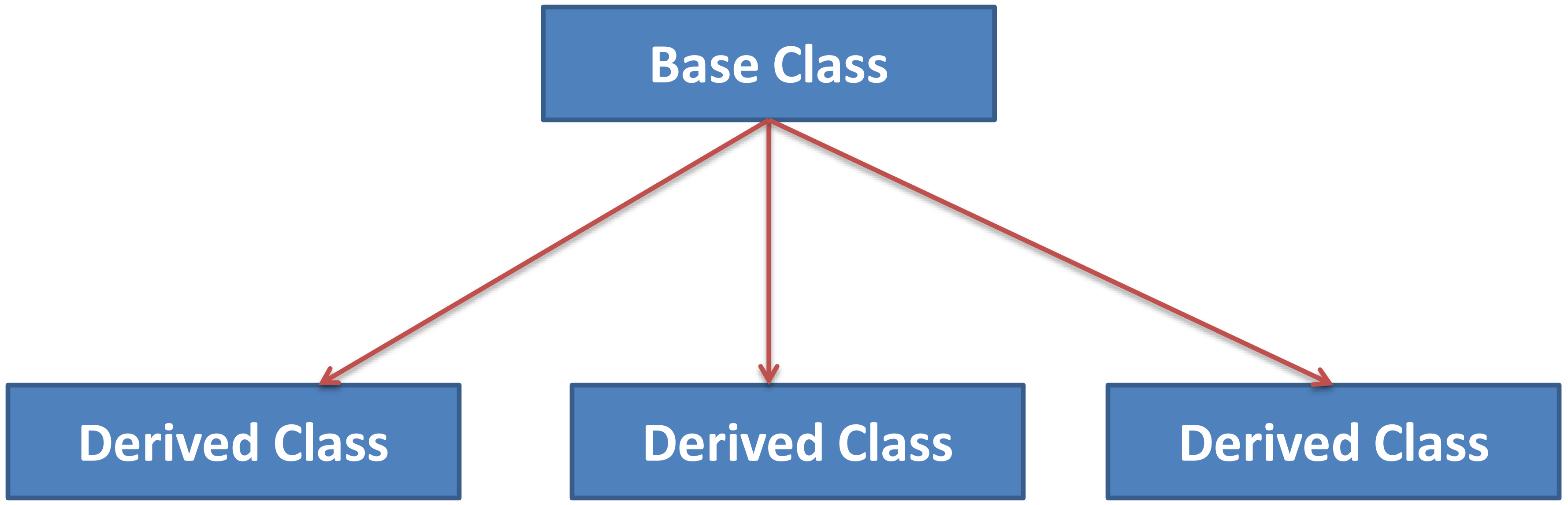
1. Single Inheritance



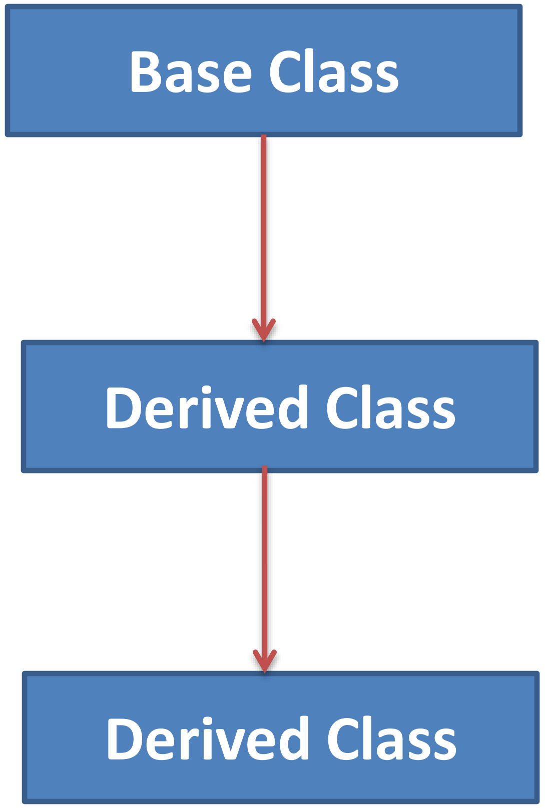
2. Multiple Inheritance



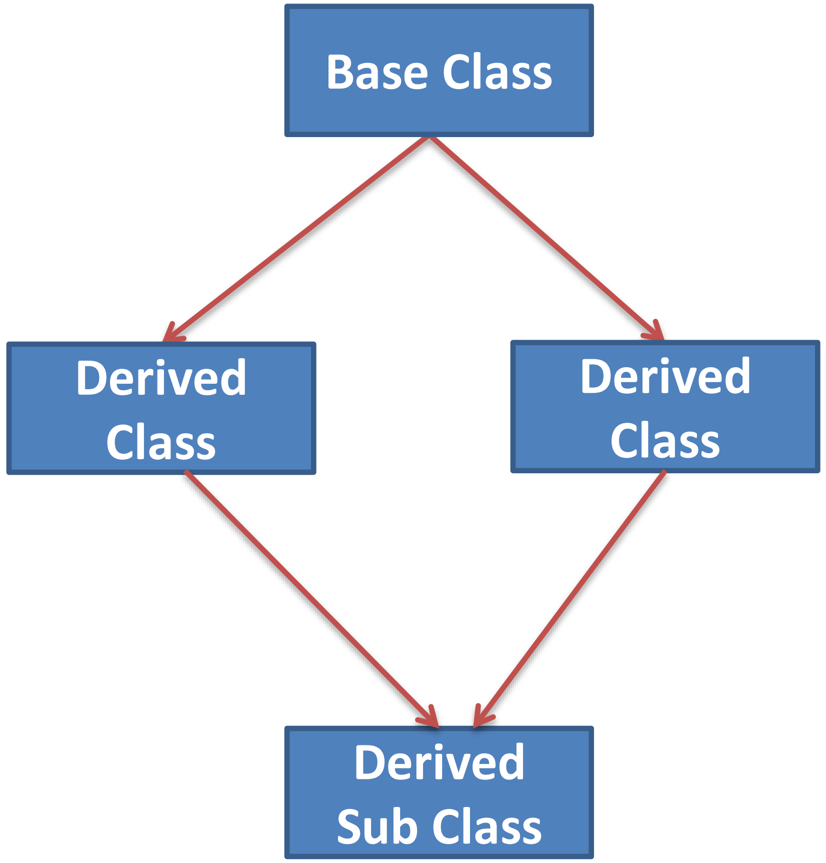
3. Hierarchical Inheritance



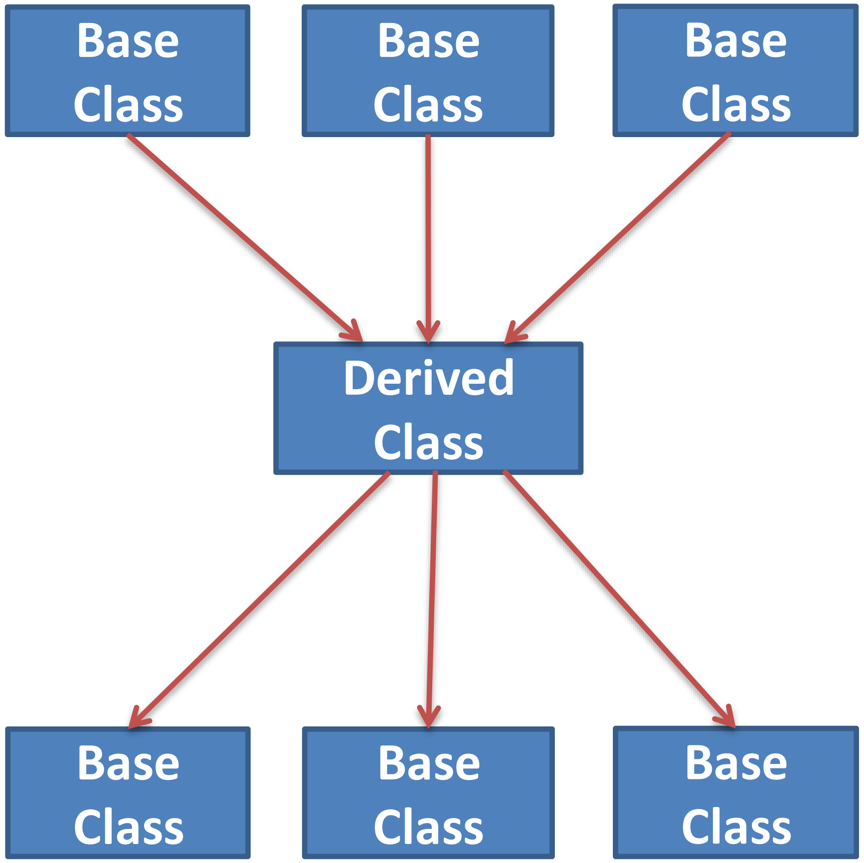
4. Multilevel Inheritance



5. Hybrid Inheritance



1



2

C++ eg. of Inheritance

1. Single Inheritance

```
class  Account
       {
           public :
             int  Account_no;
             chat Acc_Type;
             double Balance;
            void  Deposit();
      };


class  Acc_Holder  :  public  Account
       {
           public :
             void  Withdrawal();
        };
```

2. Multiple Inheritance

```
class  Bank
     {
        public :
          char Bank_Name[30];
          char  grade;
    };
```

```
class  Account
     {
        public :
           int  Account_no;
           chat Acc_Type;
           double Balance;
          void  Deposit();
    };
```

```
class  Acc_Holder  :  public  Bank,  public Account
     {
        public :
           void  Withdrawal();
     };
```

3. Hierarchical Inheritance

```
class  Account
      {
         public :
            int  Account_no;
            double Balance;
     };
```

```
class  Acc_Current : public Account
      {
         public :
           void  More_Deposit();
           void  More_Withdrawal();
     };
```

```
class  Acc_Saving  :  public  Account
      {
         public :
           void Once_Deposit()
           void  Once_Withdrawal();
     };
```

4. Multilevel Inheritance

```
class  Account
     {
         public :
           int  Account_no;
           double Balance;
    };
```

```
class  Acc_Current : public Account
     {
         public :
           void  More_Deposit();
           void  More_Withdrawal();
    };
```

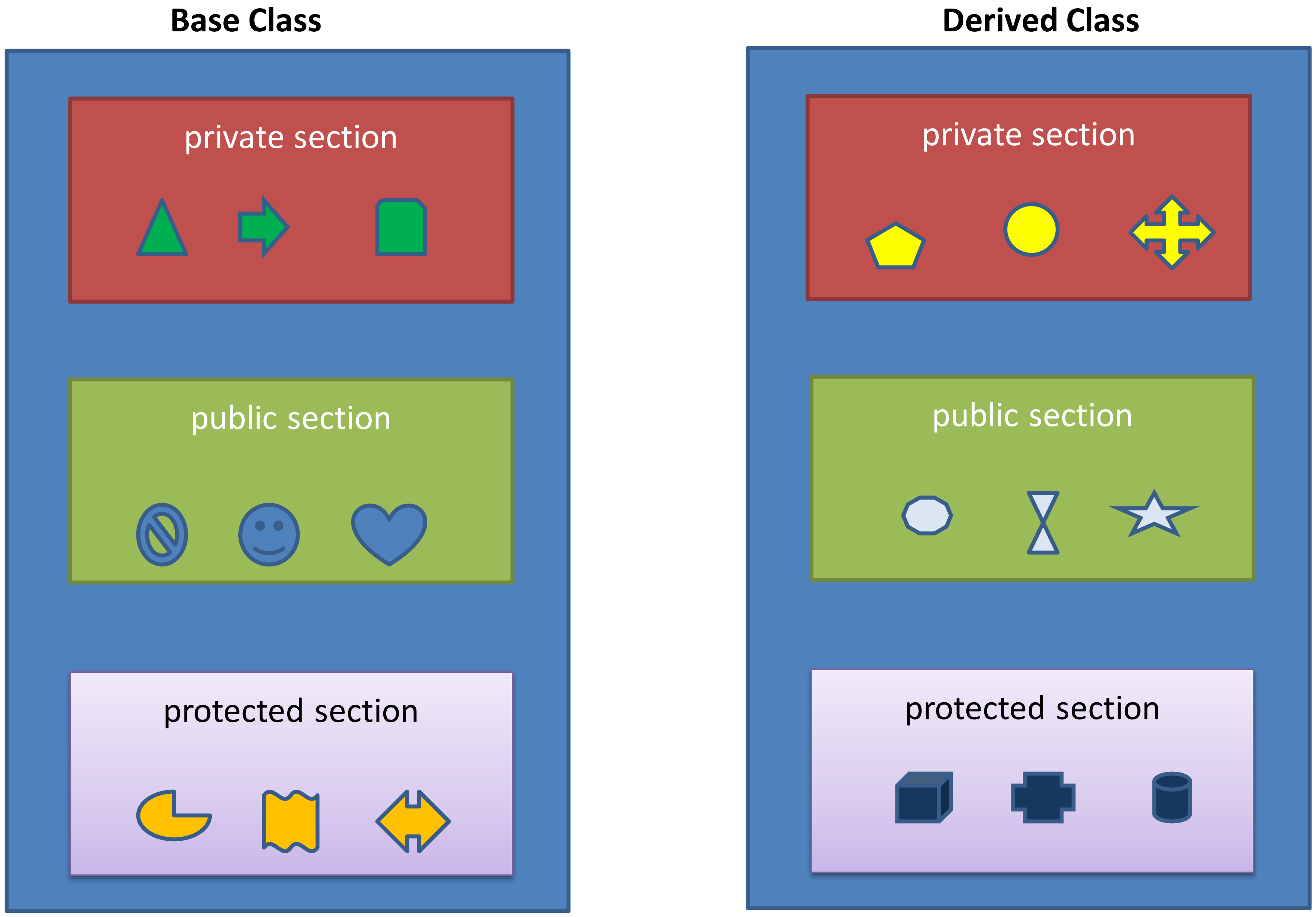
```
class  Acc_Holder :  public Acc_Current
     {
         public :
           char Holder_Name[30];
           int Account_Mode;
     };
```

5. Hybrid Inheritance

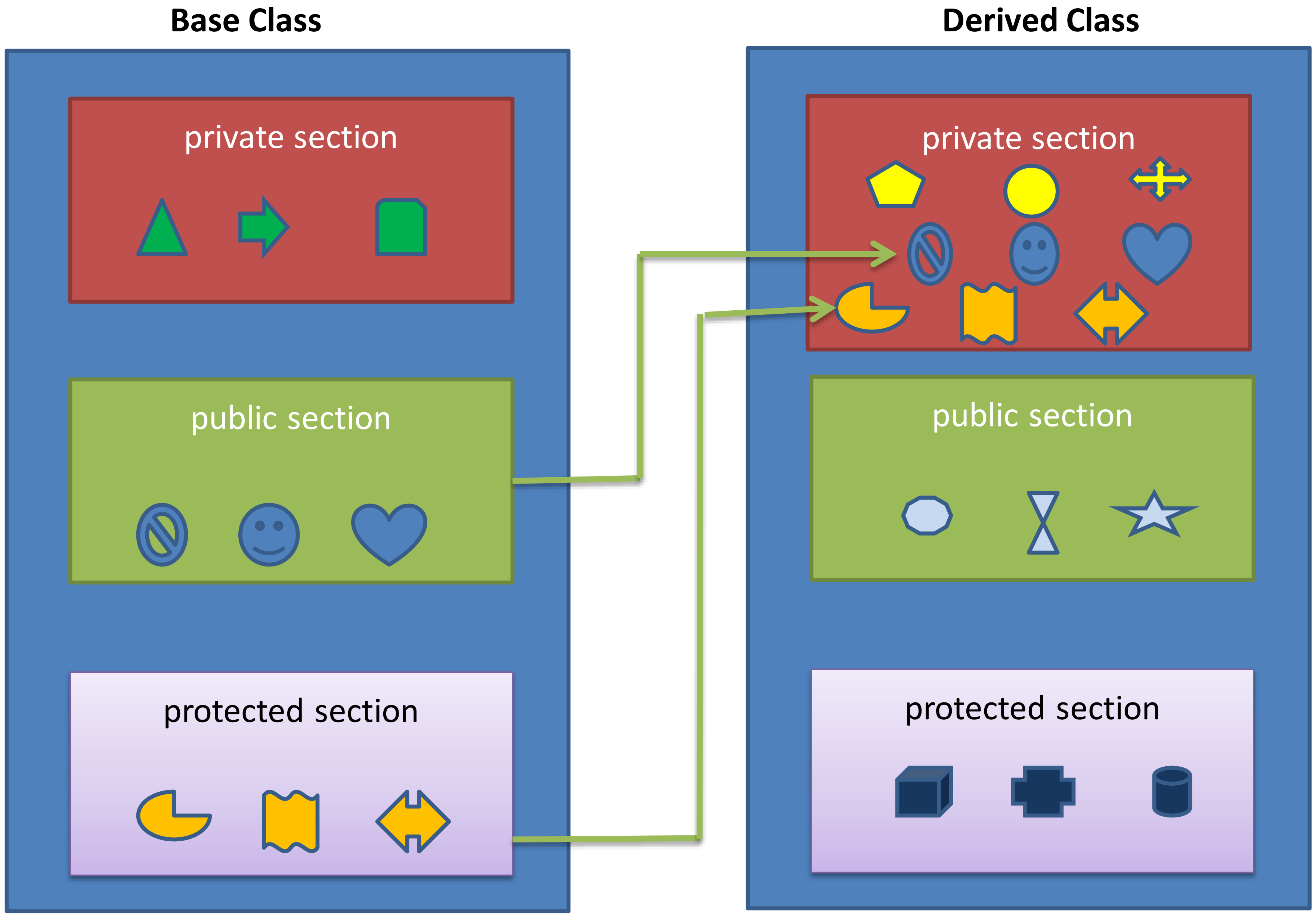
Combination of two or more type of inheritance.

Visibility Mode in Inheritance

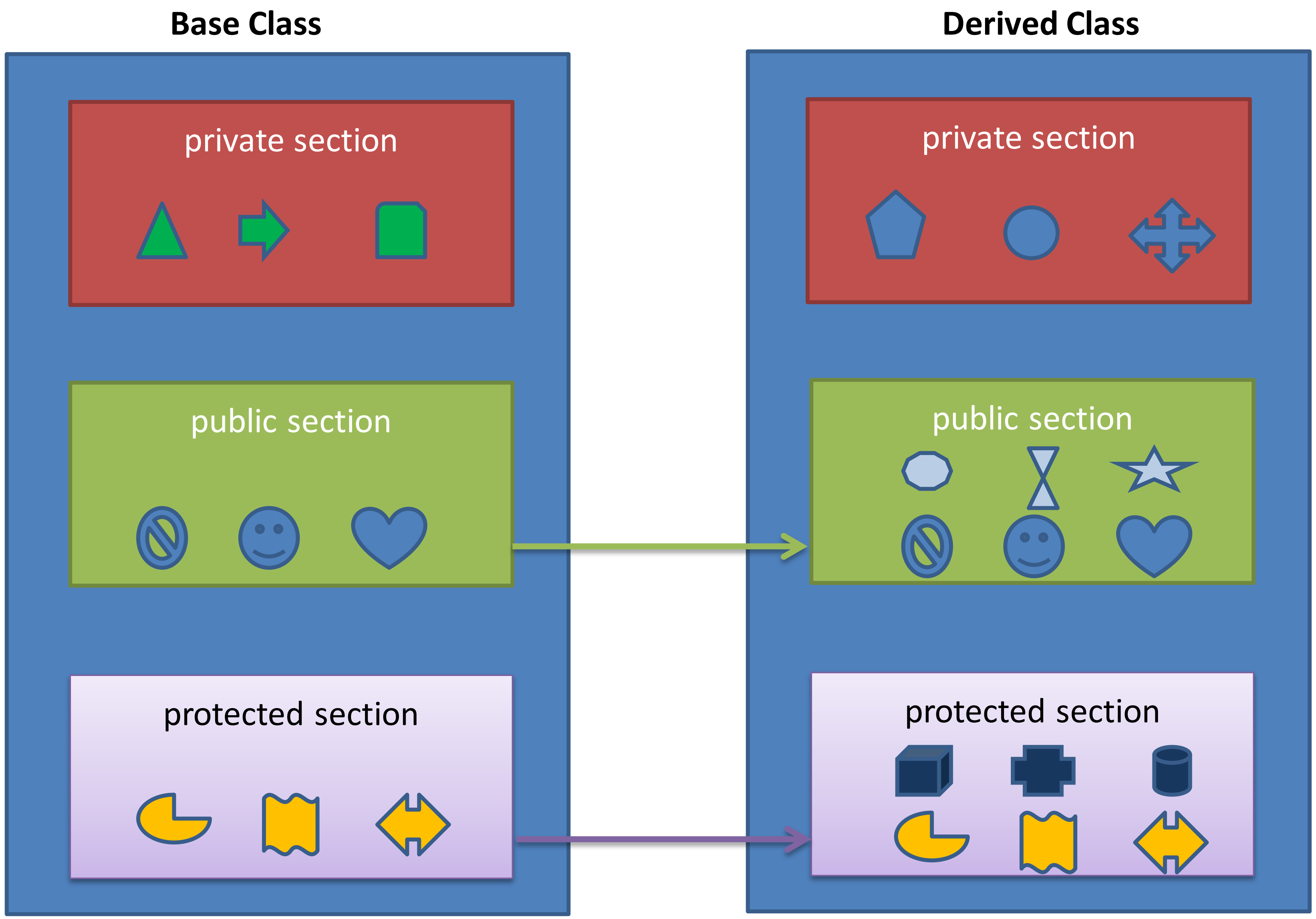
Inheritance



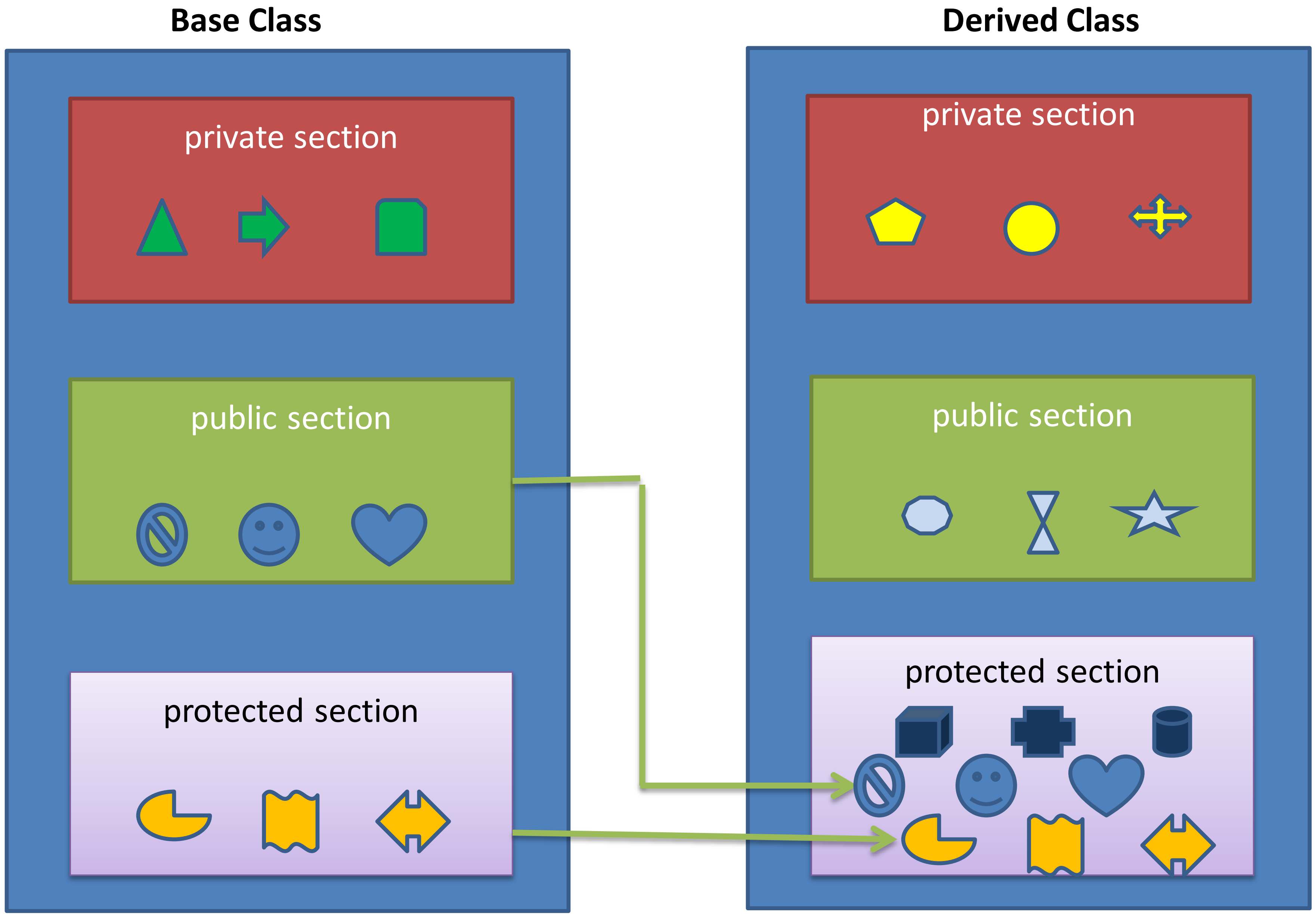
private inheritance



public inheritance



protected inheritance



Inheritance
Constructor and Destructor

- Constructor and Destructor can never be inherited.

- Then, if there is the parameterized constructor in base class then we have to pass value from derived class separately followed by base class name.

eg.

```
class BASE {
               int a;
               int b;
             public :
                BASE ( int i,   int  j )
                       {  a = i;
                          b = j;
                       }
            };


class DERIVED : public BASE
           {
               int  x;
               int  y;
             public :
                DERIVED ( int m,  int  n,   int  p,   int   q ) : BASE (p, q)
                       {  x = m;
                          y = n;
                       }
            };
```

eg. Execution of Constructor

```
class BASE_1 {
            int a;
            int b;
           public :
             BASE ( int i,   int  j )
                    {  a = i;
                       b = j;
                    }
          };


class BASE_2 {
            int c;
            int d;
           public :
             BASE ( int k,   int  l )
                    {  c = k;
                       d = l;
                    }
          };
```

Continued……………

```
class DERIVED : public BASE_1,  public BASE_2
   {
      int  x;
      int  y;
    public :
      DERIVED ( int m,  int  n,   int  p,   int   q,  int r,   int s ) : BASE _1(p, q),  BASE_2(r, s)
         {  x = m;
            y = n;
         }
   };
```

Constructor runs 3rd

Constructor runs 1st

Constructor runs 2nd

```
class DERIVED : public BASE_1,  public BASE_2
  {
     int  x;
     int  y;
   public :
     DERIVED ( int m,  int  n,   int  p,   int   q,  int r,   int s ) : BASE_2(r, s) ,   BASE _1(p, q)
           {  x = m;
              y = n;
           }
  };
```

Constructor runs 3rd

Constructor runs 1st

Constructor runs 2nd

NOTE : Which Base Class will be used first , the constructor will run first and at end the derived class constructor will run.

Virtual Base Class

```
class A {   public :
                int a;
            };

class  B1  :  public  A
           {    public :
                int x;
            };

class  B2  :  public  A
           {    public :
                int  p;
           };

class  C  :  public  B1,   public  B2
           {    public :
                int  q;
           };
```

```
class A {   public :
                int a;
            };

class  B1  :  virtual public  A
           {    public :
                int x;
            };

class  B2  :  virtual public  A
           {    public :
                int  p;
           };

class  C  :  public  B1,   public  B2
           {    public :
                int  q;
           };
```

Thanks…….

By : Dinesh Patel
PGT [Computer Science]
Kendriya Vidyalaya, NAD,
Karanja