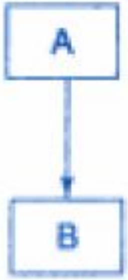
Inheritance

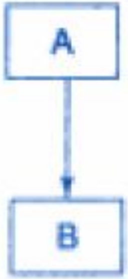
- Reusability
- Types of Inheritance
  - Single Inheritance
  - Multiple Inheritance
  - Hierarchical Inheritance
  - Multilevel Inheritance
  - Hybrid Inheritance

Types of Inheritance

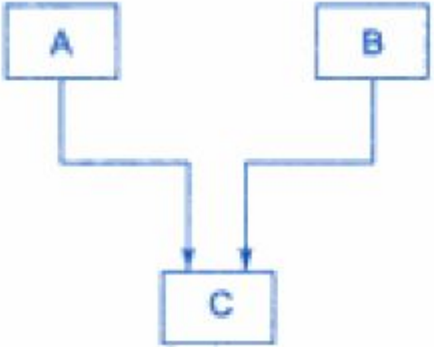


(a) Single inheritance

Types of Inheritance

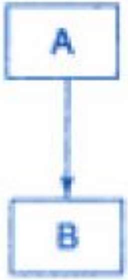


(a) Single inheritance

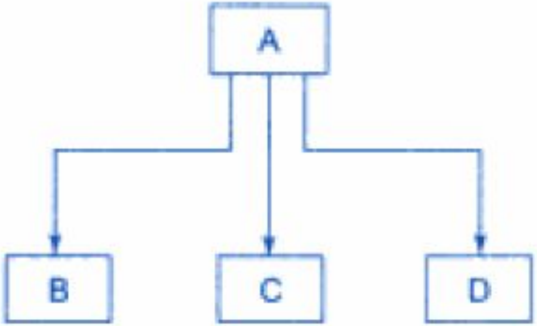


(b) Multiple inheritance

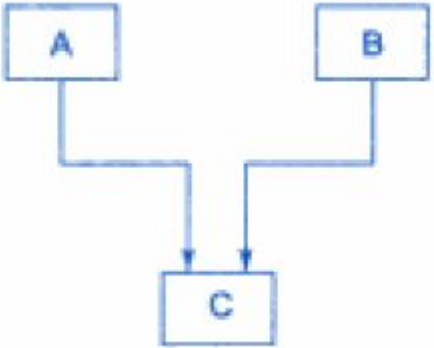
Types of Inheritance



(a) Single inheritance

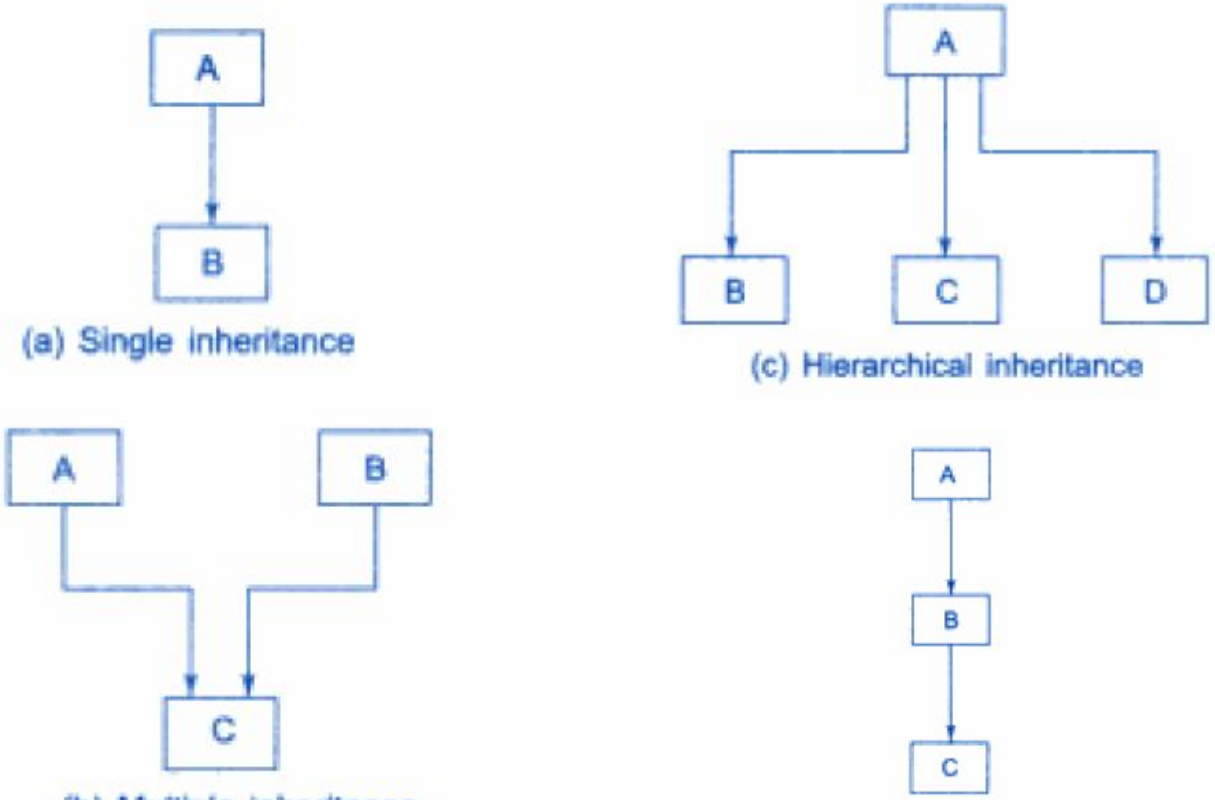


(c) Hierarchical inheritance



(b) Multiple inheritance

Types of Inheritance



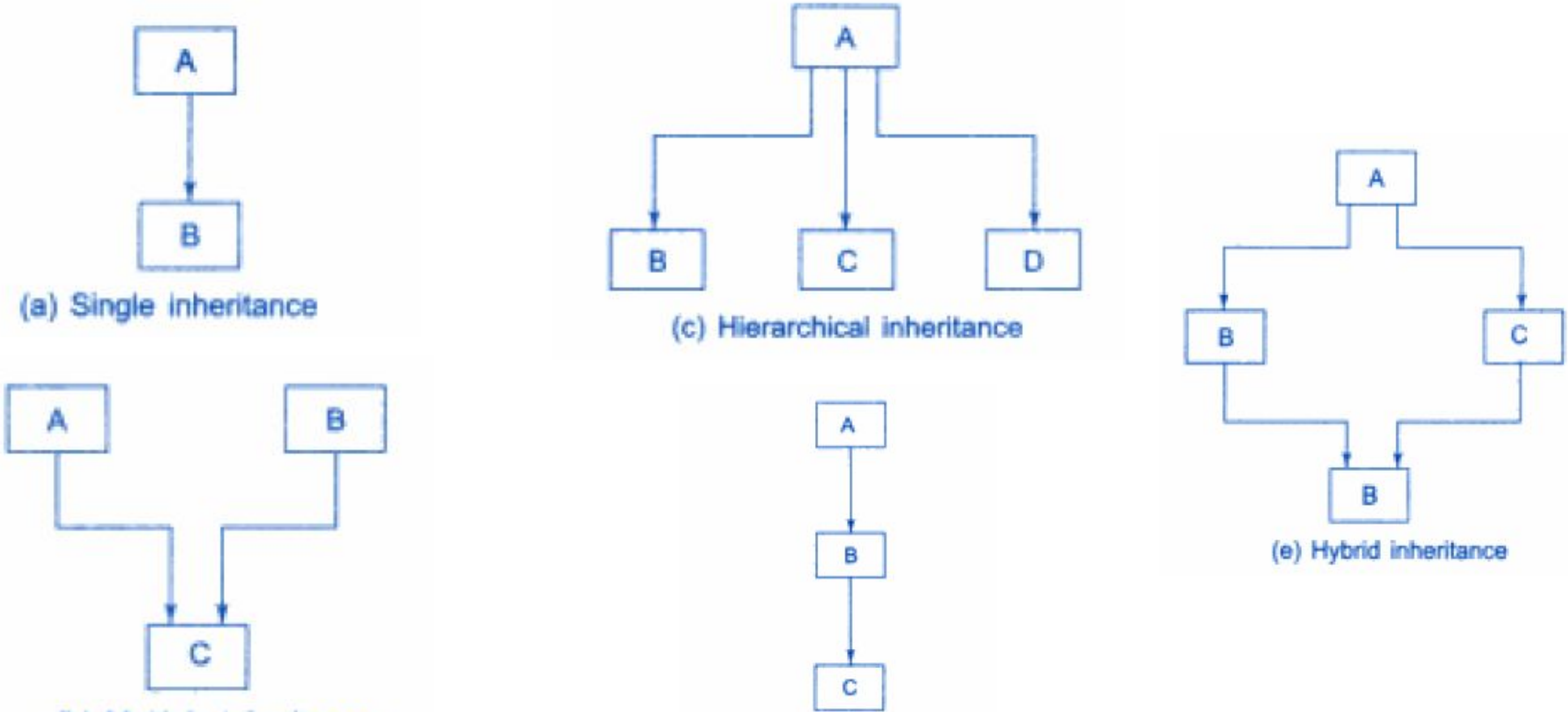
(a) Single inheritance

(c) Hierarchical inheritance

(b) Multiple inheritance

(d) Multilevel inheritance

Types of Inheritance



(a) Single inheritance

(b) Multiple inheritance

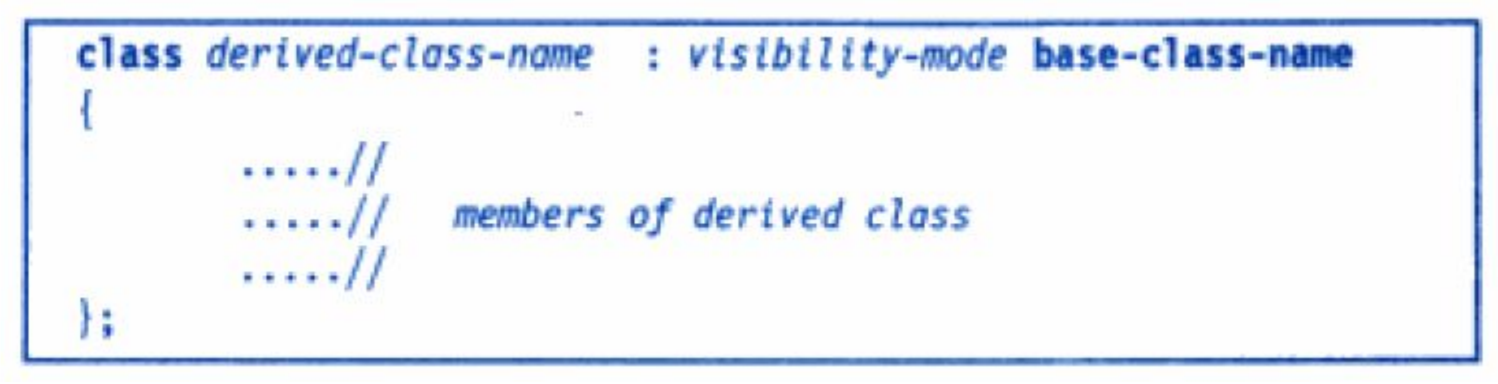
(c) Hierarchical inheritance

(d) Multilevel inheritance

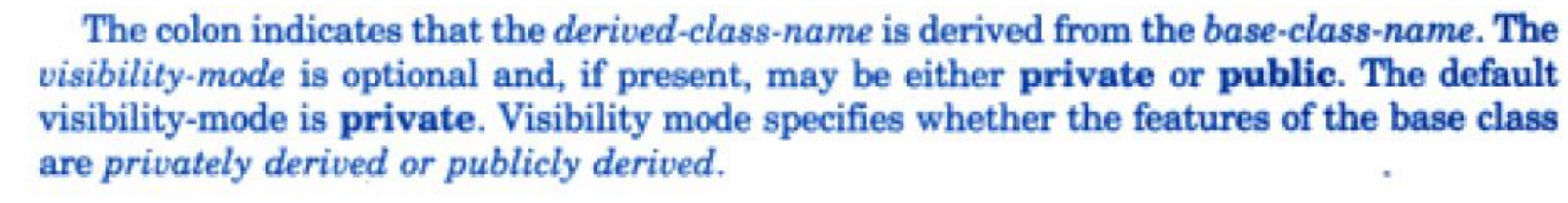
(e) Hybrid inheritance

Defining a Derived Class

A derived class can be defined by specifying its relationship with the base class in addition to its own details

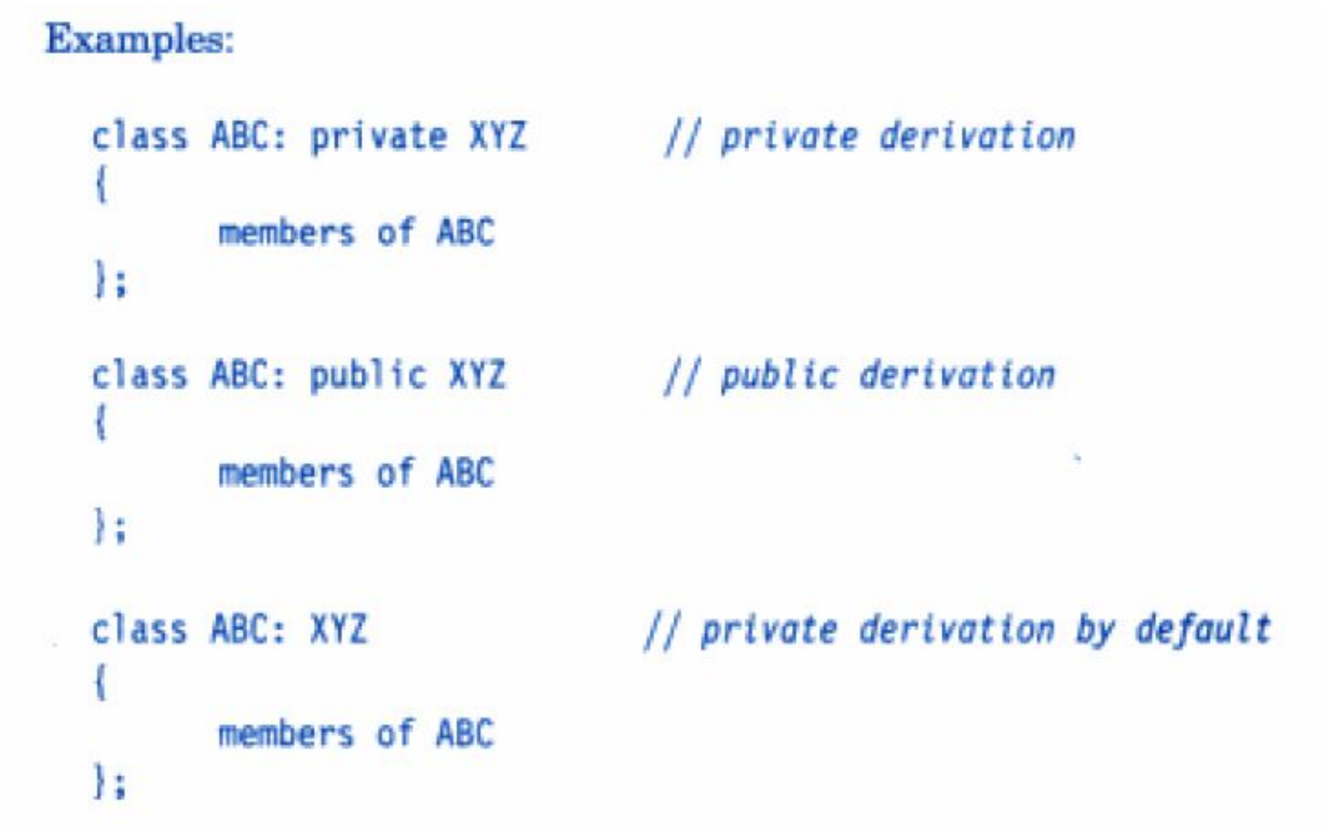


```
class derived-class-name  : visibility-mode base-class-name
{
        .....//
        .....//  members of derived class
        .....//
};
```



The colon indicates that the *derived-class-name* is derived from the *base-class-name*. The *visibility-mode* is optional and, if present, may be either **private** or **public**. The default visibility-mode is **private**. Visibility mode specifies whether the features of the base class are *privately derived or publicly derived.*

Defining a Derived Class



Examples:

```
class ABC: private XYZ       // private derivation
{
        members of ABC
};

class ABC: public XYZ        // public derivation
{
        members of ABC
};

class ABC: XYZ              // private derivation by default
{
        members of ABC
};
```

Single Inheritance

```
class baseclass
{
    private:
        int a;
    public:
        int b;
        void get_ab();
        int get_a();
        void display_a();
};
```

Single Inheritance

```
class baseclass
{
    private:
        int a;
    public:
        int b;
        void get_ab();
        int get_a();
        void display_a();
};
```

```
class derived:public baseclass
{
    private:
        int c;
    public:
        void mult();
        void display_mult();
};
```

Single Inheritance

```
void baseclass::get_ab()
{
    a=5; b=10;
}

int baseclass::get_a()
{
    return a;
}

void baseclass::display_a()
{
    cout<<"a= "<<a<<”\n”;
}
```

```
void derived::mult()
{
    c=b*get_a();
}

void derived::display_mult()
{
    cout<<"a= "<<get_a()<<"\n";
    cout<<"b= "<<b<<"\n";
    cout<<"c= "<<c<<"\n";
}
```

Single Inheritance

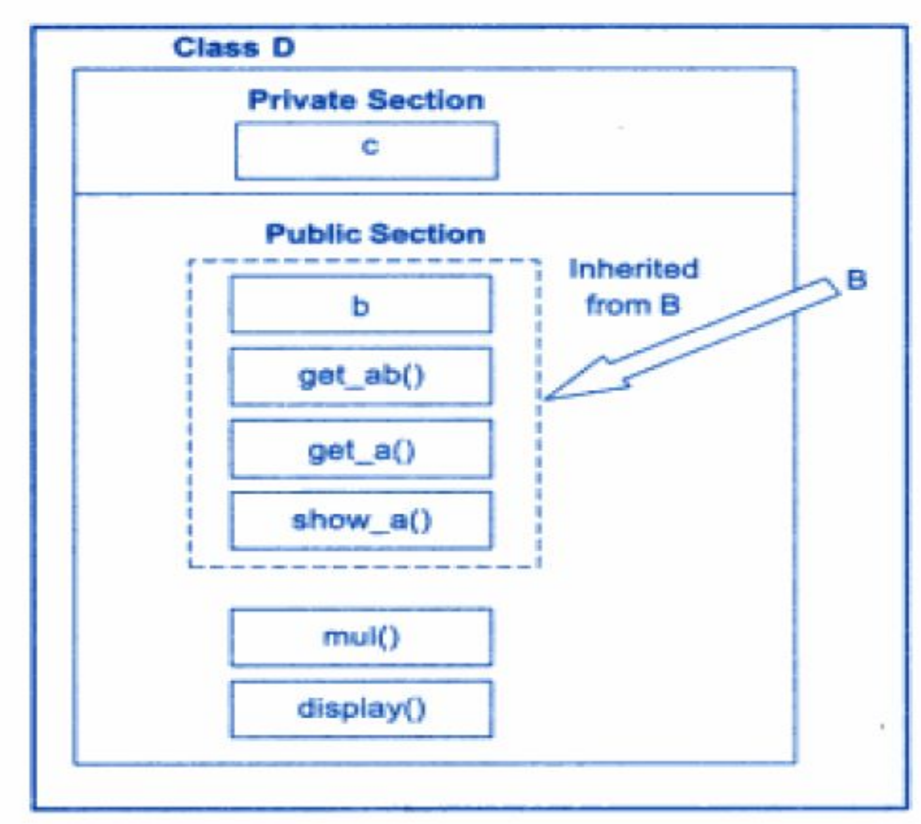
```
int main()
{
    derived d;
    d.get_ab();
    d.mult();
    d.display_a();
    d.display_mult();

    d.b=20;
    d.mult();
    d.display_mult();
}
```

Single Inheritance

```
int main()
{
    derived d;
    d.get_ab();
    d.mult();
    d.display_a();
    d.display_mult();

    d.b=20;
    d.mult();
    d.display_mult();
}
```

```
Output:

a=5
a=5
b=10
c=50

a=5
b=20
c=100
```

Single Inheritance



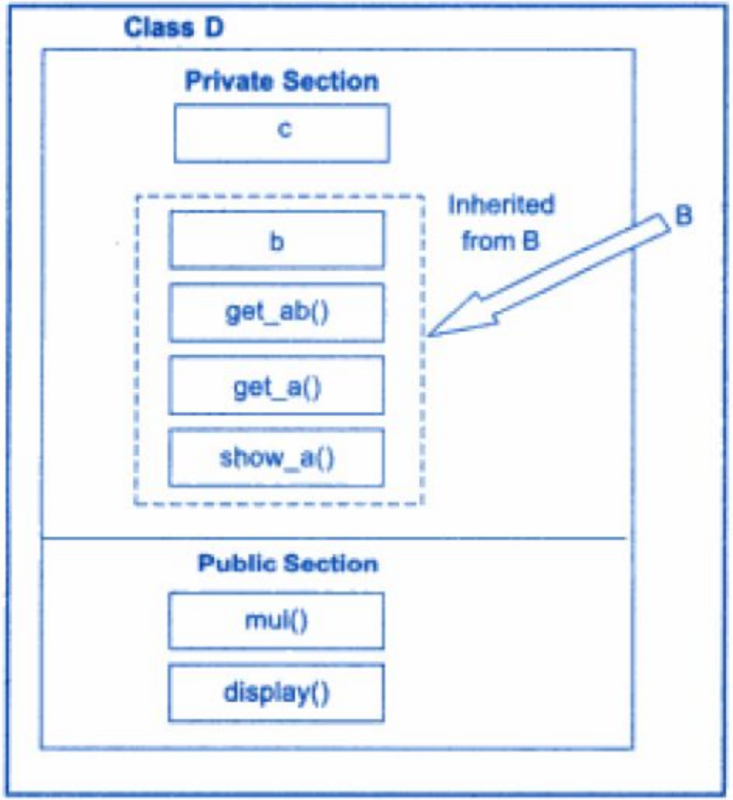
Single Inheritance

Now, lets us consider the case of private derivation

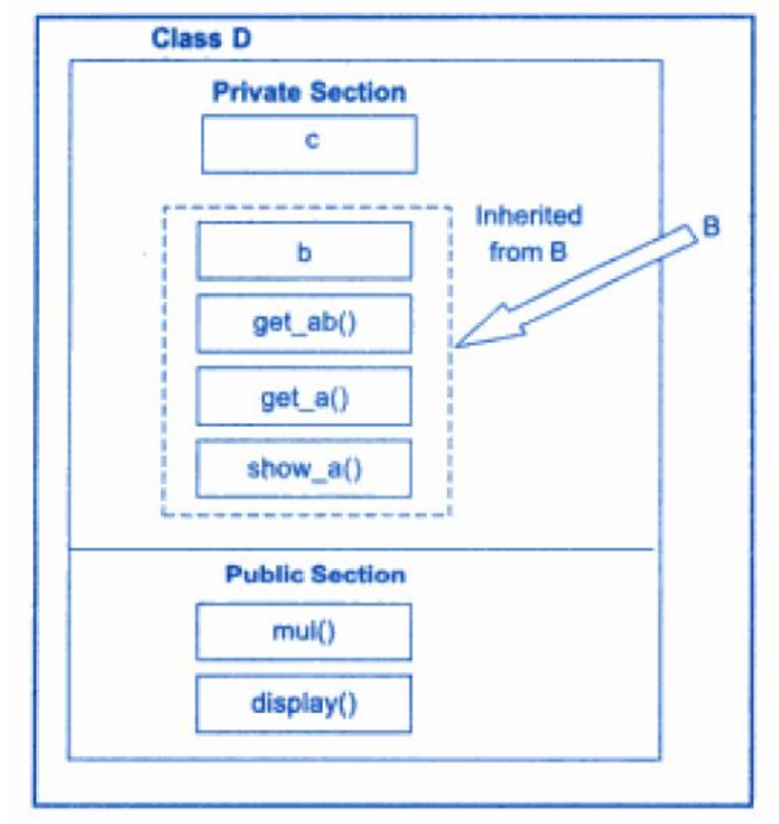
```
class baseclass
{
    private:
        int a;
    public:
        int b;
        void get_ab();
        int get_a();
        void display_a();
};
```

```
class derived:private baseclass
{
    private:
        int c;
    public:
        void mult();
        void display_mult();
};
```

Single Inheritance



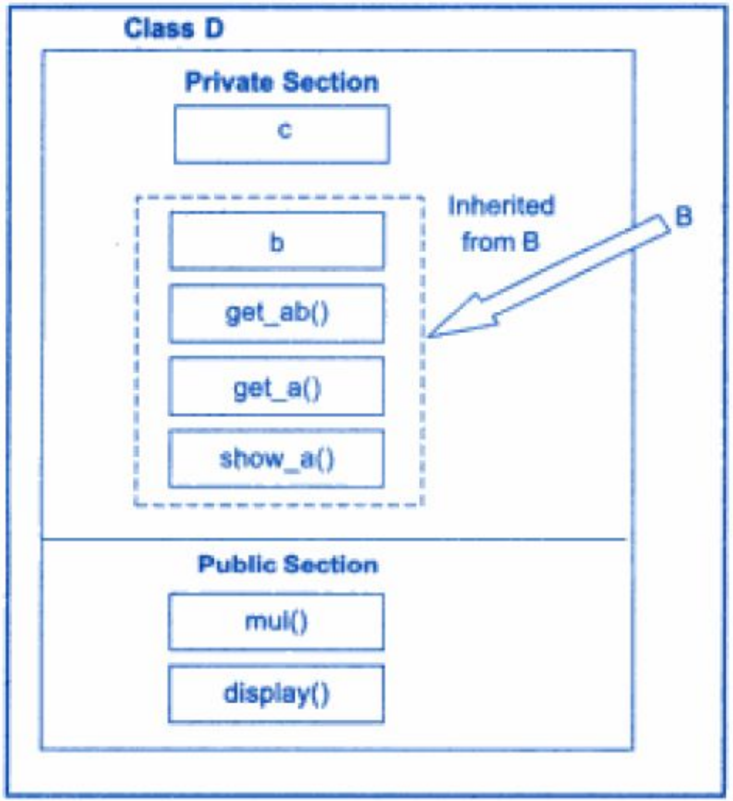
Single Inheritance



The statements such as

```
    d.get_ab();
    d.get_a();
    d.display_a();
```

Single Inheritance



```
The statements such as
    d.get_ab();
    d.get_a();
    d.display_a();
```

This will not work

Single Inheritance

```
void derived::mult()
{
    get_ab();
    c=b*get_a();
}

void derived::display_mult()
{
    display_a();
    cout<<"b= "<<b<<"\n";
    cout<<"c= "<<c<<"\n";
}
```

```
int main()
{
    derived d;
    d.mult();
    d.display_mult();
    return 0;
}
```

Output:

```
   a=5
   b=10
   c=50
```