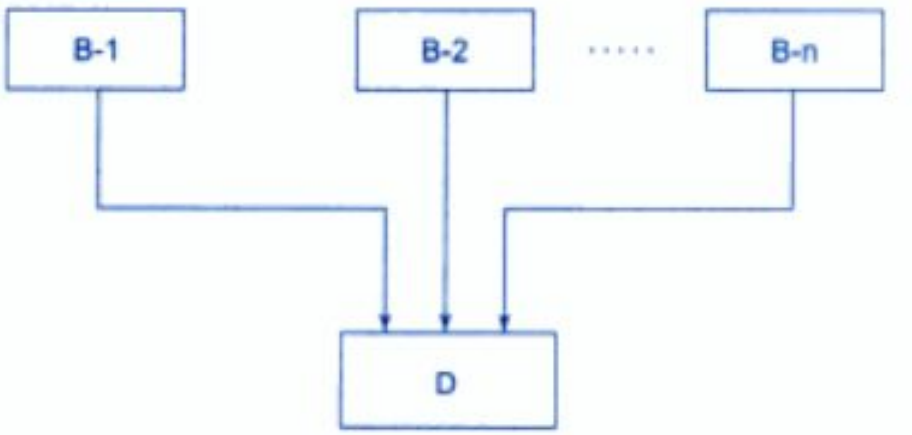
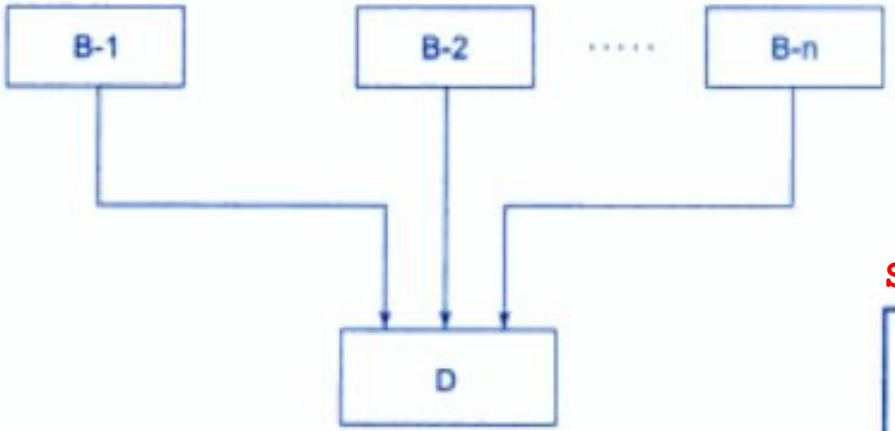
Multiple Inheritance

A class can inherit attributes from two or more base classes called multiple inheritance.



Multiple Inheritance

A class can inherit attributes from two or more base classes called multiple inheritance.



Syntax

```
class D: visibility B-1, visibility B-2 ...
{
      .....
      .....(Body of D)
      .....
};
```

Multiple Inheritance

```
class M
{
  protected:
       int m;
  public:
       void get_m(int);
};
```

```
class N
{
  protected:
       int n;
  public:
       void get_n(int);
};
```

Multiple Inheritance

```
class M
{
  protected:
      int m;
  public:
      void get_m(int);
};
```

```
class N
{
  protected:
      int n;
  public:
      void get_n(int);
};
```

Classes P is derived from class M and N as follows;

```
class P : public M, public N
{
  public:
     void display(void);
};
```

Multiple Inheritance

The derived class P,would contain all the members of M and N

```
class P
{
  protected:
      int m;
      int n;
  public:
      void get_m(int);
      void get_n(int);
      void display(void);
};
```

Multiple Inheritance

The derived class P,would contain all the members of M and N

```
class P
{
  protected:
        int m;
        int n;
  public:
        void get_m(int);
        void get_n(int);
        void display(void);
};
```

```
void P :: display(void)
{
      cout << "m = " << m << "\n";
      cout << "n = " << n << "\n";
      cout << "m*n =" << m*n << "\n";
};
```

Multiple Inheritance

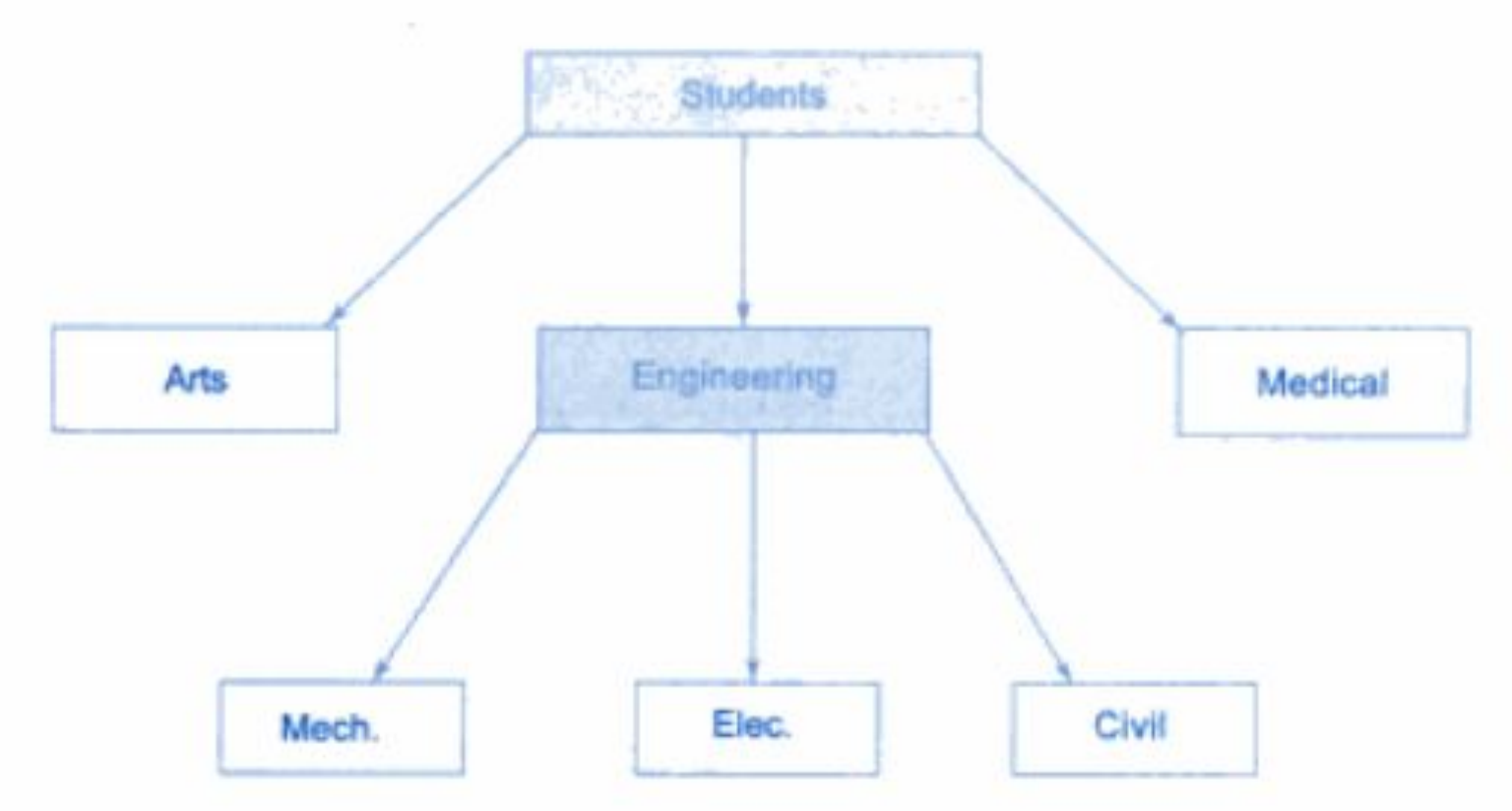
The derived class P,would contain all the members of M and N

```
class P
{
  protected:
        int m;
        int n;
  public:
        void get_m(int);
        void get_n(int);
        void display(void);
};
```

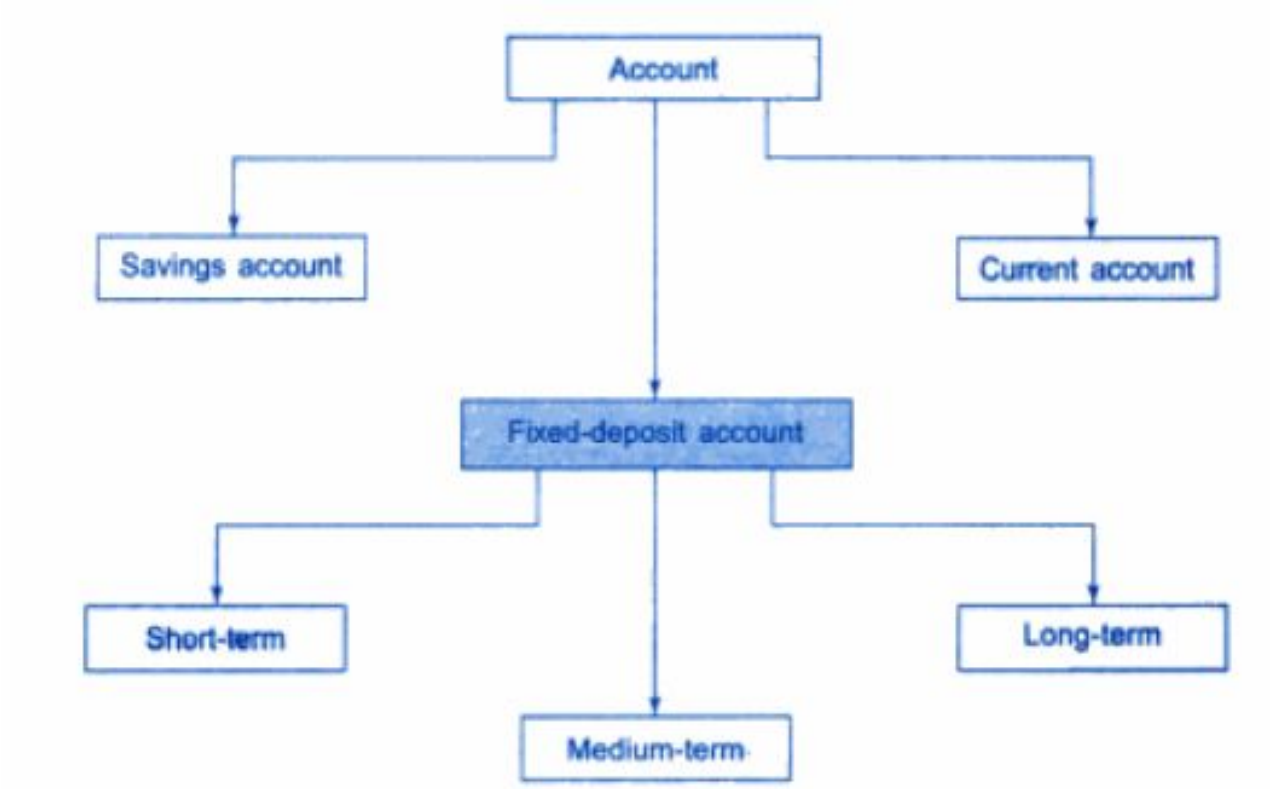
```
void P :: display(void)
{
      cout << "m = " << m << "\n";
      cout << "n = " << n << "\n";
      cout << "m*n =" << m*n << "\n";
};
```

```
main()
{
      P p;
      p.get_m(10);
      p.get_n(20);
      p.display();
}
```

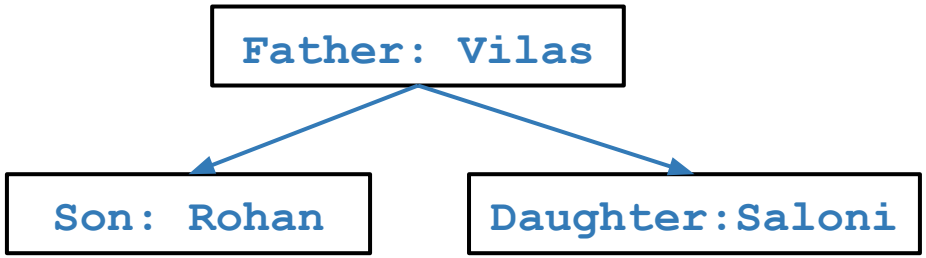
Hierarchical Inheritance



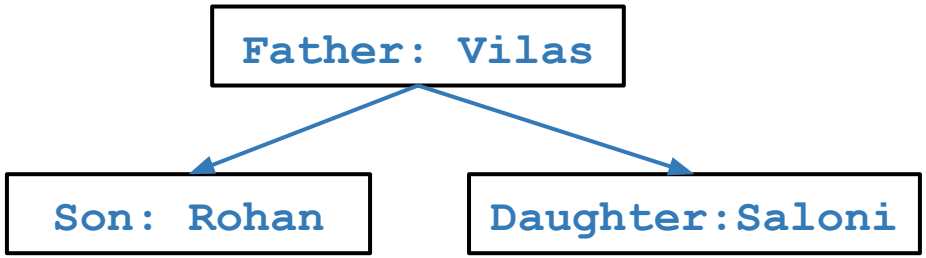
Hierarchical Inheritance



Hierarchical Inheritance

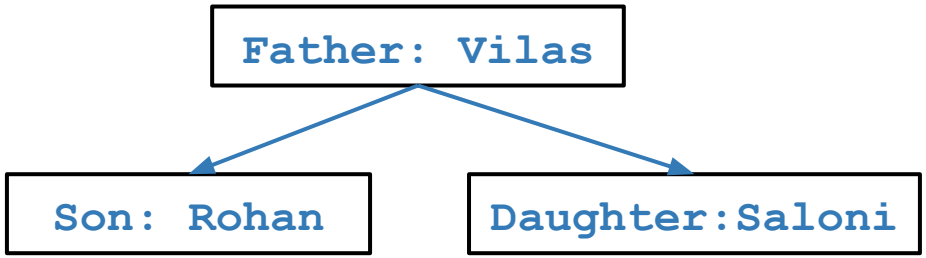


Hierarchical Inheritance




```
Class father
{
      public:
            void father_name()
            {      cout<<"Vilas"; }
};
```

Hierarchical Inheritance




```
Class father
{
      public:
            void father_name()
            {      cout<<"Vilas"; }
};
```

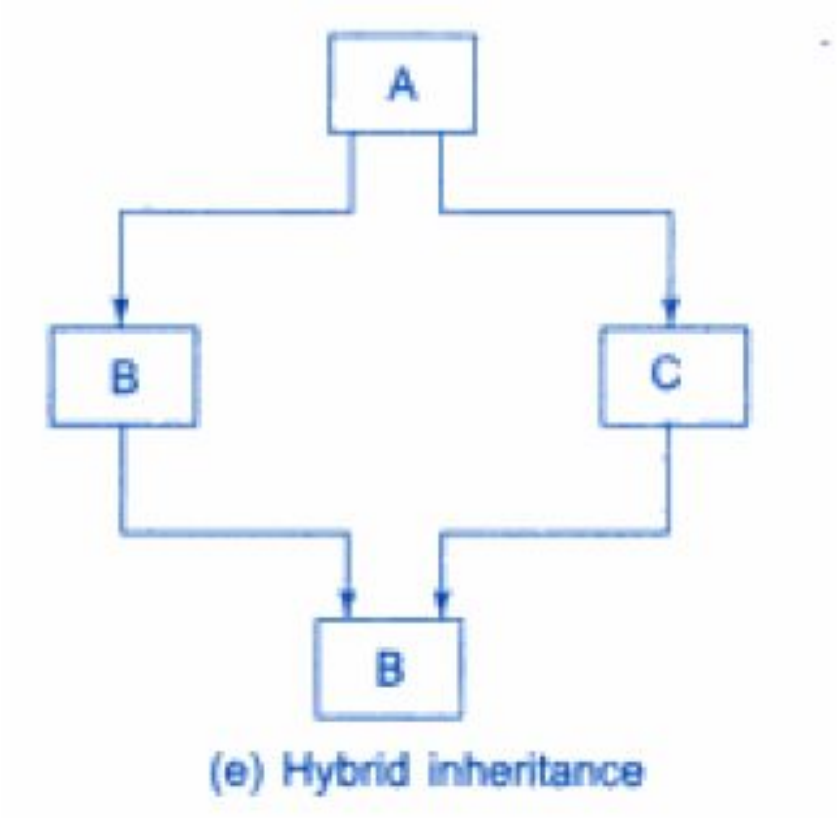
```
Class son:public father
{
      public:
            void son_name()
            {      cout<<"Rohan"; }
};
```

```
Class daughter:public father
{
      public:
            void daughter_name()
            {      cout<<"Saloni"; }
};
```

Hierarchical Inheritance

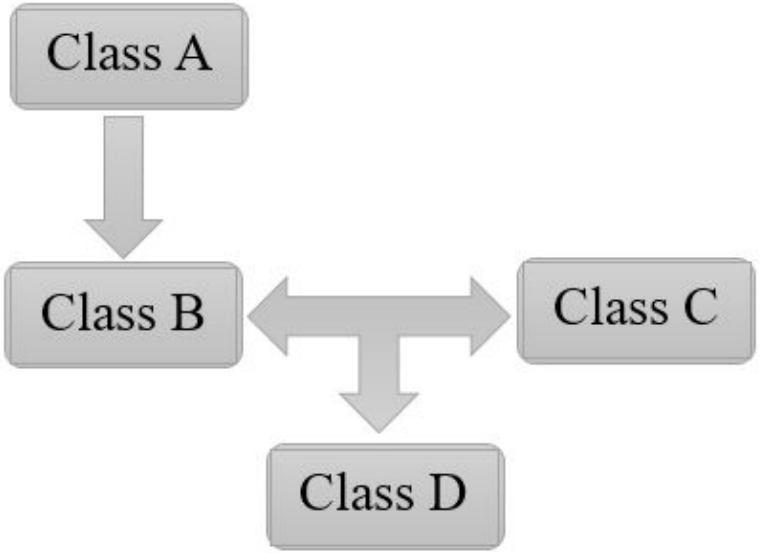
```
int main()
{
    son s;
    s.father_name();
    s.son_name();

    daughter d;
    d.father_name();
    d.daughter_name();
    return(0);
}
```

Hybrid Inheritance

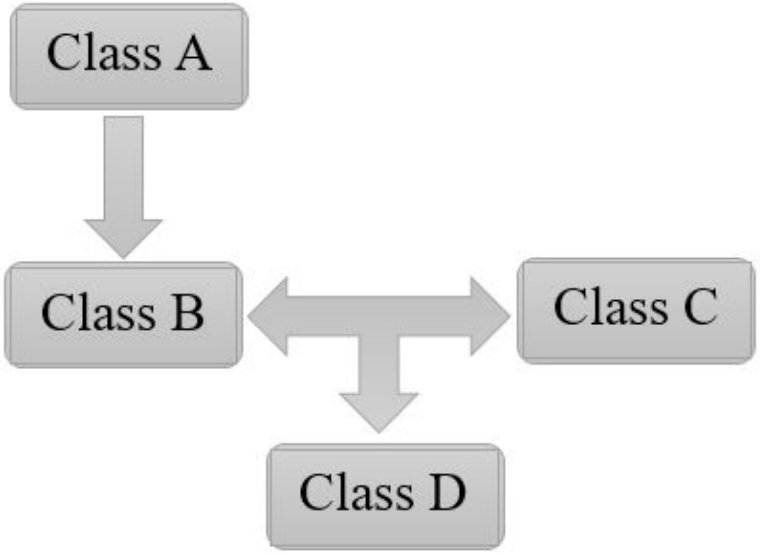


(e) Hybrid inheritance

Hybrid Inheritance



Hybrid Inheritance




```
class A
{
    .....
};
class B : public A
{
    .....
} ;
class C
{
    .....
};
 class D : public B, public C
{
    .....
};
```

Hybrid Inheritance