Course: PARADIGMS AND COMPUTER PROGRAMMING FUNDAMENTALS (PCPF)

Course Instructor

Mrinmoyee Mukherjee B.E (Electronics), M.E (EXTC), PhD (Pursuing)
Assistant Professor
Department of Information Technology
St. Francis Institute of Technology
email: mrinmoyeemukherjee@sfit.ac.in
Academic Year: 2023-24 (Odd Semester)

OUTLINE OF UNIT-2

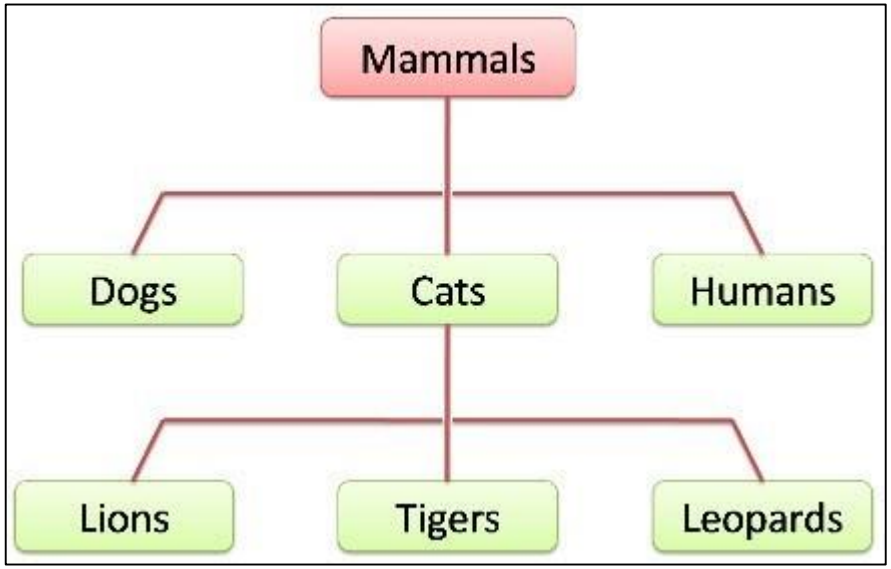
Sub-Unit	Contents
2.1	Grouping of data and operations
2.2	Encapsulation
2.3	Overloading and polymorphism
2.4	Inheritance
2.5	Initialization and finalization
2.6	Dynamic Binding

INHERITANCE

- Inheritance is a relationship between two or more classes where derived class inherits the properties of existing base classes

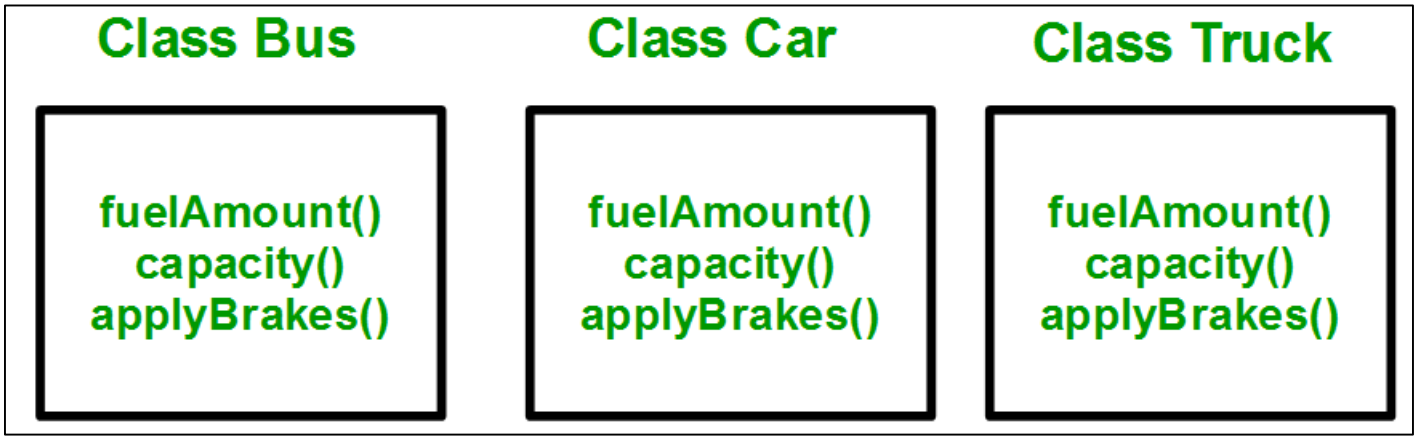
- Base class: It is the class whose properties are inherited by another class. It is also called as Super class or Parent class

- Derived class: It is the class that inherit properties from the base class(es). It is also called sub class or child class

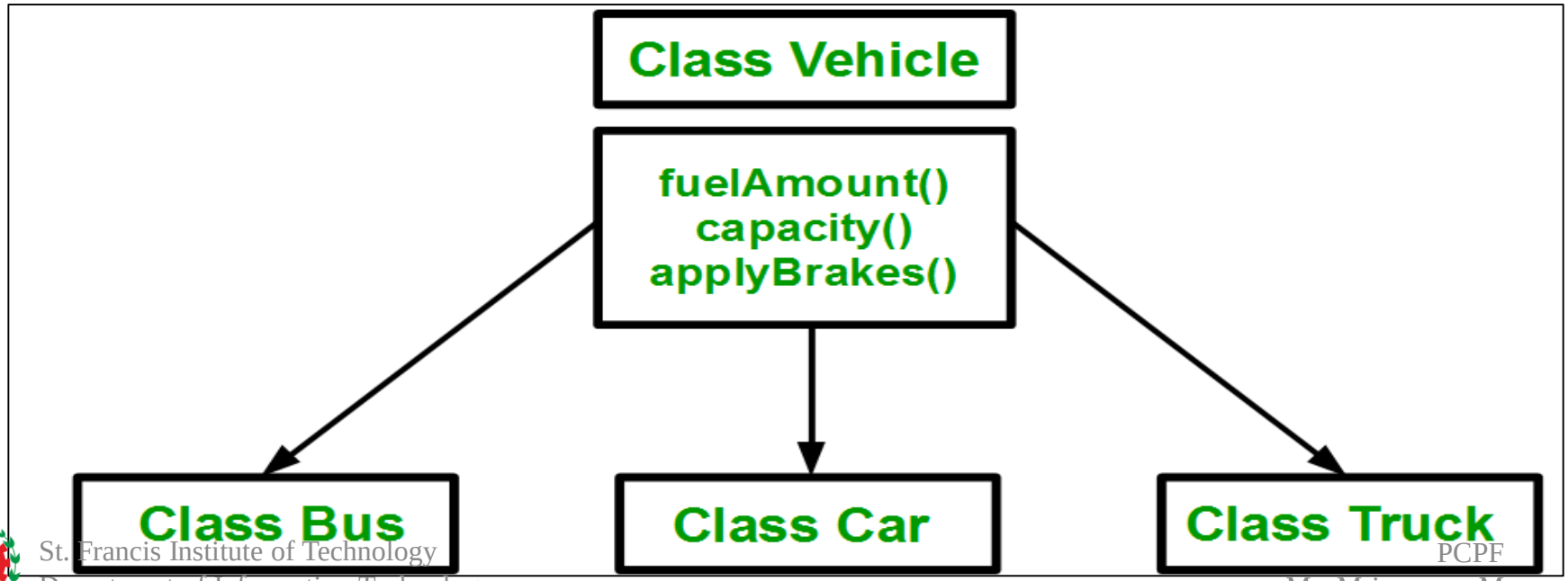


It is useful for code reusability: reuse attributes and methods of an existing class when you create a new class

NEED FOR INHERITANCE



- There is duplication of same code 3 times
- This increases the chances of error and data redundancy.
- To avoid this type of situation, inheritance is used



Example of Inheritance

Inheritance in java

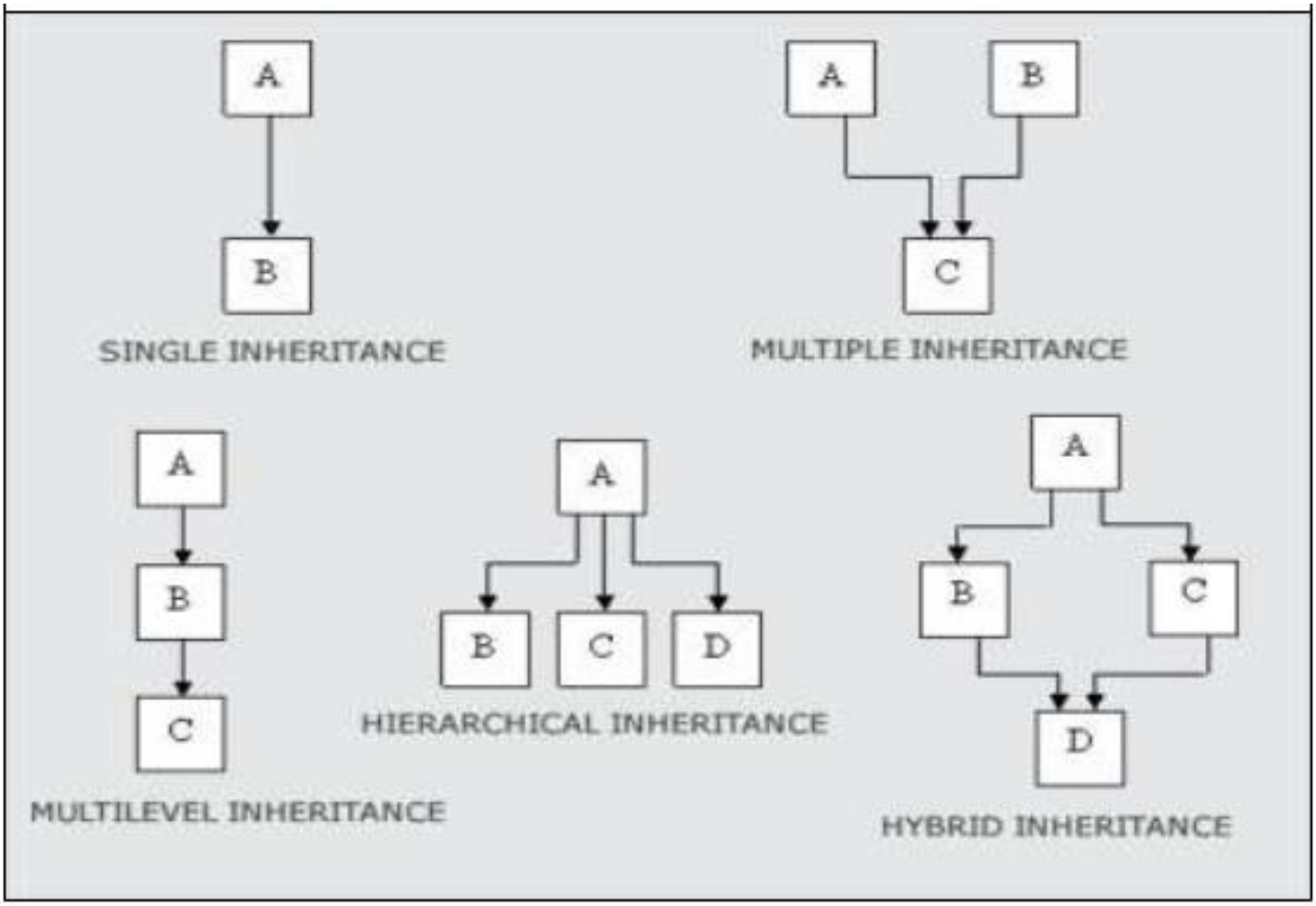
```
class Area {
   public int getArea (int l, int b) {
              return l * b;
              }
}
class Rectangle extends Area {
   int length; int breadth;
   public Rectangle() {
         length = 7; breadth = 4;
              }
}

# main class
class myClass1 {
   public static void main(String[] args) {
      System.out.println("Hello, World!");
      Rectangle rt=new Rectangle();
      System.out.println(rt.getArea(4,2));
}}
```

Inheritance in C++

```
#include <iostream>
using namespace std;
class Area {
    public:
    int getArea (int l, int b) {
                return l * b; }
};
class Rectangle : public Area {
     int length; int breadth;
     public:   Rectangle() {
           length = 7; breadth = 4;
                       }
      int area() {
           return Area::getArea(length, breadth);
                }
};
int main()  {
      Rectangle rt;
      cout << "Area : " << rt.area() << end;
      return 0; }
```

TYPES OF INHERITANCE



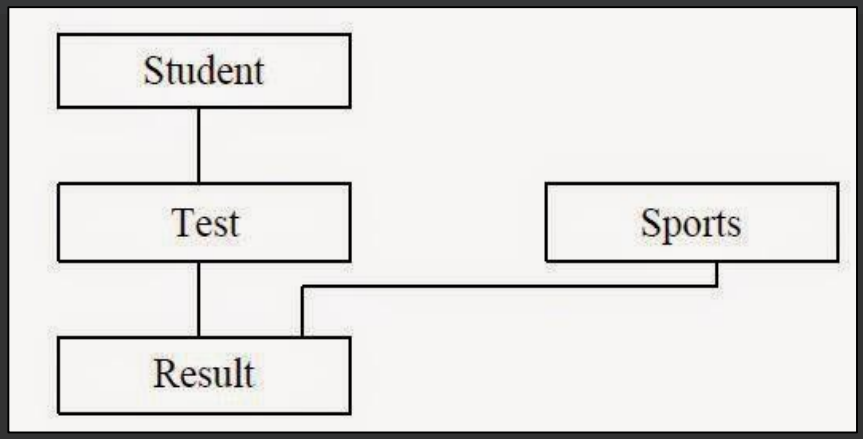
MULTIPLE INHERITANCE

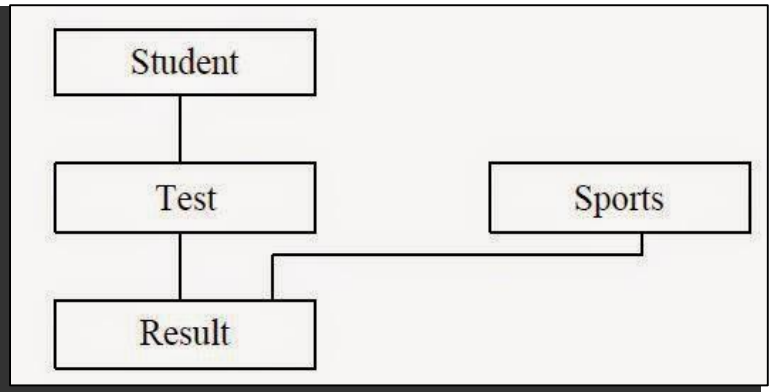
```
#include<iostream>
using namespace std;
//Define class A
class A
{
    protected:
    int m;
    public:
    void get_m(int);
};
void A::get_m(int x)
{
    m=x;
}
//Define class B
class B
{
    protected:
    int n;
    public:
    void get_n(int);
};
void B::get_n(int y)
{
    n=y;
}
```

```
//Define class P from A and B
class P:public A,public B
{
    public:
    void display(void);
};
void P::display(void)
{
    cout<<"m="<<m<<endl;
    cout<<"n"<<n<<endl;
    cout<<"m*n"<<m*n<<endl;
}
int main()
{
    P p;
    p.get_m(10);
    p.get_n(20);
    p.display();
    return 20;
}
```

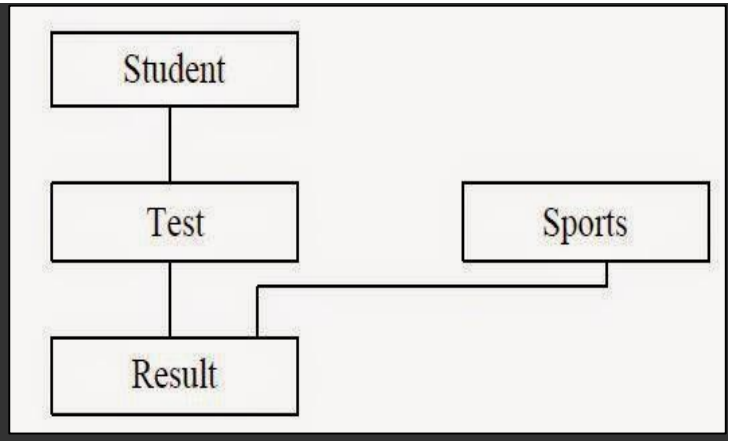
MULTIPLE INHERITANCE

```
1
2
3   #include<iostream>
4   using namespace std;
5
6   class student
7   {
8       protected:
9       int rollno;
10      public:
11      void get_number(int a)
12      {
13          rollno=a;
14      }
15      void put_number(void)
16      {
17          cout<<"The roll number is "<<rollno<<endl;
18      }
19  };
```



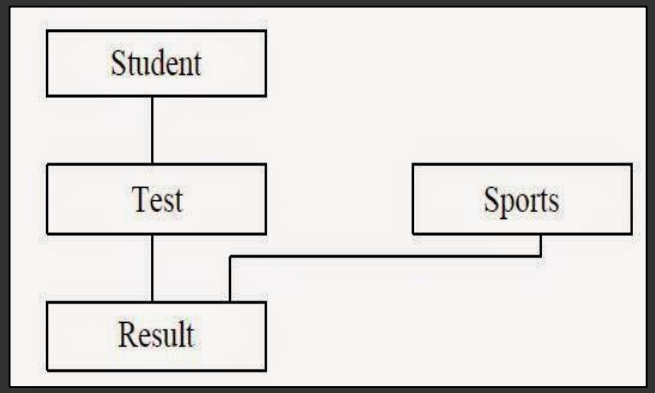



```
20
21  class test:public student
22  {
23      protected:
24      float p1,p2;
25      public:
26      void get_marks(float x, float y)
27      {
28          p1=x;
29          p2=y;
30      }
31      void put_marks(void)
32      {
33          cout<<"Marks obtained are"<<endl;
34          cout<<"Part1 marks are"<<p1<<endl;
35          cout<<"Part2 marks are"<<p2<<endl;
36      }
37  };
38
```




```
38
39  class sports
40  {
41      protected:
42      float score;
43      public:
44      void get_score(float s)
45      {
46          score=s;
47      }
48      void put_score(void)
49      {
50          cout<<"The score is "<<score<<endl;
51      }
52  };
53
```

```
53
54  class result: public test,public sports
55  {
56      float total;
57      public:
58      void display(void)
59      {
60          total=p1+p2+score;
61          put_score();
62          put_marks();
63          put_number();
64          cout<<"Total score"<<total<<endl;
65      }
66  };
67
```



```
67
68  int main()
69  {
70      result r;
71      r.get_number(1234);
72      r.get_marks(27.5,33.0);
73      r.get_score(6.0);
74      r.display();
75      return 0;
76  }
77
```

```
The score is 6
Marks obtained are
Part1 marks are27.5
Part2 marks are33
The roll number is 1234
Total score66.5
```

References-

1. Michael L Scott, “ Programming Language Pragmatics”, Third edition, Elsevier publication (Chapter-9, specifically 9.1 and 9.2)

2. Ravi Sethi, “ Programming Languages-concepts and constructs”, Pearson Education (Chapter-6)

3. NPTEL lecture series on Programming in Java, IIT Kharagpur
https://www.youtube.com/watch?v=K9gQwLeNXyw&list=PLbRMhDVUMngcx5xHChJ-f7ofxZI4JzuQR&index=8