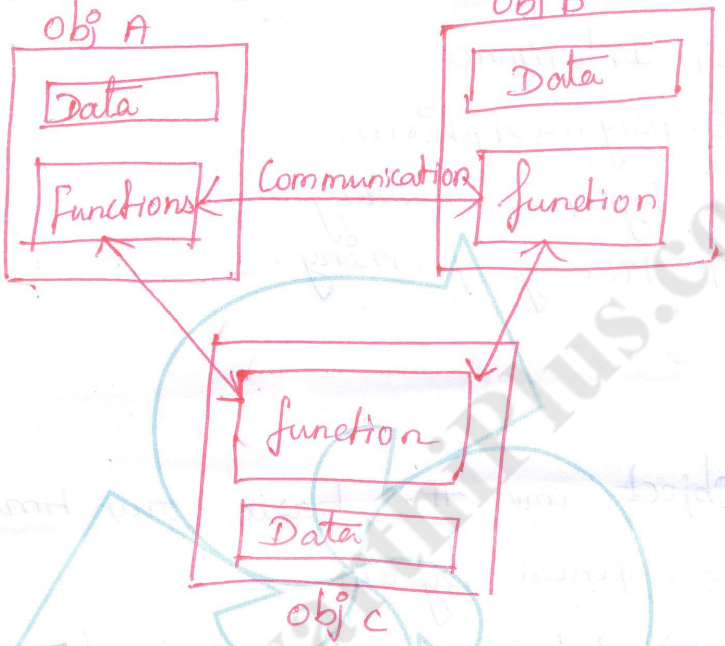
UNIT-I

Object Oriented Programming Concepts:-



1. Emphasis is on data rather than procedure
2. Pgm are divided into objs.
3. Data Structure are designed such that they characterize the
4. function that operate on the data of an obj are tie together in the data structure.
5. Data is hidden and can't be accessed by external
6. Obj may comm with each other through fun.
7. New data & fun can be easily added whenever necessary
8. follows bottom-up approach in pgm design.

Basic OOP concepts are:-

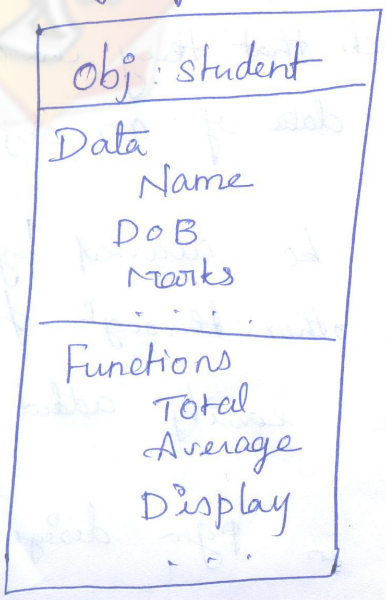
1. objects
2. classes
3. Data abstraction and encapsulation
4. Inheritance.
5. polymorphism.
6. Dynamic binding.
7. Message passing.

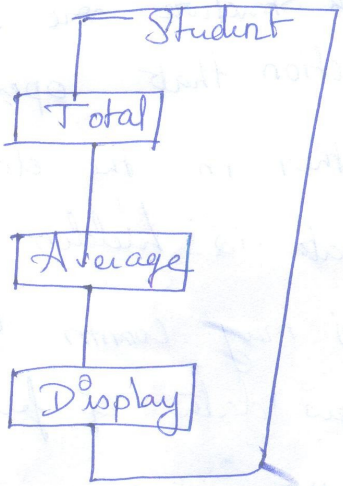
Objects:-

* object are the basic run time entities an object-oriented System.

eg → a person, a place, a bank.

Two way of representing an Object





इंडियन बैंक
Indian Bank

Define object:-

An entity that can store data and, send and receive messages. An instance of a class.

Classes:-

* Object contains data and code to manipulate th data. The entire set of data and code of an object can be made a user-defined data type with the help a class.

* Obj are variable of type class.

* class is a collection of objects of similar type

eg fruit mango;

↓

obj mango belonging to the class fruit.

Data Abstraction:-

~~The wapping up of data and function into a~~

~~Single unit is~~

Abstraction refers to the act of representin essential features without including the background detai or explaination.

→ Classes list the set of abstract attributes such as size, weight, cost and functions to operate on these attributes.

Data Encapsulation:-

The wrapping up of data and function a single unit is known as encapsulation.

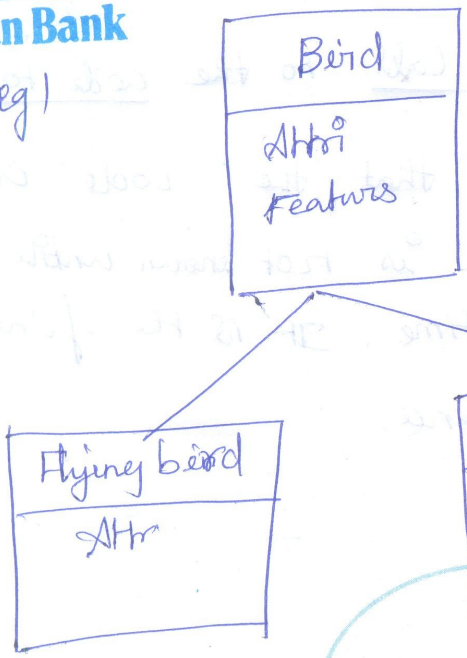
* data is not access by outside but those function can wrap the data.
* So the data are hidding for the fune

Inheritance:-

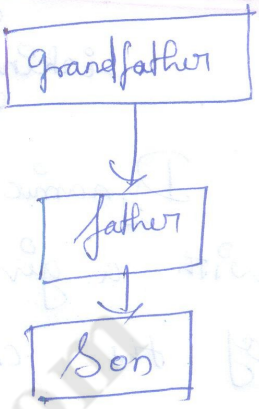
* Inheritance is the procss by which obj of one class acquire the properties of object of another class.
* Support hierarchical classification
* provide reusability.

Types : 1) Single Inheritance

eg. eg1



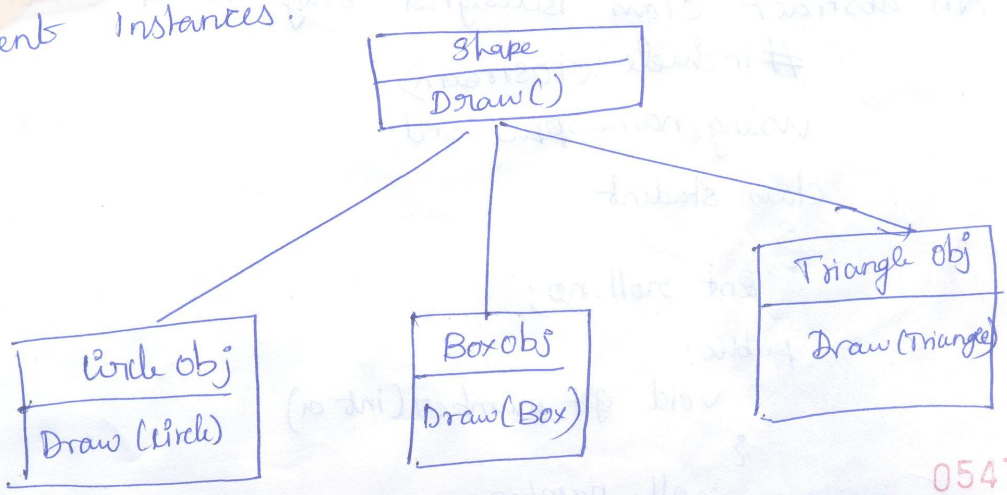
eg2



Polymorphism:-

An operation may exhibit different beha in different instances. The behaviours depends upon the of data used in the operation.

Operator overloading → exhibits different beha in different instances.



05471

Dynamic Binding :-

Linking the procedure call to the code to be ex

Dynamic Binding means that the code associa with a given procedure call is not known until the of the call at run-time. It is the function polymorphism and inheritance.

Message passing :-

Steps :-

1. Creating classes that def. objs & their behav
2. Creating obj from class definitions and
3. Establish comm among objects.

Abstract classes :-

Abstract class is not used to create objects.

An abstract class is designed only to act as a base cla

```
#include <iostream>
using namespace std;
class student
{
    int roll_no;
public:
    void get_number(int a)
    {
        roll_number = a;
```

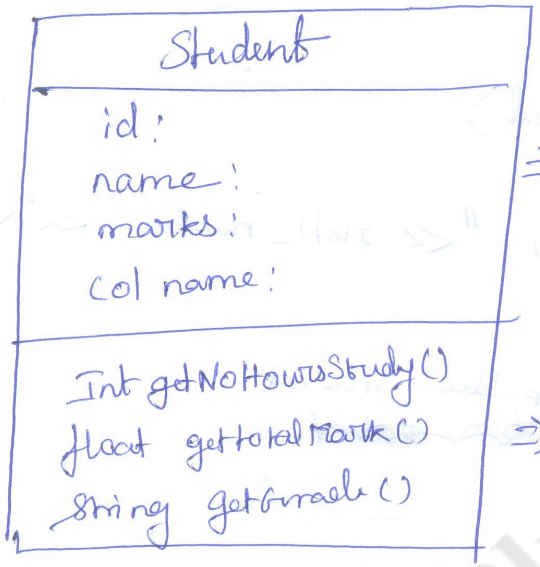
इंडियन बैंक
Indian Bank

```
};
void put_number (void)
  {
   cout << " Roll NO : " << roll_ number << "\n";
  }
};
class test : virtual public student
{
  float part1, part2;
 public:
   void get_marks (float x, float y)
   { part1 = x; part2 = y;
   }
   void put_marks (void)
   { cout << "marks obtained : " << "\n" << "part1 =" << part1 << "\
                << "part2 = " << part2 << "\n";
  }; };
int main ()
{ student student_1;
  student1 . get_number (1);
  student1 . put_number ( );
  test student_2;
  student2 . get_mark (100, 90);
  student 2 . put_mark ( );
```

0547128

CHMZ

Methods and Messages



Introduction to C# objects:

Class — bind data and its associated function together.

=> data are hidden

=> It is like a Abstract data type.

Two part

1. class declaration
2. class function definition.

general form of class declaration

```
class class name
{
    private:
        variable declaration
        function declaration
    public:
        variable declaration
        function declaration
};
```

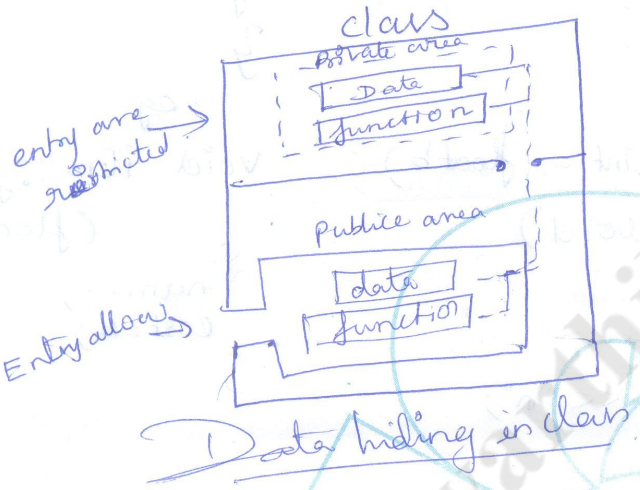
member fun. → private: variable declaration, function declaration } mem
public: variable declaration, function declaration } meth

इंडियन बैंक
Indian Bank

visibility label - private, public.

Suppose both are private, private then a class is completly hidden from the outside of the world.

member function - data member and function.



eg:
class item
{
int no ; //v
float cost; //
// fun declar public
// void getdata ()
// using prototype // Avoid putdata
}; // end with

Creating object:-
- instance of an class.

class
{
.
.
} x,y,z ;

Item x,y,z
types of items

Accessing class members:-

syntax
objname . function name (actual-argum

CHMZ

Defining Member function.

> Outside of class definition → Syntax

> inside of class definition

↓

eg

```
class item
{ int num;
  float cost;
public:
  void getdata (int a, float b);
  void putdata (void)
  {
    cout << number << "\n";
    cout << cost << "\n";
  }; }
```

Syntax

```
return type class na
    function nam (arg
{
  function body;
}
```

eg

```
void item :: get
        (float c
{ num=a;
  cost=b;
}
```

A C++ pgm with class

```
#inclued <iostream>
using namespace std;
class item
{ int n
  float c.
public:
  void getdata (int a, float b); //prototype
  // inside function
  void putdata (void)
  { cout << " number 1" << n << "\n";
    "          "    c   ";
};
```

```
// Member fun Definiti
void item :: getdata (int a
{
  number = a. // pri var d
  c = b;
}
// Main pgm
int main()
{
  itemx; // Create ob
  cout << "\n obj x"
  x.getdata (100, 299.
  x.putdata();
  item y;
  cout << "\n obj y" <
  y.getdata (200, 70a);
  y.putdata();
  return 0;
```

Inline function → a mem fun outside the class definition
then inline (qualifier is used)
↳ header line of fun def.

```
class item
{
public:
    void getdata(int a, float b); //declaration
};
inline void item :: getdata (int a, float b) // definition
{
    n = a;
    c = b;
}
```

Nesting of Mem fun:-

eg :- class set

```
{
public
    void i/p(void);
    void display(void);
}; void largest(void);
int set :: Largest(void)
{
}
void set :: Input(void)
{
}
int main()
{
    set A;
    A.i/p(); A.display();
```

Private Mem fun

eg

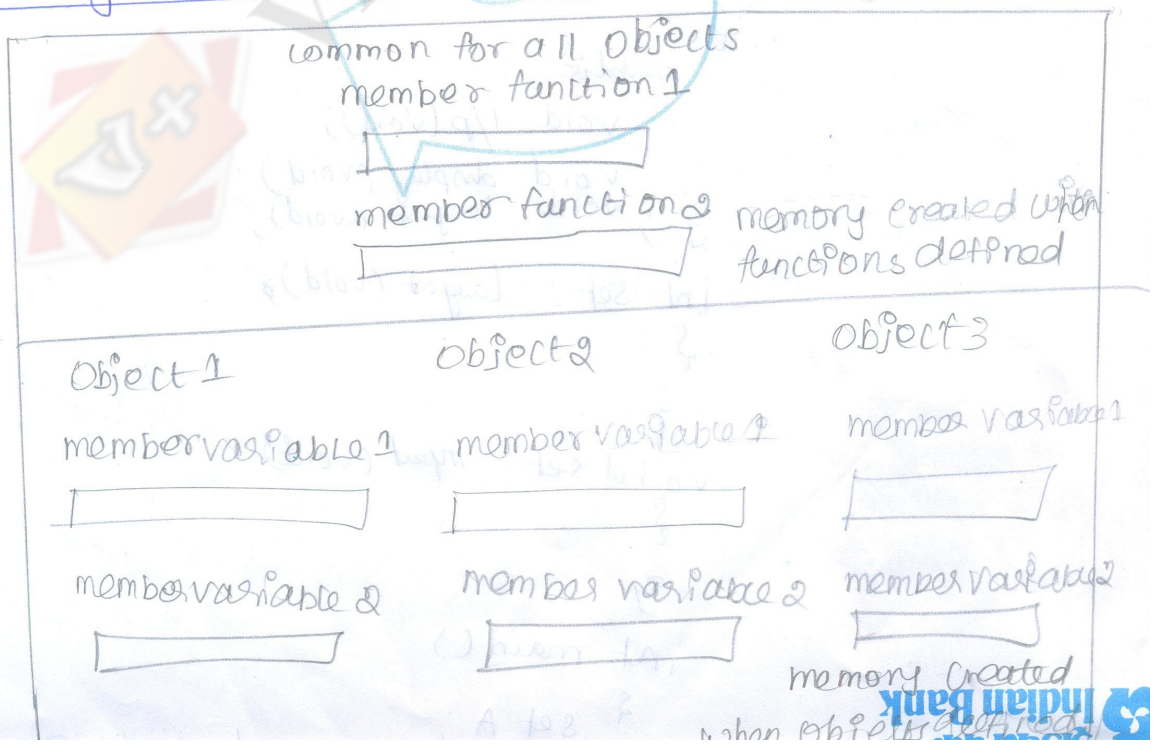
```
class sample
{
    int m;
    void read (void); // Pvt mem fun.
public:
    void update (void);
    void write (void);
};
```

S1.read → cannot accn.

car access

```
void Sample :: update(void)
{
    read (); // simple call, no obj create.
}
```

Memory Allocation for obj



ZWHC

0547126

Static Data mem → only one copy of the data is maintained for all obj of the class.

Static mem fun → Static function access → static m

Syntax

class name :: function name

using class name instead of object.

eg

```
class test.
{
  static void show co
  main()
  {
    test :: show coun
  }
```

Array of obj

eg manager[1].putdata();

↓ objects

Constructors :-

A Constructor is a special mem fun who is to initialize the obj of its class.

eg

```
Class integer
{ int m,n;
  public:
    integer(void);          // constructor dcd
};
integer :: integer(void)    // Constructor defi
  {
    m=0; n=0;
```

Constructor :

- explicitly calling method [integer int1 = integer (0,10
- implicitly " " [integer int1 (10,100)] sho

Multiple Constructors in a class.

```
class integer
{
 int m, n
 public
   integer () { m=0; n=0; } // C1
   integer (int a, int b)
     { m = a; n = b; } // C2
   integer (integer & i)
    { m = i.m ; n = i.n; } // C3
};
```

C3 } copy constructor.

Copy Constructor

eg.

```
class code
{ int id;
 public:
   code () { }
   code (int a ) { id = a; }
   code ( code & x )
   { id = x.id
   }
   void display (void)
   { cout << id;
   };
```

```
Int main ()
{ code A(100);
  code B(A) ; // copy constr called.
  code C = A // " " again
}
```

4) Dynamic Construction
5) Two dimensional "

Destructors :-

→ is used to destroy the obj that have been created by constructors.

→ Never take arg & no return value.

eg ~integer() { }.

Accepted

Devanathan
27/6/12
HOD/EEE.