①

Modular programming (close label)

Bank (Application) ⊙ It is a collection of function.

Open acc ( )
Deposit ( )
Withdraw ( )
check bal ( )
⋮
} Function

⊙ Convental programming Using high level language.

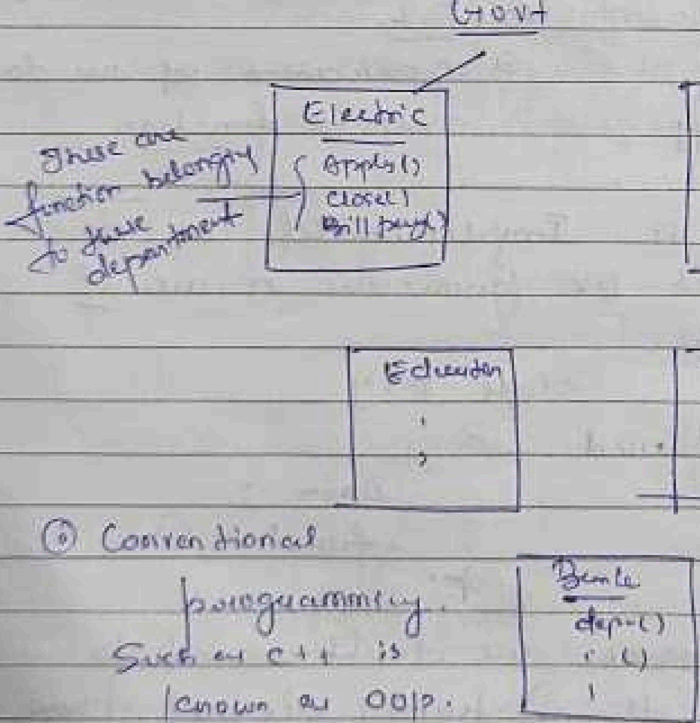
⊙ A no. of function are Written to accomplish these tasks. Application.

⊙ Large program are divided into small chunks known as function

⊙ Function transform data from one form to another.

Object Oriented programming. (Fact available)

Ex:-



⊙ It is a collection of set of object
⊙ Each object have it's relavent function
⊙ as well as data related to these function.

⊙ Conventional programming. Such as C++ is known as OOPS.

⊙ Decompose it problem into no. of Entities known as object.

Application

- Emphasis is on data rather than procedure.
- Data is hidden and can't be ~~[illegible]~~ accessed by external function.
- Obj may communicate with each other using function.
- New data & function can easily added whenever necessary.

Principle of object - oriented programming

- Abstraction.
- Encapsulation.
  - Data Hinding
- Inheritance
- Polymorphism.

Abstraction
↓
When we don't ~~want~~ ~~to~~ know the internal detail How is it working is known Abstract

Two elements are here for program
- data
- Operation of the data i.e function.

Mean:- How it is Implemented we don't know just we know the name of function.

Ex:- printf function we had used so many time but we don't know how it is working (which is hidden from ~~[illegible]~~ us) i.e Abstraction.

Ex:-
```
class My
{
        Data ;
        function ();
};
```

Here we don't want to know the Internal detail How is it working we just have to know the function name

And we can group the set of function with the help of class.

③

Encapsulation

NOTE :- class given two things i.e. Abstraction also and Encapsulation also.

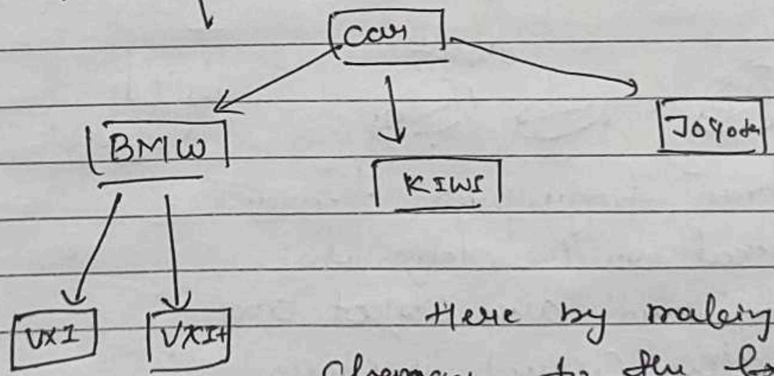
- The wrapping / binding the data and the function into a single component is known as Encapsulation.
- The data isn't accessiable by outside function.
- Only those function are able to acces who are defined inside the class.

we use private, public just ~~&~~ not to misshandling the data.

EX:- Class Car
{
private:
Data
public
function();
};

Here,
we hide the data
and show the function
so, we do data hiding
also with Encapsulation.

Inheritance



Here by making some changes to the ~~[illegible]~~ exesting car they are making some new car that is ~~for~~ Nothing but Inheritance.

- It is the method by which object of one class get the properties of object of another class.

- The programmer can add new properties to the existing class without changing it.

- The new class possess features of both class.

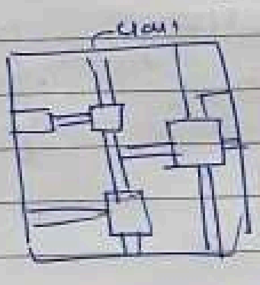
polymorphism:-

It is derived from Inheritance - -

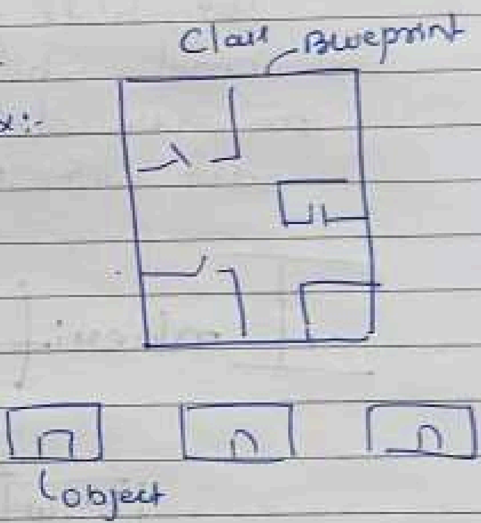
- Ex:- If you know how to use a browser do you need to learn how to use chrome, Internet Explorer, (no)
  Here thing's are common.

Class & object

Ex



Ex:-



All these fingerprint scanner are based on this design so fingerprint are Object and the Circuit is Class

- A class is grouping of object having ~~identical~~ identical properties, Common behaviour & shared relationship.

* ⊙ Class will Contain data and function, So data is called property and function is Called as behaviour.

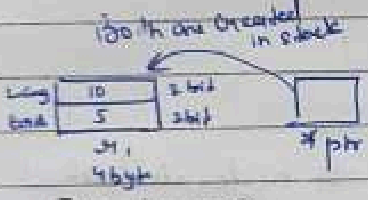
⊙ Object are nothing but variable of type class.

Ex:-

```
class Rectange
{
data <  float length;
        float breadth;
function < float area();
           float perimeter();
           :
};
Rectange r1, r2, r3
         └object.
```

Ex:-

```
# Include <iostream>
Using namespace std;
class Rectange {
    public:
        int length, breadth;
            int area()
            {
                return length * breadth;
            }
            int perimeter()
            {
                return 2 * (length + Breadth);
            }
};
```



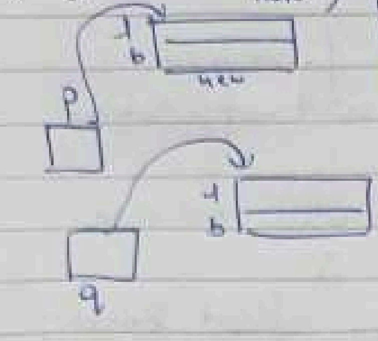
Pointer on Obj rectangle

```
int main()
{ rectangle r1;
  Rectange *ptr;
   ptr = &r1;
  ptr -> length = 10;
  ptr -> breadth = 5;
  Cout << ptr -> area();
  Cout << ptr -> perimeter();
}
```

~~or~~

```
int main ()
{ rectange r1;
      r1. length = 10;
      r1. breadth = 5;
      Cout << "Area is" << r1. area();
      Cout << "perimeter is" << r1. perimeter();
}
```

(object is created in heap here)



How to create an object on Heap using pointer

```
Void main ()
{ ~~rectangle *~~
  Rectangle *ptr = new Rectangle();
  ptr -> length = 10;
  ptr -> breadth = 5;
  Cout << ptr -> area() ;
  Cout << ptr -> perimeter () ;
}
```

Output → 50
30

Rectangle r ; → Created in stack
Rectangle *p = new Rectangle();
↳ Created in Heap

NOTE :- In java we can not create a object in Stack but in C++ ✓
C++ give more option to programmer

Data Hiding

It is a technique used in OOP to hide internal object detail (data member)

If we use public: in Data member then, the user can make changes in those so, to avoid it we privat the data member and public: the function member. Just to call that function.

By Doing private : the data member we can't read or write the data member Only we can use the function to read and write.

Ex:- class Rectangle

```
{ private :
        int length;
        int breadth;
  public:
        Void ~~g~~ setLength (int l)
        { if (l>=0)
              length=l;
          else
              length = 0;
        }
        void SetBreadth (int b)
        { if (b>=0)
              breadth = b;
          else
              breadth = 0;
        }
        int getLength ()
        { return length;
        }
        int get~~Length~~ Breadth ()
        { return breadth;
        }
};
```

property fun^c
Accessor - getxx
Mutator - Setxx

```
Void main()
{ Rectangle r;
   r.setlength (10);
   r. setBreadth (5);
Cout << r.getLength ();
```

Constructor

It is a special member function whose task is to initialise the object of its class.

void main {
} rectangle r1 (10,5)

Value is initialise with the object of it's class.
(bcz we don't get the box with Garbage Size)

(color-white
At ~~the~~ time of order we describe the color)

# There are three type of user-defined Constructor we define the Constructor inside the class

User defined [
- ⊙ Non parameterized
- ⊙ parameterized
- ⊙ Copy constructor

Buildin Constructor ⊙ — ⊙ Default Constructor.
defined by Compiler

⊙ They should be declared in the public section.

⊙ A Constructor is a function which will have the same name as the class name.

⊙ They do not have any return type

Syntax:-

```
                      class rectangle
                      {  private:
                         int length;
int main ()              int breadth;
{                        public:                NON-parameterized
  rectangle r;           rectangle ()
  rectangle r ();        {  length = 0;
                            breadth = 0;
                         }                      parameterized
rectangle r(10,5);
rectangle r2(r);         rectangle (int l, int b)
                         {
                            setlength (l);
                            setBreadth (b);
                         }                      Copy Co-
                         rectangle (Rectangle & r)
                         {
                            length = r.length
                            breadth = r.breadth
                         }
```

Both will work on non-parameterized

Here,
rectangle function using again & again that is function Overloading

① ~~when an object is declared~~ a
Copy constructor is used to initialize the member of a newly created Object by copying the member of an already existing object.

① It take a reference to an object ~~through a~~ copy. of the same class as an argument.

```
Sample (Sample&t)
  {
        id = t.id;
  }
```

problem on copy constructor

If there is an any dynamic memory allocation done by an object then the copy constructor may not create a newmemory for it, It will pointing on the same memory.

Ex:-

```
        # Include <iostream>
      using namespace std;
   Class Rectangle
     {  private:
              int length;
              int breadth;
        public:                    (and it doesn't have
                                     any return
           rectangle ()            // constructor   type)
Default       {    length = 1;
constructor        breadth = 1;
              }
Parameterised rectangle (int l, int b)
Constructor {
                  setlength (l);
                  setbreadth (b);
              }
```

Copy Constructor

```
rectangle (rectangle &r)
{ length = r.length;
  breadth = r.breadth;
}

void setlength ( int l)
{
        if (l>0)
            length =l;
        else
            length=1;
}

Void ~~get~~ setbreadth (int b)
{
      if (b>0)
          breadth = b;
      else
          breadth = 1;
}
int getLength ( )
    { return ~~L~~length; }
int getbreadth ( )
    { return breadth; }
int area ( )
    { return length * breadth; }


int main ()
{
   // rectangle r1 ;
      rectangle r1 ( 10,5) ;
   // Cout << r1.area () << endl;
      Cout << r1. area () << endl;
}
//Copy Constructor
        Rectangle r1 (10, 5);
        rectangle r2 (r1) ;
        Cout << r2. area () << endl;
```

Types of functions in class

⊙ The member function defined inside the class are treated as <u>inline func</u>.

If the member function is small then it should be defined inside the class. Otherwise it should be defined outside the class. by using scope resolution and its prototype declaration must be done inside the class.

⊙ The member function defined outside the class can be made inline by prefixing the keyword inline to function declarator.

```
Class Rectangle
{ private:
      int length;
      int breath;
  public:
                     ┌ Rectangle ();
      Constructor    │ Rectangle(int l, int b);
                     └ Rectangle(rectangle &r);
      Mutator or     ┌ void SetLength(int l);
      Setter method  └ void St breadth (int b);
      Acessor or     ┌ int get length ();
      getter method  └ int getbreadth ();
      facilitator    ┌ int area();
                     └ int perimeter ();
      Inspector      [ int issquare ();
      function         which return true or false

      Destructor     [ ~Rectangle ();
      function
};
```

Scope Resolution Operator (::).

```
Class rectangle
{   private:
        int length, breadth;
    public:
        int area()
        {   return length* breadth;
        }
        int perimeter ();
};

int rectangle::perimeter ()        — Scope Resolution.
{                                    it scope the scope of
                                     this function is within this class
    return 2*(length + breadth);
}
```

(12) ** How to declare the function Inside the class and implement their body outside the class by using ::

```
#Include <iostream>
 using namespace std;
  class Rectangle
  { private :
          int length;
          int breadth;
     public :
          Rectangle();
          Rectangle (int l, int b);
        Rectangle (Rectangle &r);
          int getLength()
            { return length; }
          int getBreadth ()
              { return breadth; }
     void setLength (int l);
     void setbreadth (int b);
       int area ();
       int perimeter ();
       bool issquare ();
         ~rectangle ();
  };
 int main ()
  {
        rectangle r1 (10,10);
   Cout << "Area is" << r1.area ();
        if (r1.issquare ())
                cout << "yes it is";
   }
   Rectangle :: Rectangle ()
    {
        length = 1;
        breadth = 1;
    }
 Rectangle :: Rectangle (int l, int b)
       { length l;
         breadth b;
       }
```

```
Rectangle :: Rectangle (
   rectangle &r)
  {
   length = r.length;
   breadth = r.breadth;
  }
 Void Rectangle ::
      SetLength (int l)
   {
     length = l;
   }
 Void Rectangle ::
      SetBreadth (int b)
   {
      breadth = b;
   }
 int Rectangle :: area ()
   {
    return length * breadth;
   }
int Rectangle :: perimeter ()
  {
   return 2* (length + breadth);
  }
bool Rectangle :: issquare ()
   {
    return length = Breadth;
   }
Rectangle :: ~Rectangle ()
   {
   Cout << "Rectangle
        Destroyed";
   }
```

(13)

Inline function

The function which are expand by in the Same line where they are called. There is no seperate block for that function.

If you define a function inside the class then it is inline
Or
If you define Outside then it is non-inline

If you want to make Outside function inline define the function inside the class like :-
inline void fun();

```
Class Test
{ public:
    void fun1()
    {
      cout << "Inline";
    }
    void fun2();
};
void Test :: fun2()
{   cout << " Non-Inline";
}
int main()
{   Test t;
    t.fun1();
    t.fun2();  }
```

Structure

```
Struct Demo
{ int x;
  int y;
  void Display()
  {
    cout << x << endl;
  }
};
```

Difference b/w structure of 'c' and c++ is (in c++) we use function inside the Structure but in c not

In c++ structure looks like a class

• But in class Demo
{
} all data member and member function are private
But in Structure everthing is public Bydefault.