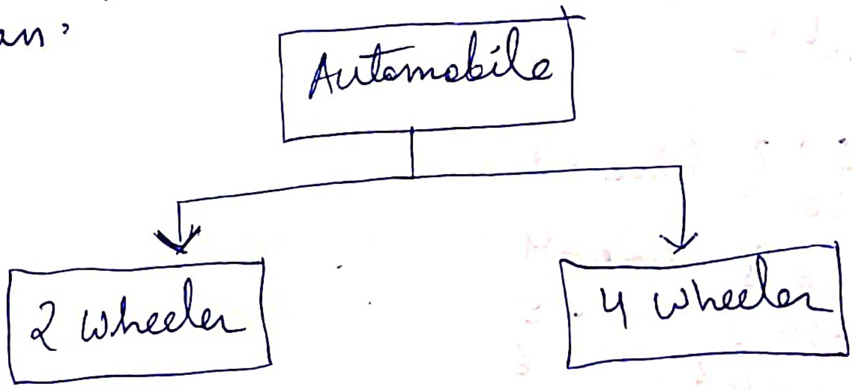
Defining Class functions

Features of OOPs

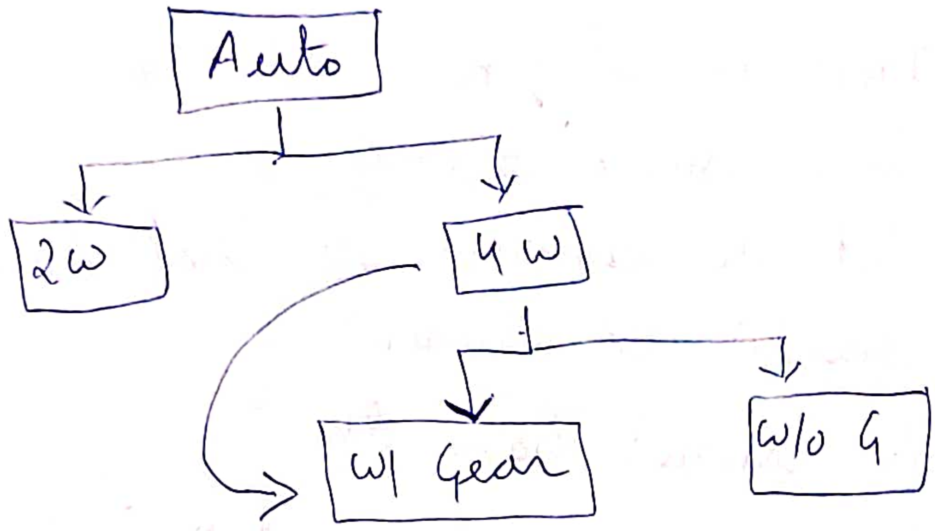
1) Encapsulation - means to combine var & fun into classes.
The word comes from capsule which is a collection of 1 or more medicine
Similarly, class is a collection of var & fun.

2) Inheritence :-
- Classes can be inherited. There will be a base class & other classes will inherit or acquire the properties of this base class.



- Suppose these classes are already created. If you want to ~~inherit~~ create a class for 4 wheeler with gears, you can simply copy code of 4 wheeler & add your own code.

For eg



- Simply copy the code of 4wheeler & add your own code to this.
- So you don't have to write code again. ~~So~~

- The benefit of inheritance is Resusability
  - You can acquire the code of base class like automobile and add your own code to it.

3) Polymorphism :-
Poly means multiple
morphism " shape.

- It means same name can have multiple use.
- Or Same name can be used for multiple fun.

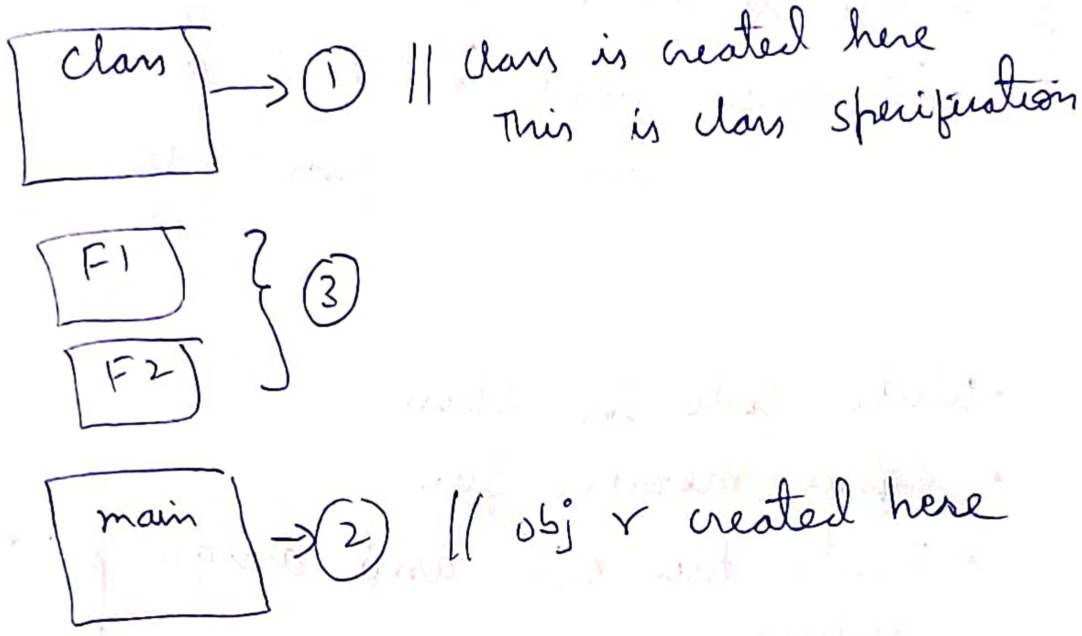
For eg:- sum(int x, int y);
sum (int x, int y, int z);
sum (float x, int y);

- There are different fun, but they all have same names.
- But, the condition is that parameters should be different.
- You cannot have this :-

```
sum( int x, int y);   } both have
sum( int a, int b);   }  same
                         param.
```

Defining Class Fun

- The code is divided into 3 regions:-



- The third region is defining ~~member~~ member fun.
  Here you will write the code for functions that you are using in your class.
  For eg. code for set(), get(), add() will be written here.

- Now let's see the complete code for my class & student.

- To define a fun, you will write it like this:

```
void <class name>::<fun name>(<Parameters>)
{
        // fun code

}
```

~~class myclass~~

Q Create a class named myclass.
It shud have 2 var : a, b
" " have 3 fun ; set ( )
get ( )
add ( )

- Write code for class
- define member fun
- Create two obj and assign, print, add values.

A

```
class myclass
{
        private:
        int a, b;
        public:
        void set (int x, int y)
        void get ();
        void add();

};
```

//class

```
void myclass :: set(int x, int y)
{
            a = x;
            b = y;
    }
```

fun-1

```
void myclass :: get ( )
{
        cout << a << endl << b << endl;

}
```
Fun-2

```
void myclass :: add ( )
{
        cout << a+b << end;

}
```
Fun-3

```
int main ( )
{
        myclass obj1, obj2;
        obj1.set (10,20);
        obj1.get ();
        obj1.add ();

        obj2.set (100,200);
        Obj 2.get ();
        obj 2.add ();

}
```
//Creating & using objects

- Notice how you have defined fun-1 fun-2 & fun-3.
  You will follow same pattern now onwards.

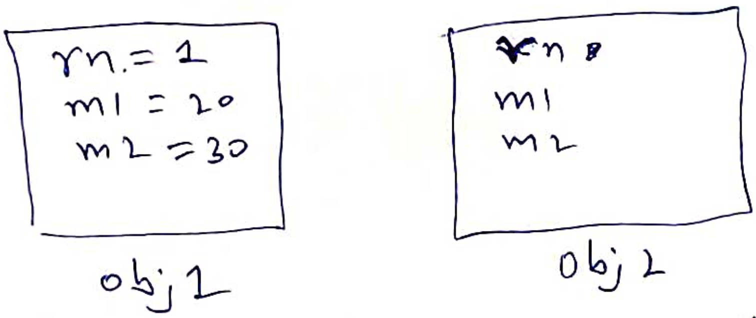
Q2 Create a class student.
It will have 3 var - Rn → roll no.
m1 → marks1
m2 → marks2

It " " 3 fun → set() → assign values
get() → Print
add() → add m1 & m2;

- Create class
- Write fun definitions
- Create 2 obj

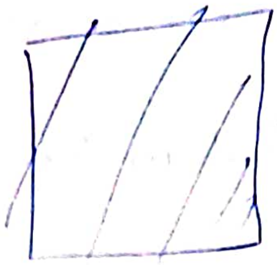
Ans
```
class student
{
    private:
    int rn, m1, m2;
    public:
    void set(int x, int y, int z)//set fun
    void get ();
    void add ();

};
```
has 3 parameter bcoz there are three var.

```
void ~~sd~~ student :: set (int x, int y, int z)
{
        rn = x;        // Assign values
        m1 = y;
        m2 = z;
 }

void student :: get ()
{       cout << rn << m1 << m2; // Print
                                    values.

 }
void student :: add ()
 }      cout << m1 + m2;      // add
                                  marks.

  }


int main ()
{
       student obj 1, obj 2; // No O/Put.
       obj 1. set (1, 20, 30) //
```



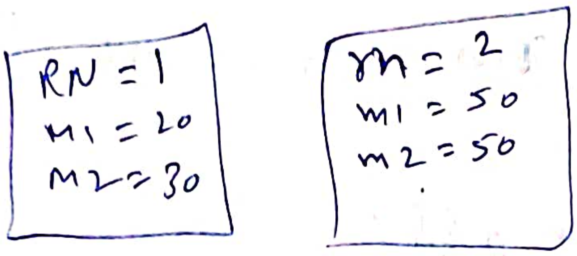
```
       obj 1. get ( ) // Print 1, 20, 30
       obj 1. add ( ) // 50.
```

~~Obj~~



obj1

```
obj2.set(2, 50, 50); //No o/p
```



obj1 obj2

```
obj2.get(); // Print 2, 50, 50
obj2.add(); // Print 100.

}
```

o/p is :-

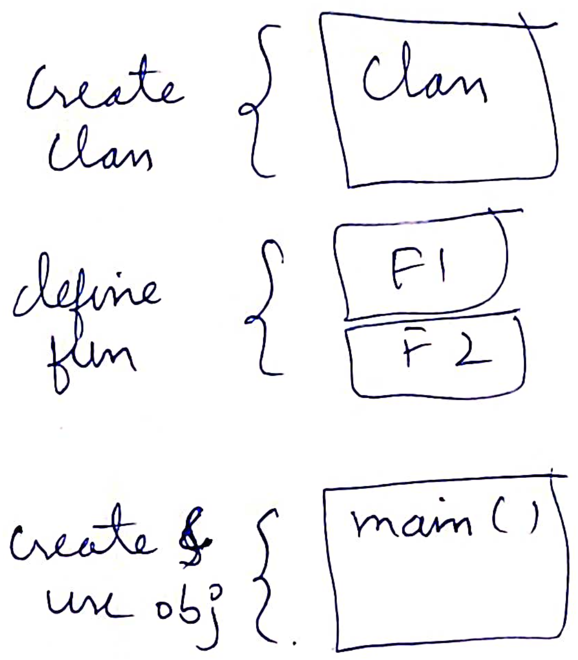
```
1
20
30
50
2
50
50
100.
```

Note :- There will be no output when you create obj & use set fun.

:- o/p will be ~~f~~ shown only for get() & add() fun.

Summary :-

Whenever you create a code, you will follow these steps :-



Practice Question Create a class ~~CalculateIt~~ calculator. It should have

three variables :- a, b, c.

6 functions :- set ( ) → to assign
get ( ) → print
add ( ) → add
mul ( ) → mul
div ( ) → div
sub ( ) → sub

- You can add, mul, div, sub these var in any order.
- Write – code for Class
  – define member fun (all 6)
  – Create two obj & perform these tasks : assign, print, add, sub, mul, div.