Friend function & Returning objects.

Q What is a FF? How do you declare a FF

A • A fun which is not a member of a class but still can access the variables of that class.

. Use friend keyword to declare a FF

Q When is a FF used? Why do you use a FF?

A • When you want to perform operations on obj belonging to different classes.

• In earlier example of obj Passing, all the three obj were of same class

i.e obj 3. add_obj (obj1, obj 2);

obj1, 2, 3 are of same class.

• This fun can't perform operations on obj belonging to diff classes.

Q How many types of fun r there?

A Member fun → wh. is a part of a class

Friend fun → ~~wh.~~ is not a part of a class, it is only a friend of a class.

Q Why don't you use a member fun (MF) to perform operations on obj of diff Classes?

A. Bcoz, MF of 1 class can't access var of obj from another class. Whereas, a FF can access var of obj from different classes.

Q Can a fun be a MF of 1 class & FF of another class?

A. Yes, it is not necessary for a fun to be FF of all classes. It can be a MF of 1 class & FF of all other classes.

Q Diff b/w FF & MF?

A 1) MF is called by using obj

eg a) obj3. add_obj (obj1, obj2);

FF is called w/o using obj

eg b) obj3 = add_friend (obj1, obj2);

2) In MF, all obj shud be of same clan.
obj3, obj 2, obj 1 in eg a) are of same Student clan.

In FF, obj can be of diff classes.
obj3, 2, 1 in eg b) are of diff. classes.

3) FF is not part of the clan to wh. it is a friend fun.
MF is a part of the clan to wh. it belongs.

Understanding different types of objects.

1) Calling obj
2) Passing obj
3) result obj (temp. obj)
4) returning obj
5) Assigning obj

1) Calling & assigning obj.

- In OP, we have seen a calling obj

$$\underbrace{\text{obj3}}_{\text{Call'g obj}}\text{. add\_obj}(\underbrace{\text{obj1, obj2}}_{\text{Pass'g obj}});$$

- In FF, there is no calling obj So assigning obj is used

$$\underbrace{\text{obj3}}_{\text{assig obj}} = \text{add\_friend}(\underbrace{\text{obj1, obj2}}_{\text{Pass}^{n}\text{ obj}});$$

2). Returning obj

- The fun will return an obj & it will be copied to the assigning obj

$$\text{obj3} = \underbrace{\text{add\_friend}(\text{obj1, obj2})}_{\text{returning obj}};$$

3) Result obj

- Inside fun, create a temporary result obj.
- This result obj will be returned by the fun.
- add var of obj1 & obj2 & copy to this result obj.
- This result obj shud be of same class as obj3.

Steps to create a FF

- There can be more than 1 classes. Suppose there r 3 classes - subject 1, subject 2 & total.

```
Class subject 2; // forward declaration
Class total;
Class subject 1
{
      ≡
  ~~total~~
  friend total add_friend(subject 1 P01,
                          subject 2 P02);
  // ~~ad~~ write friend fun in Class-1
};
// write fun of Class-1 except FF.
//    :    :

Class subject 2
{
      ≡
  friend total add_frid (subjet 1 P01
                         sub 2 P02);
  // declare FF in
}; // each class

// write fun of class-2 except FF
      ≡

Class subject 3.
{
      ≡
  friend total add_frid (subjet 1 P01
                         subjet 2 P02);
}; //
// write fun of class-3. except FF
```

//After writing all fun of three classes
//. the FF definition is written.

```
total ~~for~~ add_friend (subject1 po1,
                          subj2 po2)
{
        -
        -
        -
        -
        -
}.

int main()
{
    obj3 = add_friend (obj1, obj2);
             //call FF.
}
```

1) **How to declare a FF**

Take eg of set fun

void student :: add (int x, int y. int z);

↓ ↓ ↓

total X add_friend (subject1 po1, subject2 po2);

↓ total → ✱ F.F has a return type of total class

↓ X → It doesn't belong to any class

↓ add_friend → Name of FF

$\underbrace{\text{subject1 po1, subject2 po2}}$ → PO sud be of different classes.

2) How to define a FF definition

```
total add_friend (subject 1 po1,
                  subject 2 po2 )
{
        -
        -
        -
        -
        -
        -
}.
```

1) First write the fun call : obj3 = add_friend(obj1, obj2);

↓ ↓

P01 P02

2) Change names of pass & obj. :

↓ ↓

3) Write their var. : P01.theory P02.theory
P01.prac P02.prac

4) Create a temp obj called result. It sud be same class as obj3 : total result;

5) add P01 & P02 & copy to result : result.theorytotal = P01.theory + P02.theory;

result.practotal = P01.prac + P02.prac;

6) return result : return result;

Steps 4, 5, 6 will be written inside FF definition.

FF definition :-

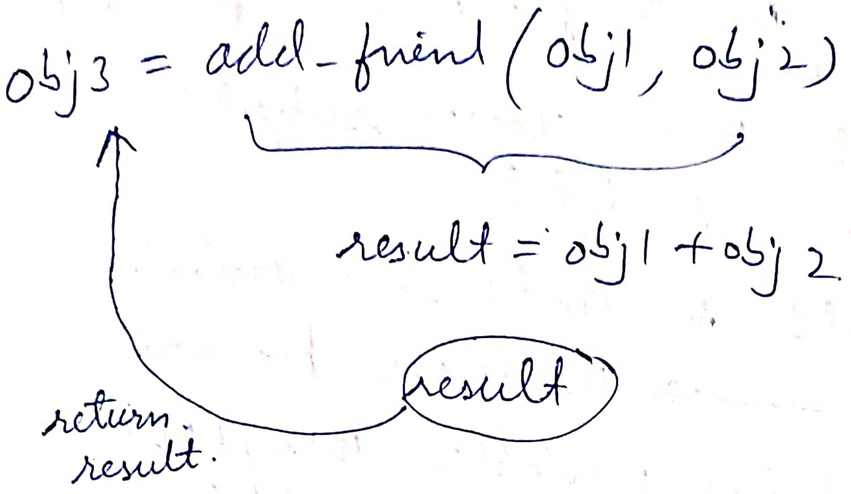
```
total add-friend (subject1 P01,
                  subject2 P02)
{
    total result;
    result.theorytotal = P01.theory +
                         P02.theory;
    result.Practotal = P01.prac +
                       P02.prac;
    return result;
}.
```

Summary

1) ~~We need~~ obj3, obj2, obj1 belong to diff classes.

2) We need to add obj2 & obj1 & copy to obj3.

3) It will be done in 2 steps inside a FF.
   - Obj1 & 2 will be added to a temp result obj.
   - This result obj will be returned & copied to the assigning obj i.e obj3.


obj3 = add_friend(obj1, obj2)
result = obj1 + obj2
result
return result.

The Complete Code for FF

Q Create three classes subject 1, subject 2 & total.

- Subject 1, & subject 2 has variables ~~xn, m1, m2~~ theorymarks, pracmarks
- total has variables theorytotal, practotal;

Create there obj

obj1 of subject 1 → (99,98);
obj2 of subject 2 → (1,2);
obj3 of total → blank.

Q/add objective:
add marks of obj1 & obj2 & store in obj3.

Answer

```
#include <iostream>
using namespace std;
class subject 2;  //forward declaration
class total;

class subject 1
{
    private:
    int theorymarks, pracmarks;
    public:
    void set(int, int);
    void get();
  friend total add_friend(subject1 Po1
                          subject2 Po2);
};
```

```
//write fun of subject 1  except FF
void subject 1 :: set (int x, int y)
{     theory marks = x;
        prac marks = y;
}
void subject 1 :: get ()
{     cout << theory marks << prac marks;
}

class subject 2
{       private:
        int theorymarks, pracmarks;
        public:
        void set (int x, int y);
        void get ();
        friend total add_friend(subject 1 p01,
                                subject 2 p02)
};
//write fun of subject 2.
void subject 2 :: set (int x, int y)
{       theorymarks = x;
          prac marks = y;
}
void subject 2 :: get ()
{       cout << theory marks << prac marks;
}
```

```
class total
{
    private:
    int theorytotal, practotal;
    public:
    void set(int, int);:
    void get();
    friend total add_friend(subject1 p01,
                            subject2 p02);
};
//write fun of total class;
void total::set(int x, int y)
{
     theorytotal = x;
     practotal = y;
}
void total::get()
{
     cout<< theorytotal << practotal;
}
//friend fun shud be written at
//last after all classes.
total ~~fr~~ add_friend(Subject1 p01,
                    subject2 p02)
{
     total result;
   result.theorytotal = p01.theorymarks +
                        p02.theorymarks;
   result.practotal = p01.pracmarks +
                      p02.pracmarks;
   return result;
}
```

```
int main ( )
{
        subject1 obj1 ;
        subject 2 obj2 ;
        obj1.set (99, 98);
        obj2.set (1,2);
        total obj3;
        obj3 = add_friend (obj1, obj2);
        obj3-get (); // Print details of
                    //   obj3 to check
                    //  whether marks r
                    //  added or not;
                    //
}
```