object oriented programming

OOP → by C++

- Class
  الكلاس
- encapsulation
  الكبسلة
- Inheritance
  الوراثة
- Polymorphism
  البلمرة

C → C++ → C with class

البرمجة الشيئية

أو البرمجة الكائنية

object → شئ →

- صفات ↓
  color → white
  hair
  eyes
  ear
  عبارة عن متغيرات variables
- مثال: قط
- أفعال ↓
  eat
  run
  sleep
  dreem
  عبارة عن دوال function

عندى 5 عربيات لكل عربية سعر لون واسم

اكيد هعرف لكل عربية ← ٣ متغيرات

∴ 5 عربيات ← ٣×٥ = ١٥ متغير

```
int price1 = 1000              | int price2 = 5000              | ..... etc
string name1 = "Bmw"           | string name2 = "Honda"         |
string color1 = "black"        | string coror3 = "red"          |
```

أكيد مش هينفع أعمل كدا لو عندى ألف متغير

← نحتاج لـ **Class**

```
#include <iostream>
using namespace std;

class car
{
    Public:-
        int Price;
        string name;
        string color;

};
int main()
{
    Car obj;
    car ob2;
    car ob3;
    car ob4;
    car ob5;
    Cout << " xxxxxxxxxx " <<endl
    obj.Price = 5000;
    obj.name = "Bmw";
    obj.color = "Black";
    cout << "Price is:" << obj.price << endl;
    cout << "name is: " << obj.name << endl;
    cout << "color is: " << obj.color << endl;
    cout << "**********" << endl;
// make this with All object ob1, ob2, ob3, ob4
                                  ob5
    return 0;
}
```

يفضل بدء Class بحرف Capital وهذا هو الاتفاق عليه بين المبرمجين

Constructor

```
Class Book
{
  Public:
    int page;
    string name;
    void print()
    {
      cout << name << endl;
      cout << page << endl;
    }
    Book()
    {
      cout << "I'm a constructor" << endl;
    }
}
```

member function

دالة جوه Class لكن أي Function لا تسمى member function

constructor

عبارة عن شهادة ميلاد لل object أول ما يتعمل run للبرنامج أو الكود هيتنفذ تلقائياً constructor من غير ما نستدعي ال constructor

Encapsulation

- Public & Private & Protected
- default constructor
- Parameterized constructor
- destructor
- overloading function
- function with object
- friend function

Default constructor

Class Book {

default constructor

```
Book()
{
  cout<<"Hi I am a default constru
  <<"\n";
}
```

```
Book(int x)
{
  cout<<"Hi I am a Parameterized construct
  <<"\n";
}
```

Parameterized Constructor

};

```
Class Test ()
{
    Test ()
    {
    }
```

→ **constructor**

لا يمكن استدعائه من int main()
لانه يتم تنفيذه عند بدء البرنامج

البناء
شهادة ميلاد لل object

```
    ~Test ()
    {
    }
}
```

هدم ← ~Test ()

لابد من الاستدعاء من int main()
ولتكن obj1.sample()

→ **destructor** الهدم

لا فائدة ← يمكن مسح القيم بعد اعطاء قيمه ابتدائية

هدم

مقارنة

	constructor	destructor
1	لا يوجد فيه كلمة (return type) يعني ميتفعش احط قبل constructor او destructor كلمة مثل int او void او...	
2	ياخذ parameter عادي معاملات (int x و int y)	ميتفعش ياخذ معاملات مستحيل
3	ممكن تكون فيه اكثر من constructor في class	لا يمكن أن تكون هناك أكثر من destructor في class
4	Book() {} ... constructor	destructor ... ~Book() {}

overloading function

يعني التحميل
أكثر من function لها نفس الاسم
ولكن تأخذ معاملات مختلفة

مثال

```
void print ( )
{
  cout << "salah hassan" << endl;
}
```

```
void print (int x )
{
  cout << "salah hassan" << endl;
  cout << "I am an overloading function" << endl;
}
```

ازاى اعرف دالة بدا الـ class

```
class cal {
private:
    int x=0;
    int y=0;
Public:
    cal ()
    { x = 20;
    y = 30;
```

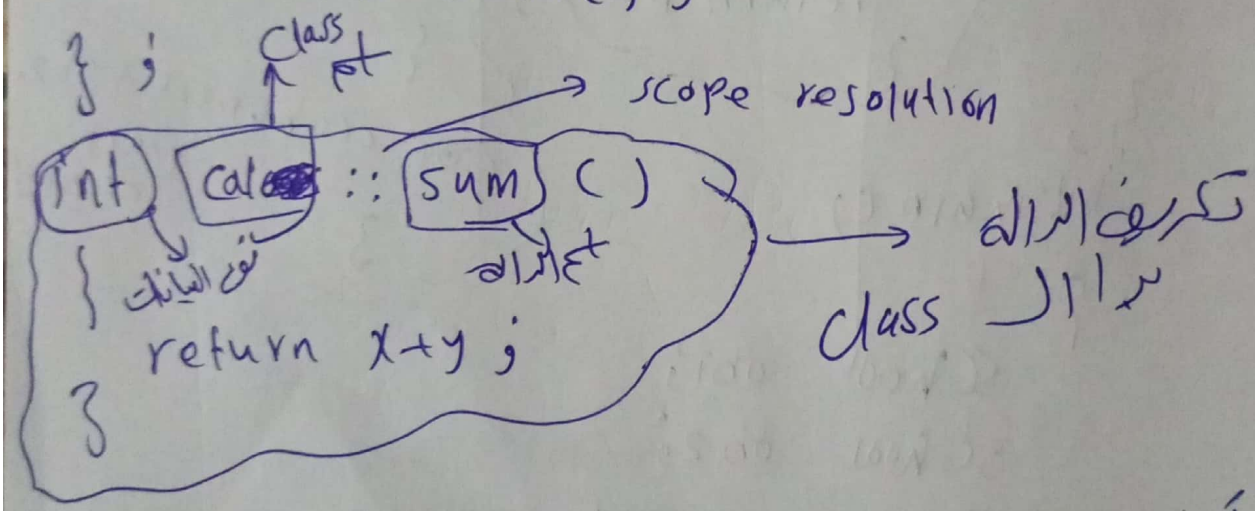
class دالة جوا ↓

```
int sum()
{
}
return x+y;
```

class كيفية تعريف الدالة sum بـ الـ

```
Class  Cal
{
   Private
   int X=0;
   int y=0;
   Public::
     CaL()
     {
       X=20;
       y=30;
     }
     int sum();
};
```

constructor



يمكن تعريف محتوى الـ constructor بـ الـ class بره

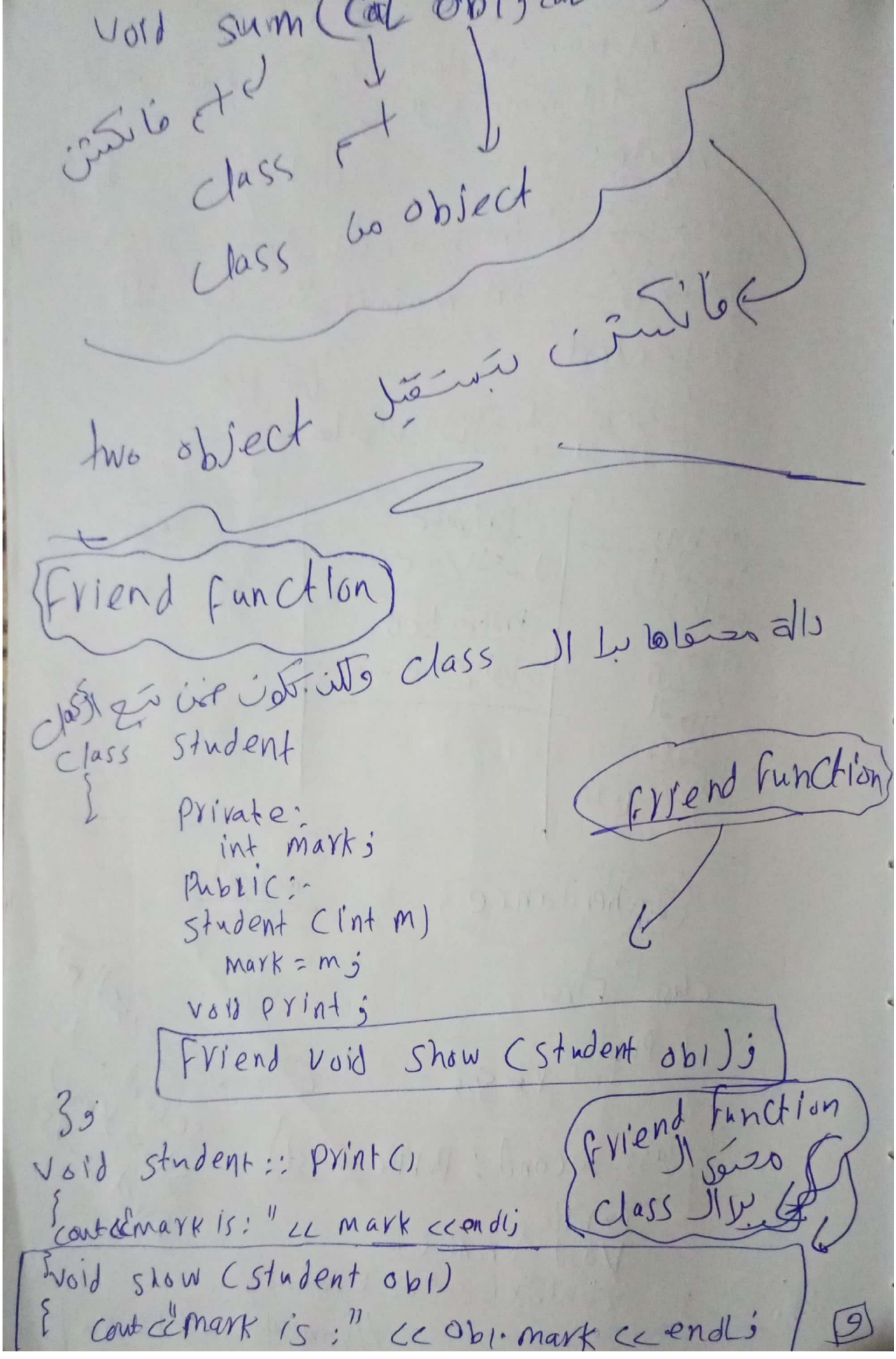
```
Cal::  CaL()
{
   X=20;
   y=10;
}
```

Function with object

```
Class School
{
    Private:
        int mark
        string name
    Public:
        School (string n, int m)
        {
            name = n;
            mark = m;
        }
        int getmark()
        {
            return mark;
        }
};
int main()
{
    School ob1;
    School ob2;
    School << ob1.getmark() + ob2.getmark() << endl;
    // ob1.getmark = 100
    // ob2.getmark = 300
    // → out Put = 400
}
```

Function is use object

```
void sum (School ob1, School ob2)
{
    cout << ob1.mark + ob2.mark << endl;
}
```


void sum (
فانکشن
class
class
to object
فانکشن
two object
Friend function
دالة محتواها بدا الـ class ولكنه تكون خارج
class student
{
private:
int mark;
public:
student (int m)
mark = m;
void print;
Friend void show (student obj);
};
void student :: print()
{ cout<<"mark is: " << mark <<endl;
Friend function
Friend function
محتوى الـ
class بدا الـ
{void show (student obj)
{ cout<<"mark is :" << obj. mark << endl;
9

Public
عام

أي حد يقدر يشوفه سواء من int main()
أو با ال int main()

Private:-
خاص

أي حد جوا int main() ميقدرش يشوفه
لكن با ال int main() يشوفه عادي

Protected:-
محمي

int main() يستخدم مع الوراثة [ممكن نشوفه من وممكن لا]

ملحوظة

Public	Private
تستخدم أكثر مع المتغيرات لكن مع الدوال بكتب Public دا	Private يستخدم مع الكبسلة
	Protected يستخدم مع الوراثة

Inheritance
الوراثة

```
class first
{ public
    int x = 50;
}
class second : public first
{
    public
    void print()
    cout<<"x is :-" << x <<endl;
}
```

// output → x is 50

تم التوريث كلاس second وراث كل خصائص كلاس first

أنواع الوراثة ← 5 أنواع

1 Single inheritance

كلاس يورث من كلاس واحد

```
Class A
Class B : Public A
```

2 multiple inheritance

كلاس يورث من كل الكلاسات

```
Class A
Class B
Class C
Class D : Public A, Public B, Public C
```

3 Hierarchical Inheritance

كل يورث من كلاس واحد

```
Class A
Class B : Public A
Class C : Public A
Class D : Public A
```

4 multiLevel Inheritance

كل كلاس يورث من الكلاس اللي قبله

```
Class A
Class B : Public A
Class C : Public B
Class D : Public C
```

operator overloading

معروف أن

+ للجمع
- للطرح
% للباقي القسمة
/ للقسمة
* للضرب

لو فرضنا كدا مثلا test
obj1+obj2 هينفع اجمع
عشان ⊕ معروفة
انها للارقام وليست
object

```
int x=2 ;
x++;
cout<<x<<endl ;
```

output → X=3

لازم اعمل ++ obj1; أو obj1++;
أو obj1 -- أو --obj1;
وهكذا
معالجة + لل operator overloading

```
Class Test(){
    Private:-
      int x=0;
    Public:-
     Test()
     { cout<<"enter x:";
       cin>>x;
     }
     void operator ++(int)
     {
        x++;
     }
```

```
int main(){
Test obl ;   obl++;
```

Post Fix
بعد

```
x++;
x--;
```

Pre Fix
مثل

```
++x;
--x;
```

operator overloading

وظيفة

```
void operator ++(int)
{
  x++;
}
```

```
void operator --(int)
{
  x--;
}
```

```
void operator ++()
{
  ++x;
}
```

```
void operator --()
{
  --x;
}
```

object ← العمليات الحسابية

```
class Test
{
  private:
    int x;
  public
    Test(){ cout << "enter x"
            cin >> x; }
    void print()
    {cout << "x is:" << x << endl;
    }
```

```
void operator += (int y)
{
  x+=y;
}
```

+= operator

```
void operator -= (int y)
{
    x-=y;
}
```

-= operator

```
void operator /= (int y)
{
    x/=y;
}
```

/= operator

```
void operator *= (int y)
{
    x*=y;
}
```

*= operator

```
void operator %= (int y)
{
    x%=y;
}
```

%= operator

```
int main()
Test obj;          // cin>>x  ---> x=5
obj+=10;           // x is 15
obj-=5             // x is 10
obj *= 2           // x is 20
obj %=2            // x is 0
   obj.print();    // x is 0
```

Different method to write constructor

```
class Count          private value=0;   x=5;
{       public
        Count(): value(20), x(10)
        {}
};
```