

Clas OPP

```
using namespace std;

/*We create class for creating objects*/
class myCar {
public:
    string _Color;
};

int main() {

    /*We create our objects*/
    myCar OTOMOBILE_1;
    myCar OTOMOBILE_2;
    myCar OTOMOBILE_3;

    /*We define their color feature*/
    OTOMOBILE_1._Color = "Blue";
    OTOMOBILE_2._Color = "Red";
    OTOMOBILE_3._Color = "Green";

    cout << OTOMOBILE_1._Color << " " << OTOMOBILE_2._Color << " " << OTOMOBILE_3._Color << endl;
}
```

Bu yukardaki en basit temelde class dır, Normal hali VS da vardır.

Private olunca ilgili .h .cpp dosyası dışındakiler erişemez, örneğin main.cpp de include etmiş olsan bile .h ı, yine de erişemez.

```
using namespace std;

class Otomobile {
public: // Access from all .h .cpp files in the myClass_Otomobile project

    /*Class members*/
    string renk;
    int beygirGucu;
    string Modeli;

    /*Class methods*/
    /*Tek bir metoddla görüntülememizi sağlayacak*/
    void ruhsatBilgileriniGoster();
    /*Main kodda objenin elini kolunu oluştururken 6 satır harcamamak için bu şekilde bir yapıcı metod (constructor) tanımladık */
    Otomobile(string _renk, int _beygirGucu, string _Modeli);

private: // Access just from .h and cpp of myClass_Otomobile , not other ones

    string cantModeli; //Otomobile constructor u erişebilir.
                        //ruhsatBilgileriniGoster metodu da erişebilir
                        //ama main.c den erişim olmaz, Yani .h .cpp dosyası(bu class için) erişebilir diğer .h .cpp ler erişemez
};

#include "myClass_Otomobile.h" // < > şeklinde sistemin kendi kütüphanesiymiş gibi dahil etmeyeceksin

/*Bu metod ile hangi araç olursa olsun çağrıldığında kaydedilenler tıklar tıklar çıkmalı*/

void Otomobile::ruhsatBilgileriniGoster() {

    cout << "Model : " << Otomobile::Modeli << ", ";
    cout << "Beygir Gucu : " << Otomobile::beygirGucu << ", ";
    cout << "Renk : " << Otomobile::renk << ", ";
    cout << endl;

    /* access private class object |
    Otomobile::cantModeli = "Kormetal";
    cout << "Cant Modeli : " << Otomobile::cantModeli;
    */
}

//Otomobil constructors
Otomobile::Otomobile(string _renk, int _beygirGucu, string _Modeli) {

    renk      = _renk;
    beygirGucu = _beygirGucu;
    _Modeli    = _Modeli;
}
```

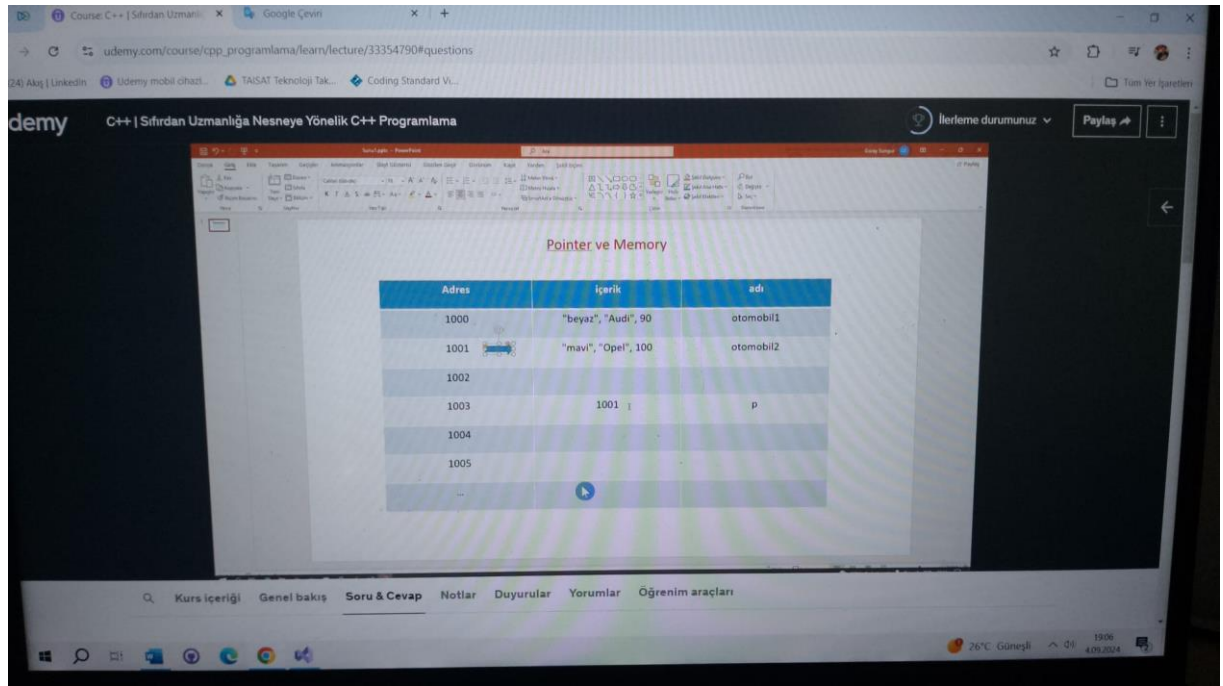
Class ı geliştirirken private tanımlarım ki ekipteki bir başkasının kendi class ını yazarken yahut algoritmayı inşa ederken kullanacağı bir şeyi benim değişkenlerimle karıştırmamasın diye kendikilerimi private yapar sonradan public e çevirebilirim

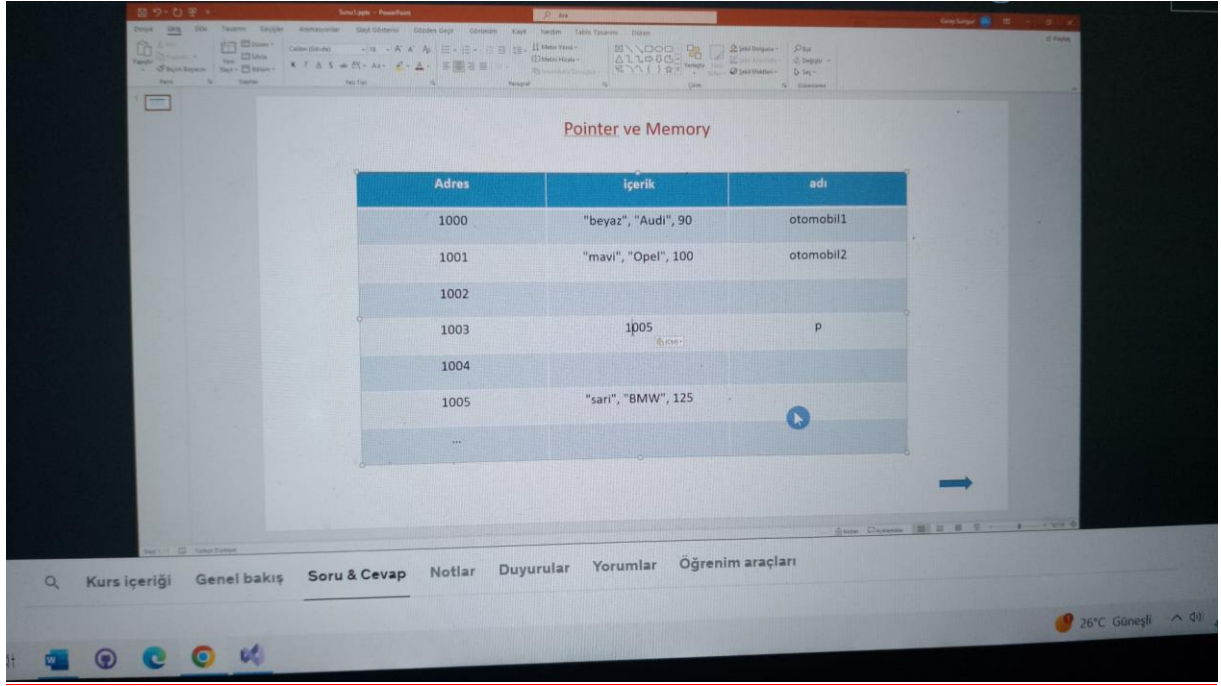
Ama ille de ulaşmak isterse , bende private ı bozmak istemezsen nasıl olacak?

➔ Metodlar ayarlarım illede kullanmak isterse public metodlarımı kullanıp içerisine yazar

Pointerlar ile Classlar

```
/******How to use pointer with class?******/  
  
// First of all , identify pointer with class name  
Otomobile *myPointer;  
  
// Secondly, we write Porsche's first variable address into the myPointer  
myPointer = &Porsche;  
  
// Finally if we ask car information or accessing other thing, use ' -> ' symbol like as shown below  
myPointer->  
  
public: void Otomobile::degistirOtoBeygirGucu(int_beygirGucu)  
Dosya: myClass_Otomobile.h
```





Eğer pointer ile nesne oluşturmak istersek yukarıdaki gibi ramde yer kaplar, Aşağıda gösterdiğim gibi programı da yazılmış olur. Tıpkı memory block için anlık yer tahsis etmek gibi. Ama işlevi bitince serbest bırakmak lazım ki diğer kullanıcılar tarafından yer tahsisine açık olsun

```
/******How to create object with pointer?******/

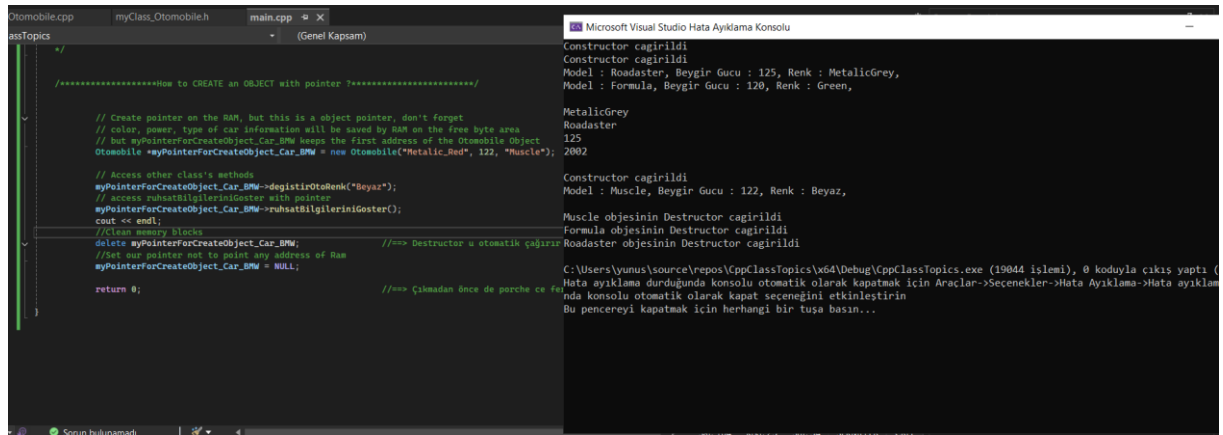
// Create pointer on the RAM, but this is a object pointer, don't forget
// color, power, type of car information will be saved by RAM on the free byte area
// but myPointerForCreateObject_Car_BMW keeps the first address of the Otomobile Object
Otomobile *myPointerForCreateObject_Car_BMW = new Otomobile("Metalic_Red", 122, "Muscle");

//Otomobile *myPointerForCreateObject_Car_BMW;
//myPointerForCreateObject_Car_BMW = new Otomobile("Metalic_Red", 122, "Muscle");

// Access other class's methods
myPointerForCreateObject_Car_BMW->degistirOtoRenk("Beyaz");

// access ruhsatBilgileriniGoster with pointer
myPointerForCreateObject_Car_BMW->ruhsatBilgileriniGoster();
```

POINTER VE DESTRUCTOR



The screenshot shows a Visual Studio IDE with two windows. The left window, titled 'main.cpp', contains C++ code for a class 'Otomobile' and its methods. The code includes comments in Turkish explaining the use of pointers and destructors. The right window, titled 'Microsoft Visual Studio Hata Ayıklama Konsolu', shows the output of the program. It displays the constructor being called for 'MetalicGrey', 'Roadaster', and 'Muscle' models, followed by the destructor being called for each model. The output also shows the memory address of the 'Otomobile' object and the destructor being called for it.

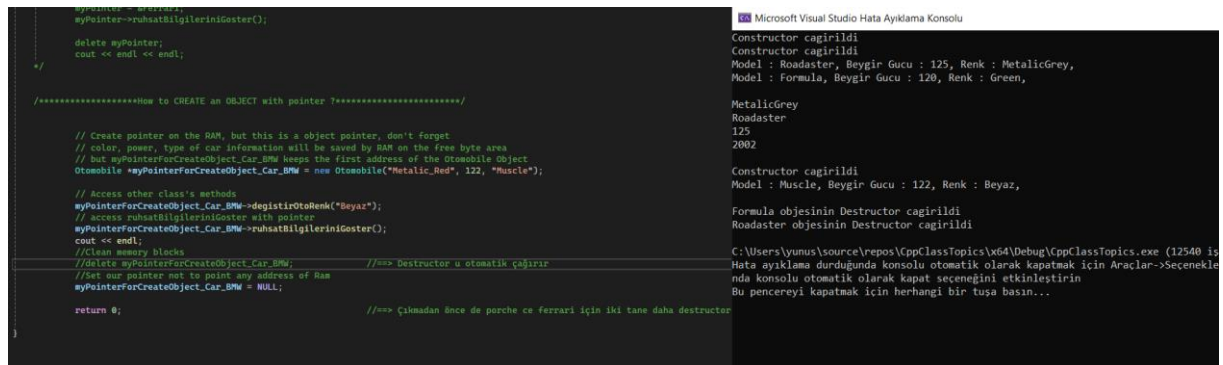
Constructor'lar ile objeleri belirli bir kalıp(şema)(class) dan üretebiliriz, destructor ile de var olanları Ram üzerinden kaldırabiliriz.

İlk iki constructor bizim porche ile ferrariyi üretiyor. Sonraki constructor ise pointer ile obje oluşturuyoruz onun için yazılıyor.

İşimiz bitince destructor porche ve Ferrari için otomatik çağırılıyor ama BMW li olanı(Yani pointerlı ürettiğimizi) bizim delete etmemiz lazım ki Ram üstündeki memory block kısmı da kalmış olsun.

Porche ile Ferrari de biz delete etmediğimizden return 0 kısmında otomatik olarak destructe ediliyor

Pointerlar ile oluşturulan objeleri mutlaka işiniz bitince delete edin yoksa sen programı kapatsan bile ilgili yer destructor olmadığı için sen PC yi kapayana kadar yahut reStart atana kadar o obje bütün memberları ile beraber RAM üstünde kalacak. Bu da çok büyük projelerde PC de çökmeler meydana getirebilir.



The screenshot shows a Visual Studio IDE with two windows. The left window, titled 'main.cpp', contains C++ code for a class 'Otomobile' and its methods. The code includes comments in Turkish explaining the use of pointers and destructors. The right window, titled 'Microsoft Visual Studio Hata Ayıklama Konsolu', shows the output of the program. It displays the constructor being called for 'MetalicGrey', 'Roadaster', and 'Muscle' models, followed by the destructor being called for each model. The output also shows the memory address of the 'Otomobile' object and the destructor being called for it.

Görüldüğü gibi delete etmeyince muscle modeli destructor edilmedi ve pc'nin raminde kalmış oldu.

Peki ben class'ımın içinde public yahut private alanda int, float, double tanımlar gibi pointer da tanımlayabilir miyim ?

EYET

Peki nasıl kullancam niye kullanayım, ama çok kullanıldığını söylelim

Ki vardı :

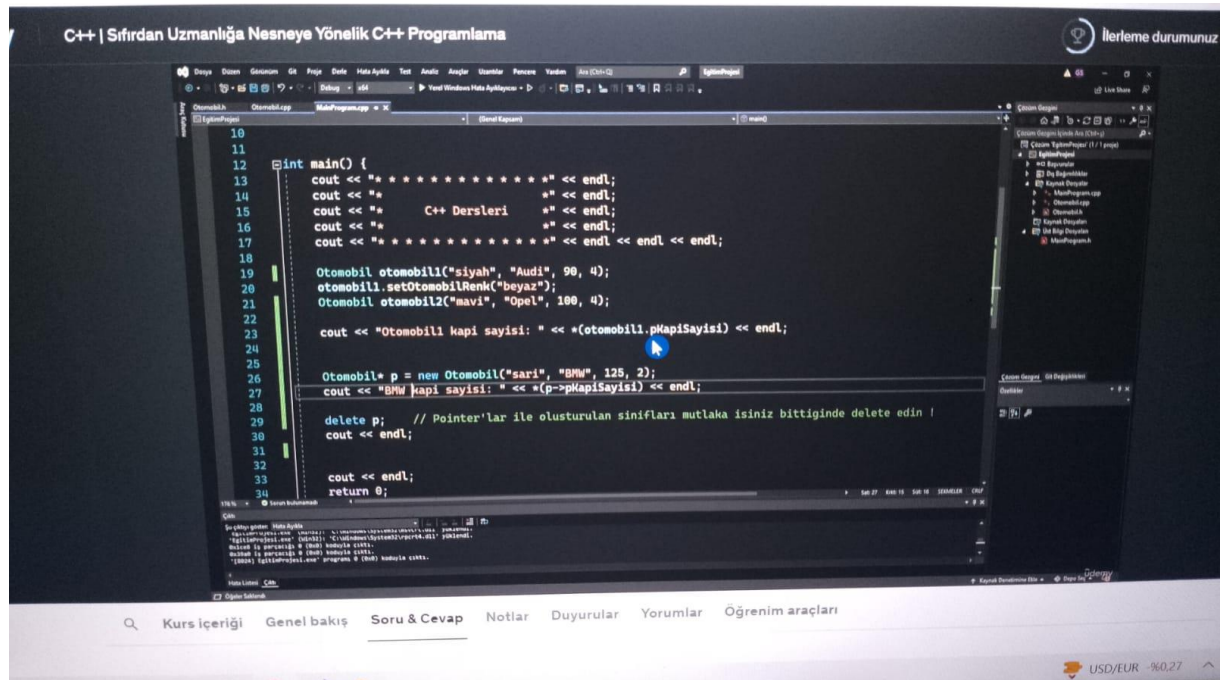
```
#define SAT_PAYLOAD_SUBSYS_DRIVERS_ACTUATOR_SERVO_COLORFILTER_TEST_STATUS
#define SAT_PAYLOAD_SUBSYS_DRIVERS_ACTUATOR_SERVO_SEPARATION_TEST_STATUS

/***** INCLUDES *****/
#include "main.h"

typedef struct
{
    TIM_HandleTypeDef *htim_X;
    uint32_t tim_channel_in;
}Actuator Servo HandleTypeDef;
```

Yukarıdaki resimde objenin pointerından bahsediliyor.s

Peki c++ de niçin kullanılıyor



The screenshot shows a C++ program in Visual Studio. The code is as follows:

```
10
11
12 int main() {
13     cout << "*****" << endl;
14     cout << " " << endl;
15     cout << "      C++ Dersleri      " << endl;
16     cout << " " << endl;
17     cout << "*****" << endl << endl << endl;
18
19     Otomobil otomobil1("siyah", "Audi", 90, 4);
20     otomobil1.setOtomobilRenk("beyaz");
21     Otomobil otomobil2("mavi", "Opel", 100, 4);
22
23     cout << "Otomobil kapi sayisi: " << *(otomobil1.pKapiSayisi) << endl;
24
25
26     Otomobil* p = new Otomobil("sari", "BMW", 125, 2);
27     cout << "BMW kapi sayisi: " << *(p->pKapiSayisi) << endl;
28
29     delete p; // Pointer'lar ile olusturulan siniflari mutlaka isiniz bittiginde delete edin !
30     cout << endl;
31
32     cout << endl;
33     return 0;
34 }
```

The program demonstrates pointer usage in C++. It defines a class `Otomobil` with a pointer member `pKapiSayisi`. In the `main` function, it creates two `Otomobil` objects, `otomobil1` and `otomobil2`, and prints their `pKapiSayisi` values using the `*` operator. It then creates a pointer `p` to a new `Otomobil` object, prints its `pKapiSayisi` value using the `->` operator, and finally deletes the pointer.

Pointer ile tanımlandıysa erişimi '->' ifadesi le yapıcaksın tıpkı 27.satırdaki gibi

Yok normal değişken gibi tanımlandıysa ' ' ile erişiceksin tıpkı 23.satırdaki gibi

```

/*****How to CREATE an OBJECT with pointer ?*****/

// Create pointer on the RAM, but this is a object pointer, don't forget
// color, power, type of car information will be saved by RAM on the free byte area
// but myPointerForObject_Car_BMW keeps the first address of the Otomobile Object
Otomobile *myPointerForObject_Car_BMW = new Otomobile("Metalic_Red", 122, "Muscle",4);

// Access other class's methods
| //myPointerForObject_Car_BMW->degistirOtoRenk("Beyaz");

// access ruhsatBilgileriniGoster with pointer
myPointerForObject_Car_BMW->ruhsatBilgileriniGoster();
cout << "Kapi sayisi BMW : " << *(myPointerForObject_Car_BMW->pkapiSayisi);

cout << endl;
//Clean memory blocks
delete myPointerForObject_Car_BMW; //==> Destructor u otomatik çağırır
//Set our pointer not to point any address of Ram
myPointerForObject_Car_BMW = NULL;

/*****How to CREATE an OBJECT with pointer ?*****/

```