



SAKARYA ÜNİVERSİTESİ
Bilgisayar ve Bilişim Bilimleri Fakültesi
Bilgisayar Mühendisliği Bölümü

CPP ile Nesne Yönelimli Programlama (NYP)



Prof. Dr. Cemil ÖZ
Prof. Dr. Celal ÇEKEN
Doç. Dr. Cüneyt BAYILMIŞ

Konular

- ✓ Nesne Yönelimli Programlama (NYP)
- ✓ Nesne (Object)
- ✓ UML ve Nesne Örnekleri
- ✓ Kapsülleme (Encapsulation)
- ✓ Mesaj Gönderme
- ✓ Sınıf (Class)
- ✓ Erişim Belirleyicileri
- ✓ Get Ve Set Yöntemleri
- ✓ Sınıf ve Nesne Örnekleri
- ✓ Polymorphism - Çok Şekillilik
- ✓ Inheritance-Kalıtım
- ✓ Yapıcılar (Constructor)
- ✓ Yıkıcı Fonksiyon (Destructor)
- ✓ Çok Dosyalı Programlar
- ✓ Fonksiyona Parametre Olarak Nesne Gönderimi
- ✓ Static
- ✓ Const
- ✓ Nesnelerin Bellek Kullanımı
- ✓ Dizi elemanı Olarak Nesne Kullanımı
- ✓ Sorular
- ✓ Kaynaklar

Nesne Yönelimli Programlama (NYP)

- ✓ Yapısal teknikte programcı doğrudan probleme odaklanır ve problemin çözümüne ilişkin yöntemleri geliştirir.
- ✓ Nesne yönelimli programlama tekniğinde ise temel bileşen **nesnedir** ve programlar nesnelerin birlikte çalışmasından meydana gelir.
- ✓ Nesne hem veriyi hem de bu veriyi işleyen fonksiyonları içerir. Programcılar dikkatlerini nesneleri oluşturan sınıfları geliştirmeye yoğunlaştırır.
- ✓ Yapısal teknikte bir fonksiyon herhangi bir görevi yerine getirmek için veriye ihtiyaç duyarsa, gerekli veri parametre olarak gönderilir. NYP de ise yerine getirilmesi gereken görev nesne tarafından icra edilir ve fonksiyonlar verilere parametre gönderimi yapılmaksızın erişebilirler.

Sistem büyüdükçe ilişkiler ve bağımlılık daha da karmaşıklaşır.

Hata bulma zorlaşır

Program içerisinde değiştirme, ekleme, çıkarma vs. yapmak zorlaşır ve beklenmeyen etkilere neden olabilir...

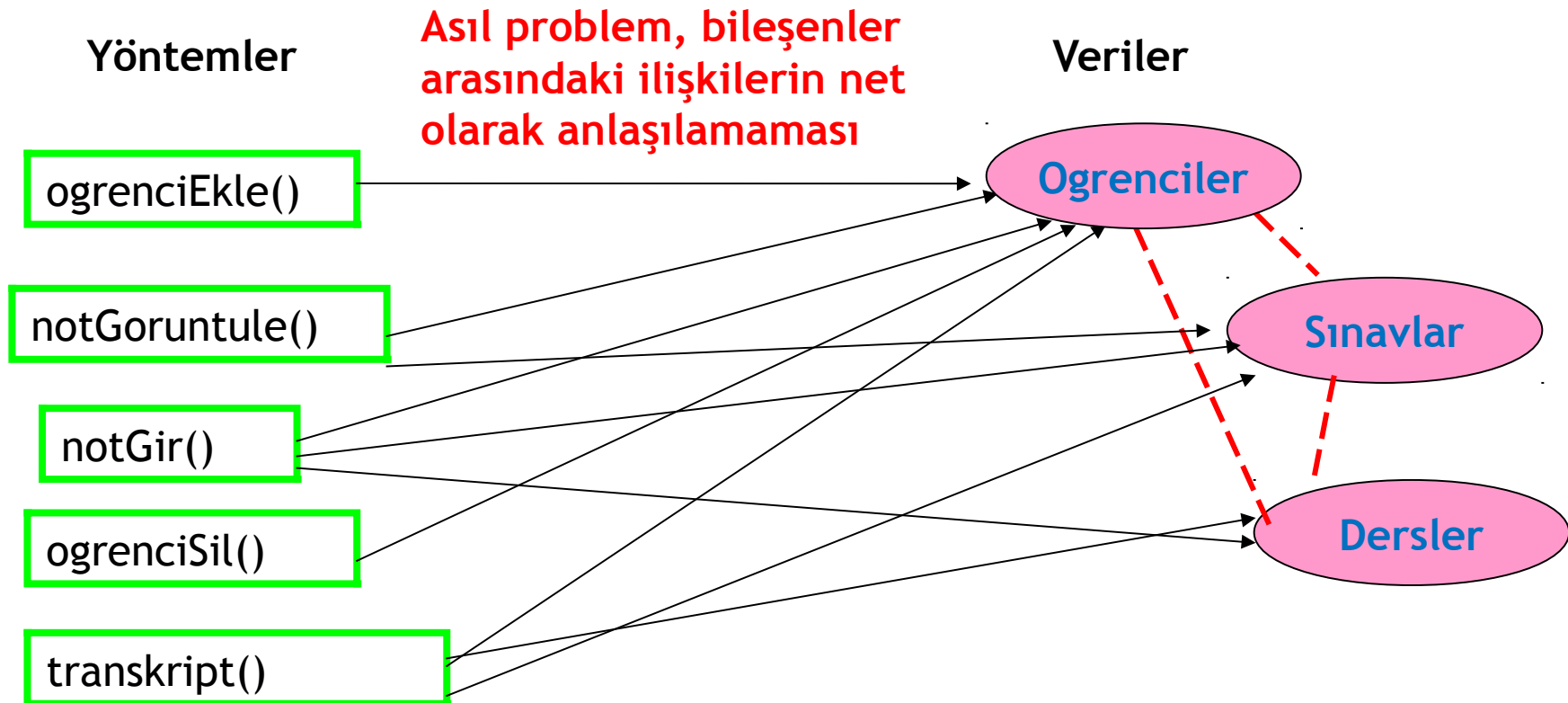
Örneğin;

Oğrenciler tablosunda öğrencinin doğum tarihi iki haneli

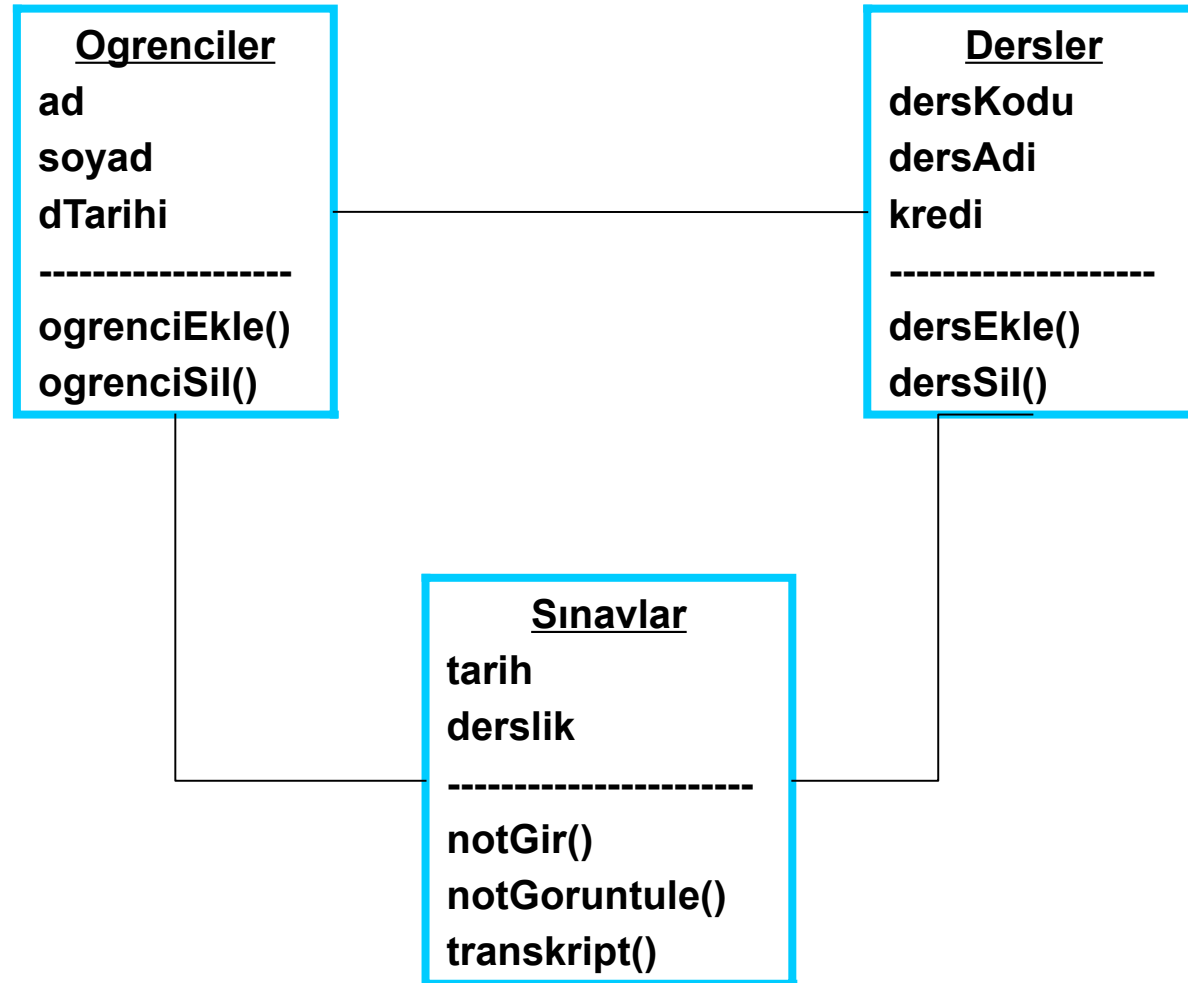
Bu alanı 4 haneli yapmak istiyoruz...

Oğrenciler tablosu Sınavlar ve dersler tablolarıyla ilişkili olduğundan beklenmeyen bir sorun çıkabilir...

Tüm yöntemler Öğrenciler tablosunu bir şekilde kullanıyor. Büyük sistemlerde de hata olma ihtimali var.. öğrenciEkle() yöntemi kesinlikle sorun çıkaracaktır...



Nesne Yönelimli Programlama (NYP)



Nesne Yönelimli Programlamanın Üstünlükleri/Özellikleri

Encapsulation, data abstraction, inheritance, and polymorphism.

- ✓ Problemler daha kolay tanımlanıp çözülebilir. Gerçek dünya düşünülerek geliştirilmiştir. Geliştirme süreci daha kolay olur.
- ✓ Bilgi Gizleme (Information Hiding, Data abstraction)
 - ✓ Nesne içerisindeki işlemler (nasıl) diğer nesnelerden soyutlanır-sadece ne yapacağını bilirler. Nesne içerisindeki değişiklik diğer nesneleri etkilemez-regression fault. Dolayısıyla bakım aşaması daha kolay olur.
 - ✓ Gereksiz ayrıntılarla uğraşılmaz, probleme odaklanılır (arabanın gitmesi için gaza basmak yeterli)- Daha hızlı geliştirme süreci.
- ✓ Modüler Programlama (Modular Programming)
 - ✓ Nesneler birbirinden tamamen bağımsız (veri + fonksiyon) (encapsulation, responsibility driven design)
 - ✓ Büyük ve karmaşık bir problem küçük parçalara ayrılarak daha kolay çözülebilir. Geliştirme ve bakım daha kolay olur.
 - ✓ Programların geliştirilmesi daha hızlı, geliştirilen bir nesne diğer programlarda da rahatlıkla kullanılabilir. (. Hata bulma-bakım daha kolay) Bisikleti başkasına verdiğimiz zaman da çalışır. string nesneleri her programda kullanılır.
 - ✓ Geliştirme sürecinde grup çalışmalarına olanak sağlanır.

Nesne Yönelimli Programlamanın Üstünlükleri/Özellikleri

- ✓ Kodların Tekrar Kullanımı (Code Reuse)
 - ✓ Nesneler başka programlara kolaylıkla aktarılabilir. Bakım ve geliştirme zamanı/maliyeti düşer
 - ✓ Kalıtım, Çok şekillilik
- ✓ Hata Bulma - Bakım/Onarım (Maintainence)
 - ✓ Bileşenler arasındaki ilişkiler açık olduğundan (veri+fonksiyon aynı yapı içerisinde) güncelleme, hata bulma ve bakım daha kolay
- ✓ Design Patterns
- ✓ Günümüzdeki en iyi yaklaşım. Gelecekte ?

Nesne (Object)

- ✓ Gerçek hayatta çevremizde gördüğümüz her şey nesnedir.
 - ✓ İnsan, masa, sıra, bisiklet, araba, köpek v.s.
- ✓ Nesneler iki özelliğe sahiptirler. Bunlar:
 - ❶ **Durum**
 - ❷ **Davranış**
- ✓ İnsan için **durumlar**; adı, yaşı, boyu v.s. iken **davranışlar**; öğrenmek, anlamak, uyumak, konuşmak, koşmak v.s.
- ✓ Bisiklet için **durumlar**; rengi, o anki vitesi, hızı, tekerlek sayısı, vites sayısı v.s. iken **davranışlar**; fren yapması, hızlanması, yavaşlaması, vites değiştirmesi v.s.
- ✓ Yazılım nesneleri de durum ve davranışlara sahiptir. Nesnelerin durumları **özellik** olarak da adlandırılır ve **değişkenler** ile ifade edilir.
- ✓ Davranışlar ise **fonksiyon** adı verilen ve nesne içerisinde yer alan alt programlar (yöntem) kullanılarak gerçekleştirilir.

Nesne

Nesne Tasarlanırken şu sorular sorulmalı:

- ✓ **Özellik** belirlenirken

Nesnenin özellikleri ne olmalı (neye sahiptir)

- ✓ **Davranış** belirlenirken

Nesne ne yapabilir? (Ne yapması istenir ?)

UML (Unified Modelling Language)

- ✓ Bisiklet nesnesinin (sınıfının) UML ile gösterimi



Nesne Örnekleri

Öğrenciler
numara ad soyad
ogrenci_ekle() ogrenci_sil() ogrenci_ara()

Dersler
kodu adi kredi
ekle() cikar()

Zaman
saat dakika saniye
basla() ayarla() goster()

Televizyon
kanalsayısı kullanılan band ses parlaklık
ac() kapat() kanal_ara() ses_ac() ses_kapa()

Hesap
no bakiye
goster() para_cek() para_yatir()

Nesneleri Nasıl Oluşturabiliriz ?

- ✓ Nesneleri Sınıf (Class) ile oluştururuz.

Sınıf (Class)

- ✓ Sınıf bir nesnenin planı ya da tipi gibi düşünülebilir.
- ✓ Sınıf nesnenin davranışını ve özelliklerini belirler.
- ✓ Her nesnenin bir sınıfı vardır ve bir nesne oluşturulduğunda sınıfın bir örneği (instance) oluşturulmuş olur.
- ✓ Bir sınıfa ait her nesne bellekte yer kaplar ve bu yerin adresi tanımlayıcı ya da referans olarak adlandırılan değişkende saklanır.
- ✓ Aynı sınıfı kullanan birden fazla nesne oluşturulabilir ve bu durumda her nesneye farklı referans değişkenleri kullanılarak erişilir.

Sınıf İsimlendirme Kuralları

- ✓ Sınıf isimleri harf, rakam ve '_' ifadelerinden oluşur.
- ✓ İlk harfi rakam olamaz. Türkçe karakterler kullanılamaz.
- ✓ Ayrılmış kelimeler (**reserved words**) kullanılamaz. If, else, for, final v.s.
- ✓ Büyük harf-küçük harf duyarlılığı vardır (**case-sensitive**).
- ✓ Sınıf isminin içerdiği veri ile ilgili olması büyük kolaylıklar sağlar.

HesapMakinesi

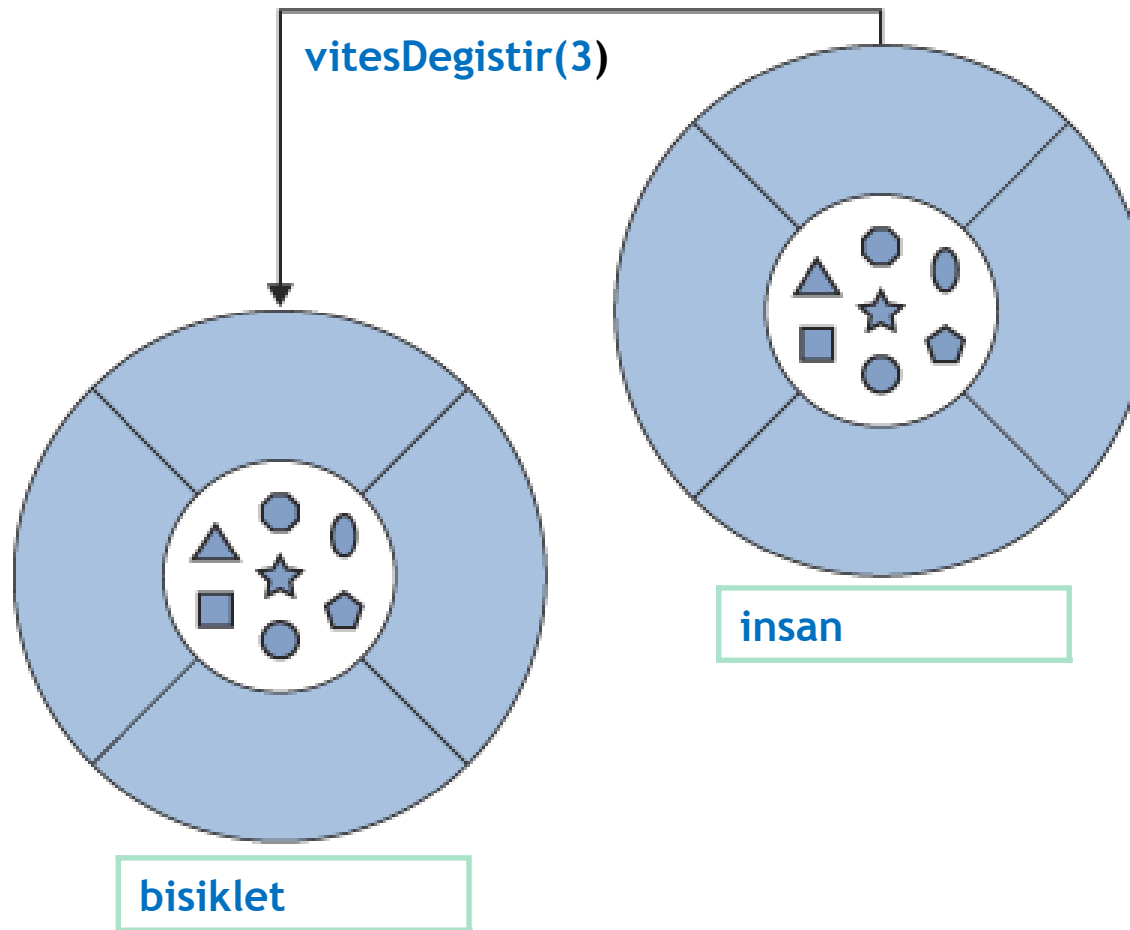
- ✓ Sınıf isimlerini oluşturan kelimelerin baş harfi büyük olmalı.

Kapsülleme (Encapsulation)

- ✓ Bir sınıfa ait değişkenleri (verileri) ve yöntemleri bir araya toplayarak birleştirmeye **kapsülleme** denir.
- ✓ Kapsülleme özelliği programcıya iki temel avantaj sağlar. Bunlar:
- ✓ **Modülerlik**: Bir nesnenin kodu diğer nesnelerden tamamen bağımsız olarak geliştirilir. Bununla birlikte geliştirilen bir nesne diğer programlarda da rahatlıkla kullanılabilir. Bisikleti başkasına verdiğimiz zaman da çalışır. string nesneleri her programda kullanılır.
- ✓ **Bilgi Gizleme**: Bir nesne, diğer nesnelerin haberleşebilmek amacıyla kullanabileceği public ara yüze sahiptir. Bununla birlikte diğer nesneler tarafından erişilmesine izin verilmeyen özel değişken ve yöntemleri de içerebilir. Kullanıcı tarafından bisikletin vites mekanizmasının bilinmesine gerek yoktur. Bilgi gizlemenin en büyük avantajlarından biri de nesneye ait ayrıntıların nesne tarafından gizlenerek kullanıcının çözülmesi gereken probleme konsantre olmasının sağlanabilmesidir.

Mesaj Gönderme

İnsan nesnesinin bisiklet nesnesine ait `vitesDegistir(3)` yöntemini çağırmasına mesaj gönderme denir.



Sınıf (Class)

Sınıf Tanımı

Sınıfın örneği olan nesneler (sınıf tipindeki nesneler)

Sınıf Adı: Otomobil

Veri:

yakıt miktarı _____

hız _____

plaka _____

Yöntemler:

hızlan:

nasıl: gaz pedalına bas.

yavaşla:

nasıl: fren pedalına bas.

Birinci Örnek:

Nesen adı: araba1

yakıt miktarı : 10 lt

hız : 55 km / s

plaka : "41 AD 44"

İkinci Örnek :

Nesen adı : araba2

yakıt miktarı : 14 lt

hız: 0 km / s

plaka : "33 NC 3240"

Üçüncü Örnek :

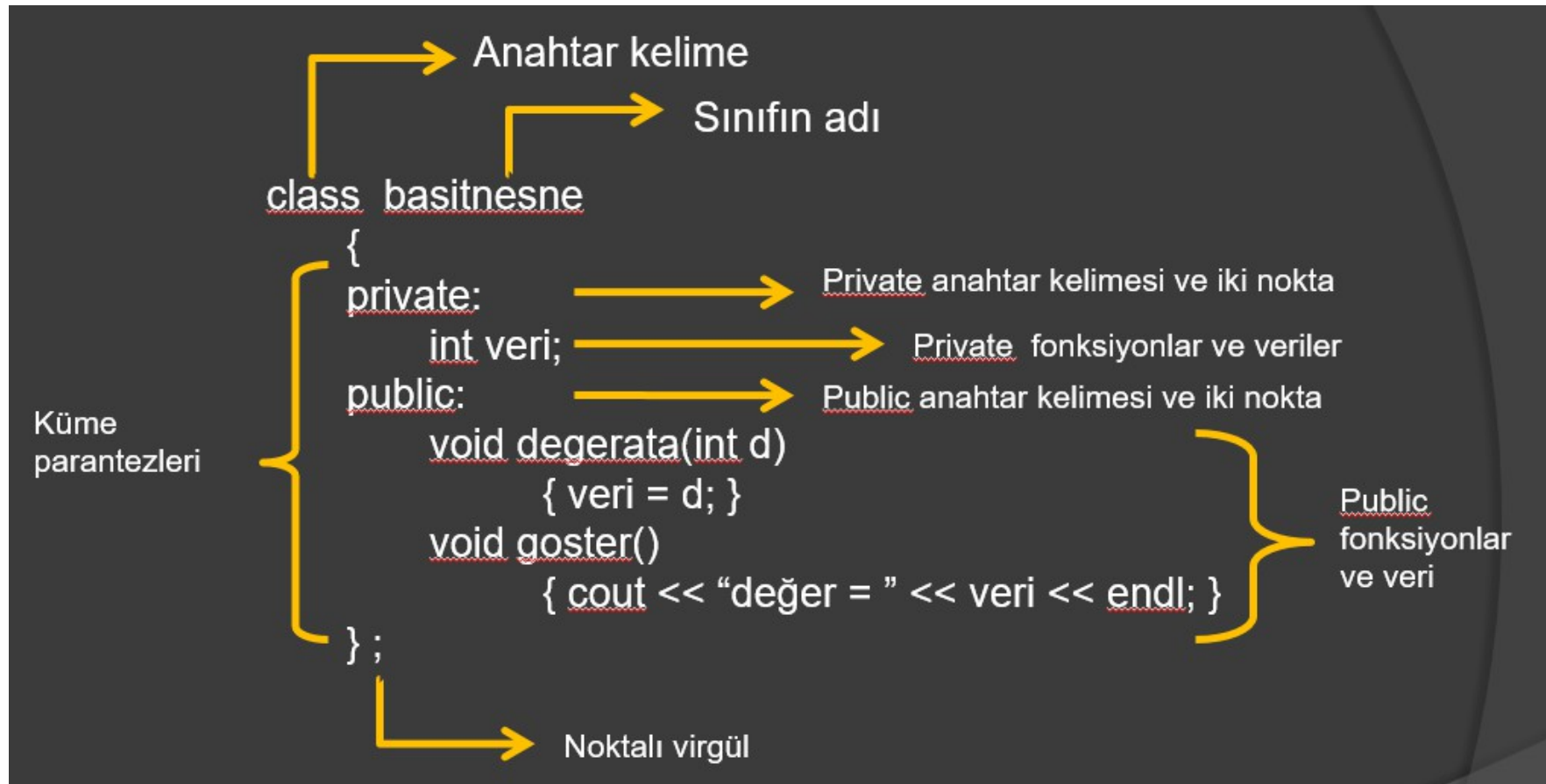
Nesen adı : araba3

yakıt miktarı: 20 lt

hız : 75 km / s

plaka: "35 LF 44"

Sınıf Örneği



Sınıf Örneği

```
class Sınıf_adi {  
    private:  
        //özel veriler ve fonksiyonlar  
    public:  
        // genel veriler ve fonksiyonlar  
    protected:  
        // korunmuş veriler ve fonksiyonlar  
} nesne_listesi;
```

```
class Arac {  
    int max_hiz;  
    public:  
        int model;  
        char marka[10];  
        void fren_yap(int ivme);  
        void hizlan(int ivme);  
        int hiz_Oku();  
};
```

Sınıf içerisindeki prototip olarak tanımlanmış fonksiyonların gövdesi (yaptıkları işlev) program içerisinde şu şekilde tanımlanır

```
Void Arac :: fren_yap (int ivme)  
{  
    .....  
}
```

Erişim Belirleyicileri

Erişim düzeyi değişken ya da fonksiyona diğer nesne ya da sınıflardan nasıl erişilebileceğini belirler.

- ✓ **public** : Tüm diğer nesnelerden erişilebilir.
- ✓ **private** : Sadece nesne içerisindeki üyeler tarafından erişilebilir. Private kısmındaki elemanlara public kısmında bulunan üye fonksiyonlar ile erişilebilir.
- ✓ **protected** : Aynı dizinde (alt dizinler de dahil) bulunan sınıflar tarafından erişilebilir.
Bu kısımdaki elemanlara kalıtım yoluyla türetilen alt nesneler tarafından erişilebilir.

Sınıf ve Nesne Örnekleri

```
class Personel
```

```
{  
    private:  
        string ad;  
        string soyad;  
        string adres ;  
        int sigortaNo;  
        double maas; //yillik maas
```

Üye Değişkenler

Personel p1;

p1 nesnesi yığın bellekte oluşturulur...

Nesneler kavramsal olarak veri üyeleri ve üye fonksiyonlar içerirken, aslında C++ nesneleri sadece veri içerir.

Derleyici, sınıf üye fonksiyonlarının sadece bir kopyasını oluşturur ve o kopyayı tüm sınıf nesneleri arasında paylaşır.

```
public:  
    void bilgiGoster()  
    {  
        cout<<"Ad: "<<ad<<endl<<"Soyad: "<<soyad<<endl  
        <<"Adres: "<<adres<<endl<<"S.No: "<<sigortaNo  
        <<"yillik Maas: "<<maas<<endl;  
    }  
  
    void bilgiGir()  
    {  
        cout<<"ad giriniz:";  
        cin>>ad;  
        cout<<"soyad giriniz:";  
        cin>>soyad;  
        cout<<"adres giriniz:";  
        cin>>adres;  
        cout<<"s. guvenlik no giriniz:";  
        cin>>sigortaNo;  
        cout<<"maas (yillik) giriniz:";  
        cin>>maas;  
    }
```

Üye Fonksiyonlar

```
}; //Personel Sinifi Sonu
```

1Personel.cpp

Set ve Get Yöntemleri

- ✓ Sınıfın veri üyelerine **set** ve **get** fonksiyonları ile erişerek veri üyelerinden kaynaklanacak değişiklik etkilerini yerelleştirmeye çalışınız.
- ✓ Temiz ve bakımı kolay programlar yazınız. Değişiklik istisnadan ziyade kuraldır. Kodunuzun değiştirileceğini öngörmelisiniz.

```
class Personel
{
    private:
        string ad;
        string soyad;
        string adres ;
        int sigortaNo;
        double maas;

    public:

        void setAdres(string a)
        {
            adres=a;
        }

        string getAdres()
        {
            return adres;
        }
}
```

Sınıf ve Nesne Örnekleri

Soru 1 :Personel adresi için getAdres yontemi tanımlayarak veri doğrulamasını aşağıdaki örneğe benzer şekilde yapınız....

```
void setAdres(string adres1)
{
    if(adres1.length()>50)
        adres = adres1.substr(0,30);
    else
        adres=adres1;
}
```

Veri Gizleme - Data Hiding

- ✓ public
- ✓ private
- ✓ protected

```
class X {  
    public:  
        int x,y;  
        void fonkA();  
    private:  
        int w,y;  
        int fonkB();  
};
```

```
main(){  
    X x1,x2;  
    x1.x=3; //geçerli  
    x1.w=4; //geçersiz  
    x2.fonkA(); //geçerli  
    x2.fonkB(); //geçersiz
```

Sınıf ve Nesne Örnekleri

```
class Karmasik
{
    private:

        double gercel;
        double sanal;

    public:
        void bilgiGoster()
        {
            cout<<"Sayinin degeri: "<<gercel<<"+"<<sanal<<"i"<<endl;
        }

        void bilgiGir()
        {
            cout<<"Sayinin gercel kismini giriniz:";
            cin>>gercel;
            cout<<"Sayinin sanal kismini giriniz:";
            cin>>sanal;
        }

    private:
        float radyandanDereceye(float a)
        {
            return a*180/PI;
        }

    public:
        void kutupsalaCevir()
        {
            cout<<"\nsayinin kutupsal karsiligi: ";
            cout<<sqrt(pow(gercel,2)+pow(sanal,2))<<" ";
            cout<<radyandanDereceye(atan(sanal/gercel));
        }
};
```

Sınıf ve Nesne Örnekleri

```
#include <iostream>
```

```
#include <string>
```

```
using namespace std;
```

```
class Ogrenci {
```

```
private:
```

```
    string ad;
```

```
    string soyad;
```

```
    int vize;
```

```
    int final;
```

```
    double ort;
```

```
public:
```

```
    void gir () {
```

```
        cout<<"Ogrenci Adini Giriniz: ";
```

```
        cin>>ad;
```

```
        cout<<"Ogrenci Soyadini Giriniz: ";
```

```
        cin>>soyad;
```

```
        cout<<"Vize Notu: ";
```

```
        cin>>vize;
```

```
        cout<<"Final Notu: ";
```

```
        cin>>final;
```

```
    }
```

```
void goster() {
```

```
    cout<<ad<<"\t"<<soyad<<"\t"<<ortalama(
    )<<"\t";
```

```
    basari(ortalama());
```

```
    cout<<endl;
```

```
}
```

```
double ortalama () {
```

```
    return ort = vize*0.5 + final*0.5;
```

```
}
```

```
void basari (double ort) {
```

```
    if (ort>=50)
```

```
        cout<<"Gecti";
```

```
    else
```

```
        cout<<"Kaldi";
```

```
}
```

```
}; // Öğrenci sınıfı sonu
```

```
main()
```

```
{
```

```
    Ogrenci ogr1,ogr2;
```

```
    ogr1.gir();
```

```
    cout<<"-----"<<endl;
```

```
    ogr2.gir();
```

```
    cout<<"-----"<<endl;
```

```
    cout<<"*** Öğrenci Bilgileri ***"<<endl;
```

```
    ogr1.goster();
```

```
    ogr2.goster();
```

```
    system ("pause");
```

```
}
```


Sınıf ve Nesne Örnekleri

```
#include <iostream>

using namespace std;

class mesafe //sınıf tanımla
{
private:
    int metre;
    float cm;
public:
    void degerata (int met, float sant)
        {metre=met; cm=sant; }
    void degeral ()
        {cout <<“\n metre gir “;cin>>metre;
        cout<<“ cm gir “;cin>>cm;
        }
    void goster()
        {cout<< metre << “ metre ” << cm << “ cm dir”;}
};

int main()
{
    mesafe m1,m2; // iki obje tanımla
    m1.degerata(5, 20); // m1 objesine değer ata
    m2.degeral(); // m2 objesine degeral

    cout<<“\n m1 değeri “; m1.goster();
    // m1 objesinin değerlerini göster
    cout<<“\n m2 değeri “; m2.goster();
    // m2 objesinin değerlerini göster

    return 0;
}
```

Yapıcılar (Constructor)

Bir yapıcı aşağıdaki özelliklere sahiptir:

- Sınıf ile aynı ada sahiptirler.

- Sınıf içerisinde hiç olmayabildiği gibi bir ya da çok sayıda da olabilir.

- Geri dönüş (return) değeri yoktur.

- Sadece bir defa, nesne oluşturulurken çalıştırılırlar.

Bir sınıf içerisinde hiç yapıcı tanımlanmamışsa, derleyici varsayılan bir yapıcıyı otomatik olarak tanımlar. Bu yapıcı nesne için bellekte yer açar. Varsayılan yapıcı temel tipteki veri üyelerine hiç bir başlangıç değeri atamadan sadece nesneyi oluşturur. Diğer sınıfların nesneleri olan veri üyeler için varsayılan yapıcı, her bir veri üyesine ait varsayılan yapıcıyı veri üyelerinin uygun bir şekilde başlatılmalarını garanti etmek üzere kapalı olarak çağırır. string veri üyesinin boş bir string şeklinde başlatılmasının nedeni budur. String sınıfı içerisindeki varsayılan yapıcı bunu sağlar.

...

Yapıcılar genellikle üye değişkenlere ilk değer ataması için kullanılır. Üye değişkenlere ilk değer ataması yapılmadan kullanılırsa mantıksa hatalar oluşabilir.

```
class Karmasik
{
    private:

        double gercel;
        double sanal;

    public:

        Karmasik()
        {
            gercel=0;
            sanal=0;
            cout<<"yapici calisti...\n";
        }

        Karmasik(double a, double b)
        {
            gercel=a;
            sanal=b;
            cout<<"iki parametrelili yapici calisti...\n";
        }
}
```

Yapıcılar (Constructor)

Bir sınıf varsayılan yapıcıya iki şekilde sahip olabilir.

1. Derleyici yapıcısı olmayan bir sınıf için kapalı olarak oluşturur. Böyle bir yapıcı sınıfın veri üyelerini başlatmamakla birlikte diğer sınıfların bir nesnesi olan her veri üyelerinin varsayılan yapıcılarını çağırır. Başlatılmamış bir değişken tipik olarak "çöp" değer içerir.

2. Argüman almayan bir yapıcıyı siz tanımlayabilirsiniz. Böyle bir yapıcı diğer sınıfların nesnesi olan veri üyelerinin varsayılan yapıcılarını çağırarak ve sizin tarafınızdan belirtilen ek başlatma işlemlerini gerçekleştirecektir. Eğer argümanı olay yapıcı tanımlarsanız, C++ kapalı olarak o sınıf için varsayılan bir yapıcı oluşturmayacaktır.

Yapıcılar Örnek

```
#include <iostream>
#include <string>
using namespace std;
class Ogrenci {
private:
    string ad;
    string soyad;
    int ort;

public:
    //Yapıcı Fonksiyon nesne oluşturulurken ve sadece 1 defa
    //calistirilir
    Ogrenci ()
    {
        ad="Şener";
        soyad="Şen";
        ort=55;
        cout<<"Parametresiz Yapici Calisti ... \n";
    }
    Ogrenci (string a, string b, int n)
    {
        ad=a;
        soyad=b;
        ort=n;
        cout<<"3 Parametrelili Yapici Calisti ... \n";
    }
}
```

```
void gir () {
    cout<<"Ogrenci Adini Giriniz: ";
    cin>>ad;
    cout<<"Ogrenci Soyadini Giriniz: ";
    cin>>soyad;
    cout<<"Ortalama: ";
    cin>>ort;
}

void goster() {
    cout<<ad<<"\t"<<soyad<<"\t"<<ort<<"\t";
    basari(ort);
    cout<<endl;
}

void basari (int a) {
    if (a>=50)
        cout<<"Gecti";
    else
        cout<<"Kaldi";
}
}; // Öğrenci sınıfı sonu

main()
{
    Ogrenci ogr1;
    Ogrenci ogr2 ("Cem","Demir",40);
    ogr1.goster();
    ogr2.goster();
    /******/
    ogr1.gir();
    ogr1.goster();
    /******/
    ogr2.gir();
    ogr2.goster();
    system ("pause");
}
```

Yıkıcı Fonksiyon (Destructor)

Bir yıkıcı aşağıdaki özelliklere sahiptir:

Sınıf ile aynı ada sahiptirler. Solunda ~ isareti vardır

Sınıf içerisinde hiç olmayabilir.

Geri dönüş (return) değeri yoktur. Parametre gönderilmez.

Sadece bir defa, nesne yok edilirken otomatik olarak çalıştırılır.

Sadece 1 yok edici fonksiyon tanımlanabilir.

```
class Karmasik
{
    private:

        double gercel;
        double sanal;

        int nesneNo;

    public:

        // Yapıcı Fonksiyon nesne oluşturulurken
        Karmasik():gercel(0.0), sanal(0.0)
        { //gercel=0;
          //sanal=0;
          cout<<"Yapici Calisti...\n";
        }

        ~Karmasik()
        {
          cout<<"Yikici calisti...\n"<<endl;
        }
}
```

Bir sınıf içerisinde hiç yıkıcı fonksiyon tanımlanmamışsa, derleyici varsayılan bir yıkıcıyı otomatik olarak tanımlar. Composition ve kalıtım için özel görevleri vardır.

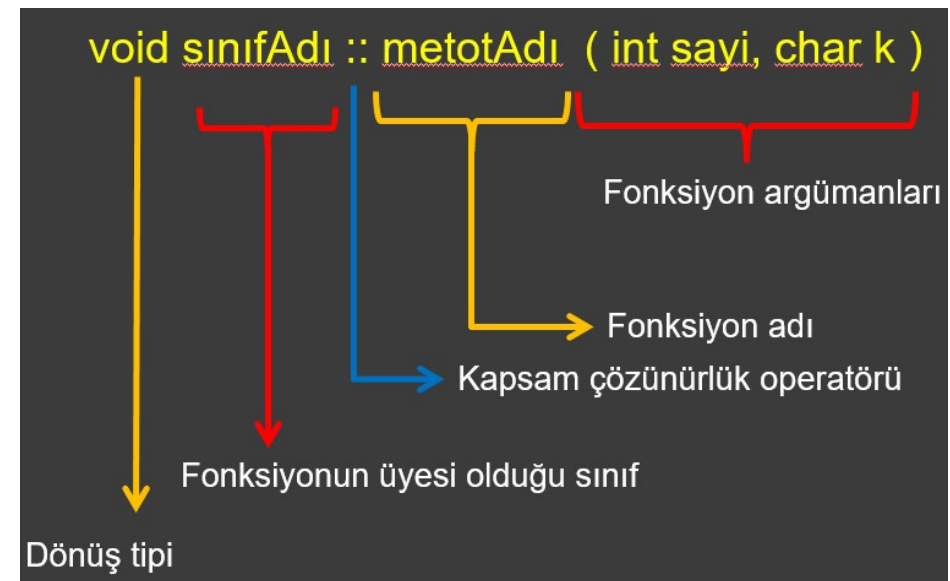
Dinamik ayrılan bellek bölgesini boşaltır (string değişken yok edilirken içerisindeki yıkıcı heap bölgesinde ayrılan yeri boşaltır. delete []buffer;), dosya veya başka sistem kaynaklarının kapatılması/bırakılması için kodlar yazılmasını sağlar

The destructor itself does not actually release the object's memory it performs **termination housekeeping** before the system reclaims the object's memory, so the memory may be reused to hold new objects

Fonksiyon Tanımlama

- ✓ Metotlar iki türlü tanımlanabilir
 - class içersinde (**inline**)
 - class dışında

```
class X {  
    public:  
        X(){cout<<"constructor";endl;}    //inline tanımlama  
        X(int a);  
        int xkare(void);  
    private:  
        int x;  
};  
X::X(int a){                                //class dışında tanımlama  
    x=a*a;  
}  
int X::xkare(void){                          //class dışında tanımlama  
    return x*x;  
}
```



```
#include <iostream>
```

```
using namespace std;
```

```
class mesafe //sınıf tanımla
```

```
{
```

```
private:
```

```
    int metre;
```

```
    float cm;
```

```
public:
```

```
    mesafe():metre(0),cm(0.0)
```

```
    { }
```

```
    mesafe(int met, float sant): metre(met), cm(sant)
```

```
    { }
```

```
    void degeral ()
```

```
    {
```

```
        cout <<"\n metre gir ";cin>>metre;
```

```
        cout<<" cm gir ";cin>>cm;
```

```
    }
```

```
    void goster()
```

```
        {cout<< metre << " metre " << cm << " cm dir";}
```

```
    void toplauzun(mesafe,mesafe); //deklerasyon
```

```
};
```

Sınıf ve Nesne Örnekleri

```
void mesafe::toplauzun(mesafe m2,mesafe m3)
```

```
{
```

```
    cm=m2.cm +m3.cm;
```

```
    metre=0 ;
```

```
    if(cm>=100.0)
```

```
    {
```

```
        cm-=100.0;
```

```
        metre++;
```

```
    }
```

```
    metre+=m2.metre+m3.metre;
```

```
}
```

```
int main(int argc, char *argv[])
```

```
{
```

```
    mesafe mesafe1,mesafe3;
```

```
    mesafe mesafe2(5,4.3);
```

```
    mesafe1.degeral();
```

```
    mesafe3.toplauzun(mesafe1,mesafe2);
```

```
    cout<<"\n mesafe1= ";mesafe1.goster();
```

```
    cout<<"\n mesafe2= "; mesafe2.goster();
```

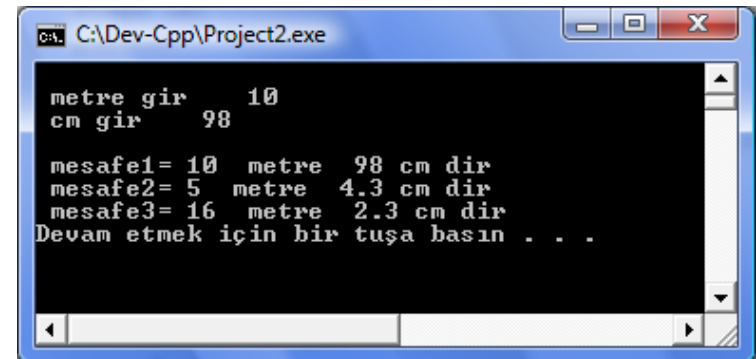
```
    cout<<"\n mesafe3= "; mesafe3.goster();
```

```
    cout<<endl;
```

```
    system("PAUSE");
```

```
    return EXIT_SUCCESS;
```

```
}
```



```
C:\Dev-Cpp\Project2.exe

metre gir 10
cm gir 98

mesafe1= 10 metre 98 cm dir
mesafe2= 5 metre 4.3 cm dir
mesafe3= 16 metre 2.3 cm dir
Devam etmek için bir tuşa basın . . .
```

Çok Dosyalı Programlar

Kodların tekrar kullanımı
Interface-implementation

Sınıf tanımı artık başlık dosyası (main içermeyen) içerisindedir ve Personel sınıfını yeniden kullanmak isteyen her programa bu başlığı dahil edebiliriz.

```
//Baslik dosyasinin defalarca eklenmesini onlemek için ilk
//iki satir ve son satirdaki onislemci komutları kullanılır.
#ifndef PERSONEL_H
#define PERSONEL_H

#include <iostream>
#include <iomanip>
#include <string>
using namespace std;
// Programin arayuzu (interface)
class Personel
{
    private:
        string ad;
        string soyad;
        string adres ;
        int sigortaNo;
        double maas;
    public:
        void bilgiGoster()
        void bilgiGir();
};
// Programin gerceklemesi(implementation)
void Personel::bilgiGir()
{
    cout<<"ad giriniz:";
    cin>>ad;
    cout<<"soyad giriniz:";
    cin>>soyad;
    cout<<"adres giriniz:";
    cin>>adres;
    cout<<"s. guvenlik no giriniz:";
    cin>>sigortaNo;
    cout<<"maas (yillik) giriniz:";
    cin>>maas;
}
void Personel::bilgiGoster()
{
    cout<<"Ad: "<<ad<<endl<<"Soyad: "<<soyad<<endl
        <<"Adres: "<<adres<<endl<<"S.No: "<<sigortaNo
        <<"yillik Maas: "<<maas<<endl;
}
#endif
```


Çok Dosyalı Programlar

Kodların tekrar kullanımı
Interface-implementation

Birden fazla include edilen dosyalar içerisindeki class lar aynı isme sahip olacağından hatalara neden olur.
Çözüm:

```
#ifndef ( "if not defined")  
#define  
  
#endif
```

Araştırma Sorusu: Birçok durumda sınıfın gerçekleştirme(implementation) kısmının sınıfı kullananlar tarafından görünmesi istenmez. Bu nedenle gerçekleştirme kısmının nesne kodu verilir... kullanıcı sadece arayüz (interface) kısmını görür ve kullanır. Aşağıda verilen Personel örneği için bu uygulamayı gerçekleştiriniz...

```
//Baslik dosyasinin defalarca eklenmesini onlemek icin ilk  
//iki satir ve son satirdaki onislemci komutlari kullanilir.  
#ifndef PERSONEL_H  
#define PERSONEL_H  
  
#include <iostream>  
#include <iomanip>  
#include <string>  
using namespace std;  
// Programin arayuzu (interface)  
class Personel  
{  
    private:  
        string ad;  
        string soyad;  
        string adres ;  
        int sigortaNo;  
        double maas;  
    public:  
        void bilgiGoster()  
        void bilgiGir();  
};  
// Programin gerceklemesi(implementation)  
void Personel::bilgiGir()  
{  
    cout<<"ad giriniz:";  
    cin>>ad;  
    cout<<"soyad giriniz:";  
    cin>>soyad;  
    cout<<"adres giriniz:";  
    cin>>adres;  
    cout<<"s. guvenlik no giriniz:";  
    cin>>sigortaNo;  
    cout<<"maas (yillik) giriniz:";  
    cin>>maas;  
}  
void Personel::bilgiGoster()  
{  
    cout<<"Ad: "<<ad<<endl<<"Soyad: "<<soyad<<endl  
        <<"Adres: "<<adres<<endl<<"S.No: "<<sigortaNo  
        <<"yillik Maas: "<<maas<<endl;  
}  
#endif
```

Fonksiyona Parametre Olarak Nesne Gönderme

Soru : Klavyeden ESC tuşuna basılıncaya kadar girilen karmaşık sayıların toplamını bulan programı yazınız. Karmaşık sayı işlemleri (bilgi girişi, toplama v.s.) için Karmasik sınıfı tipinde nesne kullanınız.

6KarmasikToplam.cpp

```
#include <iostream>
#include <iomanip>
#include <cmath>

using namespace std;

const double PI = 3.14159265;

#include "Karmasik.h"

int main()
{
    Karmasik sayi1(3,2);
    Karmasik sayi2(6,8);
    Karmasik sayi3;

    sayi1.karmasikTopla(sayi2);

    sayi3=sayi1.karmasikTopla(sayi1,sayi2);
    sayi3.bilgiGoster();

    system("pause");

    sayi1.bilgiGoster();
    sayi2.bilgiGoster();

    system("pause");

    sayi1.bilgiGir();
    sayi2.bilgiGir();
    sayi1.bilgiGoster();
    sayi1.kutupsalaCevir();

    system("pause");
    return 0;
}
```

Dizi Elemanı Olarak Nesne Kullanımı

```
#include "Karmasik.h"

int main()
{
    Karmasik sayilar[5];

    sayilar[0].bilgiGoster();

    for(int i=0;i<5;i++)
        sayilar[i].bilgiGir();

    for(int i=0;i<5;i++)
        sayilar[i].bilgiGoster();

    system("pause");
    return 0;
}
```

Soru: Karmasik sınıfı içerisine karmaşık sayının kutupsal koordinatlarıyla ilgili r ve teta değerlerini döndürmek üzere rDondur ve tetaDondur adlı iki üye fonksiyon ekleyiniz.

En fazla 50 eleman girileceğini düşünerek klavyeden ESC tuşuna basılıncaya kadar girilen karmaşık sayıları dizide saklayan ve girilen karmaşık sayıların r değerlerinin aritmetik ortalamasını bulan programı geliştiriniz...

Static

Static veriye tüm nesneler erişebilir. Bir sınıftan oluşturulan tüm nesnelerin ortak bilgi paylaşımı yapmaları gerektiğinde üye değişken static olarak tanım kullanılır.

Static fonksiyonlar nesne oluşturulmadan da kullanılabilirler.

Soru : Donusturucu adında bir sınıf tanımlayınız. Bu sınıf içerisinde dışarıdan aldığı parametreyi dönüştüren; inch2cm, cm2inch, radyan2derece, derece2radyan, F2C, C2F, dolar2TL, euro2TL ... üye fonksiyonlar tanımlayınız. Tanımladığınız sınıftaki üye fonksiyonları (nesne oluşturmadan) kullanan ana programı yazınız.

8Static.cpp

Sakarya Üniversitesi

BSM 103 Programlamaya Giriş

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

class Araba
{
    private:
        int renk;
        int model;
        double yakitMiktari;
    public:
        static int sayi;

    public:
        Araba()
        { cout<<"+sayi<<endl; }
        static int getSayi()
        //int getSayi()
        { return sayi; }
        ~Araba()
        { cout<<"sayi--<<endl; }
};

int Araba::sayi = 0; // statik üyeler burada başlatılabilir

int main()
{
    cout<<"ilk:"<<Araba::getSayi();
    {
        Araba f1, f2, f3;

        cout << "Aktif araba sayisi " << f1.getSayi() << endl;
        cout << "Aktif araba sayisi " << f2.getSayi() << endl;
        cout << "Aktif araba sayisi " << f3.getSayi() << endl;
    }

    system("pause");
    return 0;
}
```

Const

Yetkiler ne kadar ayrıntılı belirlenebilirse yazılımlar o oranda kaliteli olur. Kodların sadece ihtiyaç duyulduğu kadarına erişilebilmesi çok önemlidir. Bu sayede hata önleme ve hatalardan kaçınma kolaylaşır.

Bir nesnenin değiştirilememesi gerektiğinde kullanılır.

- ✓ **Const** olarak tanımlanan nesneler içerisindeki üye fonksiyonların kullanılabilmesi için onların da **const** olması ve üye değişkenleri değiştirmemesi gerekir.
- ✓ **Const** fonksiyon üye değişkenleri değiştiremez.

Yapıcı ve yıkıcı fonksiyonlar **const** olamazlar.

- ✓ **Const** bir üye fonksiyon yine **const** olan bir üye fonksiyonu kullanabilir.
- ✓ **Const** bir üye değişkeni başlatmak için yapıcı içerisinde atama işareti kullanılamaz. Bunun yerine:

Olcu(int ft, float in) : metre(ft), cmetre(in) kullanılmalı

- ✓ **Const** olarak oluşturulmayan bir nesne **const** olan fonksiyonları kullanabilir...

Önemli Not: Sınıf içerisindeki üye değişkenleri değiştirmeyen fonksiyonların **const** olarak tanımlanması olası hataları önleme açısından önemlidir...

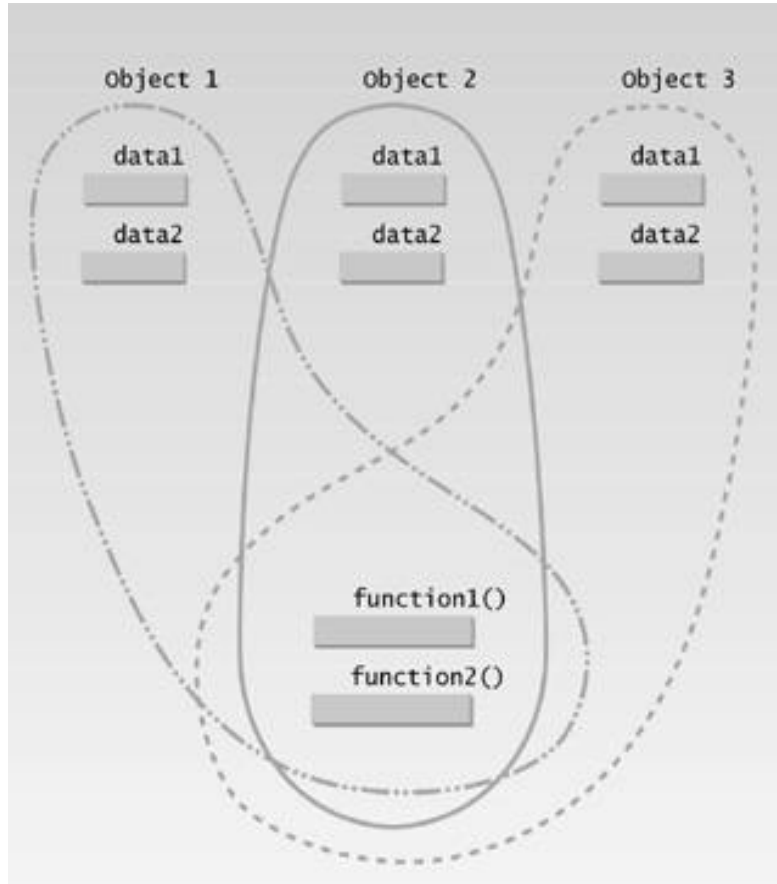
Öğrenci Örneği: Öğrenciler modülü için **const** öğrenci nesnesi oluşturulabilir
(Öğrencilerin değişiklik yapması söz konusu değil)
İdari personel modülünde nesne normal olarak tanımlanabilir (Değişiklik yapılabilirmeli...)...

Const

```
#include <iostream>
#include <cstring>


using namespace std;
////////////////////////////////////
class Ogrenci //English Olcu class
{
    private:
        string ad;
        string soyad;
    public: //2-arg constructor
        Ogrenci(string a, string s) : ad(a), soyad(s)
        { //ad=a;
        }
        void bilgiGir() //const olmayan nesne kullanabilir
        {
            getline(cin,ad);
            getline(cin,soyad);
        }
        void bilgiListele() const //const olan nesne bunu kullanır
        {
            cout<<ad<<"\t"<<soyad;
        }
        void bilgiListele() //const olmayan nesne bunu kullanır
        {
            cout<<ad<<"\tconst olmayan\t"<<soyad;
        }
};
////////////////////////////////////
int main()
{
    const Ogrenci ol("Ayse","Yilmaz");
    //ol.bilgiGir();//Hata, sadece const fonksiyonlar kullanılabilir
    ol.bilgiListele(); //const olan fonksiyon...
    system("pause");
    return 0;
}
```

Nesnelerin Bellek Kullanımı

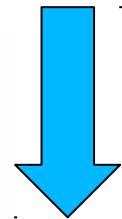
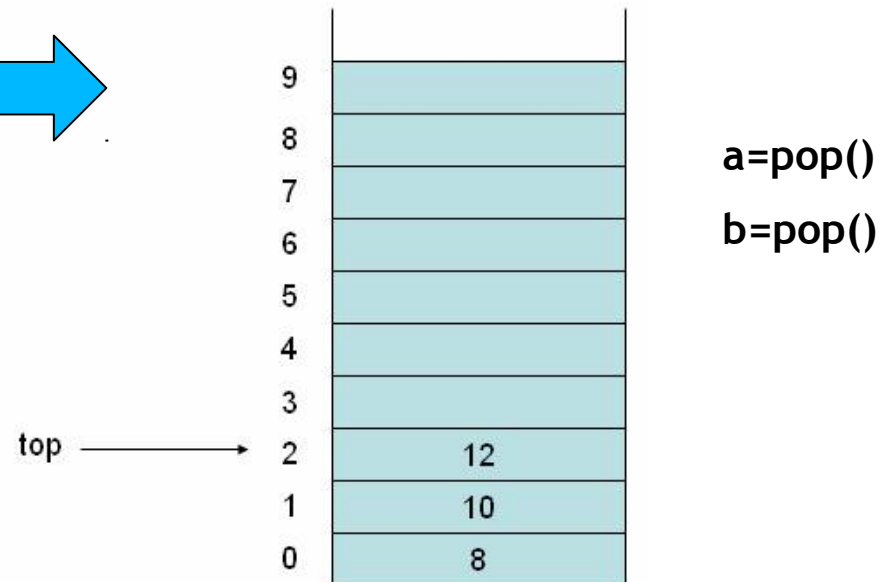
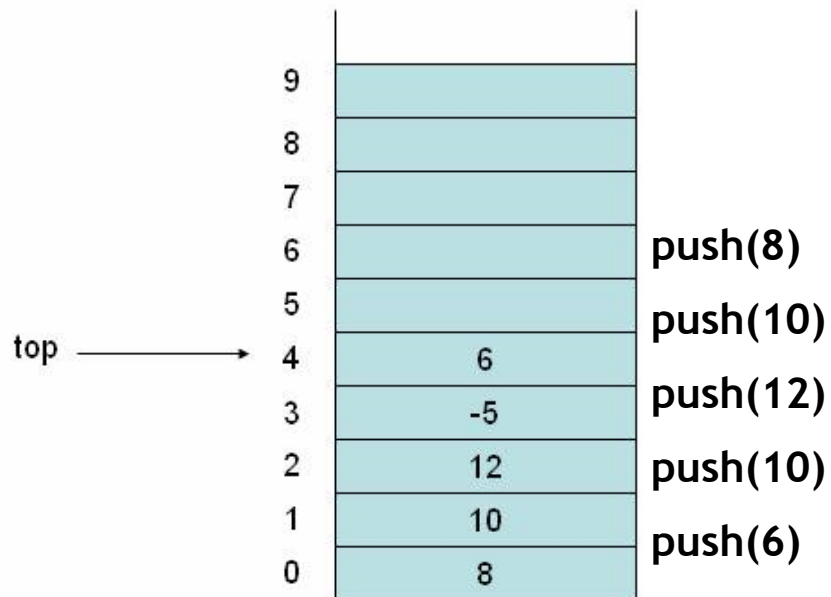


Araştırma Sorusu: Belleğin hangi bölümleri vardır (code, stack, heap, static v.s.)? Nesne yönelimli programlamada bellek organizasyonu nasıldır? (Örneğin; oluşturulan bir nesne belleğin hangi bölümündedir? Yerel değişken, global değişken statik üye v.s. Hangi bölümlerdedir)

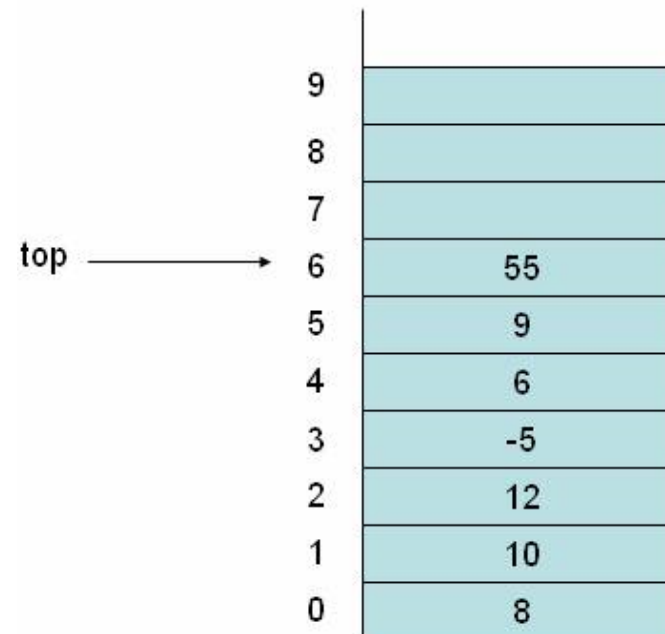
Örnek

-  iostream
-  std
-   Sifrele
 - mesaj : string
 - sifreliMesaj : string
 - anahtar : int
 - Sifrele()
 - sifrele() : void
 - sifreCoz() : void
 - ^c getAnahtar() : int
 - setAnahtar(int) : void
 - ^c getMesaj() : const string&
 - setMesaj(const string&) : void
 - ^c getSifreliMesaj() : const string&
 - setSifreliMesaj(const string&) : void
- main() : int

Stack



push(-5)
push(6)
Push(9)
push(55)



Stack

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

class Stack
{
    private:
        enum { MAX = 10 };
        int st[MAX];
        int top;
    public:
        Stack()
        { top = -1; }
        void push(int var)
        { st[++top] = var; }
        int pop()
        { return st[top--]; }
};

int main()
{
    Stack s1;

    s1.push(11);
    s1.push(22);
    cout << "1: " << s1.pop() << endl;    //22
    cout << "2: " << s1.pop() << endl;    //11
    s1.push(33);
    s1.push(44);
    s1.push(55);
    s1.push(66);
    cout << "3: " << s1.pop() << endl;    //66
    cout << "4: " << s1.pop() << endl;    //55
    cout << "5: " << s1.pop() << endl;    //44
    cout << "6: " << s1.pop() << endl;    //33

    system("pause");
    return 0;
}
```

Sorular

1. **AsalSayi** sınıfını tanımlayınız... Bu sınıf içerisinde :
 1. **asalSayiGir** fonksiyonu sayi üye değişkenine klavyeden değer girilmesini sağlamalı
 2. **asalmi** fonksiyonu sayi üye değişkeninin asal olup olmadığı bilgisini geri döndürmeli .
 3. **enYakinAsal** fonksiyonu parametre olarak aldığı sayiya en yakın büyük asal sayiyi geri döndürmeli

1. **Karakter katarı** işlemlerinde kullanılacak **KarakterKatari** adlı sınıfı tanımlayınız. Bu sınıf içerisinde karakter katarını tutacak katar uye değişkenini tanımlayınız. Üye fonksiyonlar :
 1. **Get ve set** fonksiyonları
 2. **Klavyeden girilen karakter katarını uye değikene alacak olan katarOku**
 3. **Katar üye değişkenindeki bilgiyi tamamen büyüğe çevirecek buyugeCevir**
 4. **kucugeCevir**
 5. **Büyükse küçük, küçükse büyük (Ali->aLİ gibi) yapan buyukKucukCevir**
 6. **Kelimelerin ilk harfini büyüğe çeviren (boşluk karakteri kontrol edilmeli...) ilkHarfBuyuk**
 7. **Girilen karakter katarının uzunluğunu döndüren fonksiyonu yazınız... (hazır fonksiyon kullanmayınız...)**
 8. **terstenYazdir**

1. **NesneOrnekleri.pdf** dosyasında bulunan sınıf örnekleriyle ilgili programlar yazınız...

Kaynaklar

- ✓ Robert Lafore, Object Oriented Programming in C++, Macmillan Computer Publishing
- ✓ Deitel, C++ How To Program, Prentice Hall
- ✓ Prof. Dr. Cemil ÖZ, Programlamaya Giriş Ders Notları
- ✓ Prof. Dr. Celal ÇEKEN, Programlamaya Giriş Ders Notları