

# C++ ile Nesne Yönelimli Programlamaya (OOP) Giriş

Prof.Dr. Kasım Kurt

- 1 Neden Nesne Yönelimli Programlama?
- 2 Sınıf ve Nesne Kavramı
- 3 Basit Bir Sınıf Örneği
- 4 Kapsülleme ve Erişim Belirleyiciler
- 5 Kurucular (Constructor)
- 6 Genel Özeti

# Neden OOP?

- Büyük projeler fonksiyonel (prosedürel) yaklaşımla zor yönetilir.
- Veriler ve bu veriler üzerinde çalışan fonksiyonlar birbirinden kopuktur.
- OOP ile:
  - **Veri + fonksiyon** aynı çatı (sınıf) altında toplanır.
  - Kod daha **modüler, bakımı kolay** ve **yeniden kullanılabilir** olur.
- Temel kavramlar:
  - Sınıf (class), nesne (object)
  - Kapsülleme (encapsulation)
  - Kalıtım (inheritance)
  - Çok biçimlilik (polymorphism)
- Bu derste **sınıf, nesne ve kapsülleme** üzerinde duracağız.

# Sınıf (Class) Nedir?

- Sınıf: Kendi veri tipimizi tanımladığımız bir **şablon**.
- İçinde:
  - **Veri elemanları** (özellikler, data members)
  - **Üye fonksiyonlar** (davranışlar, member functions)
- Örnek düşünce:
  - Ögrenci sınıfı:
    - Özellikler: ad, numara, not ortalaması
    - Davranış: bilgileri yazdır, ortalama güncelle, vb.

# Nesne (Object) Nedir?

- Sınıf: Plan, tasarım, şablon.
- Nesne: Bu planın **gerçek hayattaki örneği**.
- C++'ta:
  - `class Ogrenci { ... };` (Sınıf)
  - `Ogrenci ali;` (ali bir nesnedir)
- Bir sınıfından çok sayıda nesne oluşturulabilir.
- Her nesnenin kendi verileri vardır, ama aynı üye fonksiyonları kullanır.

# Basit Sınıf Örneği: Nokta

```
class Nokta {  
public:  
    int x;  
    int y;  
  
    void yazdir() {  
        cout << "(" << x << ", " << y << ")" << endl;  
    }  
};
```

- Nokta bir sınıfıtır.
- x ve y: veri elemanları (özellikler).
- yazdir(): üye fonksiyon (davranış).

# Sınıftan Nesne Oluşturma

```
int main() {  
    Nokta p1;          // p1 bir Nokta nesnesi  
    p1.x = 3;  
    p1.y = 5;  
  
    Nokta p2;          // p2 başka bir Nokta nesnesi  
    p2.x = -1;  
    p2.y = 10;  
  
    p1.yazdir();      // (3, 5)  
    p2.yazdir();      // (-1, 10)  
  
    return 0;  
}
```

- p1 ve p2 aynı sınıfından türeyen iki farklı nesnedir.
- Her nesnenin x ve y değerleri birbirinden bağımsızdır.

# Kapsülleme (Encapsulation)

- Amaç: Veriyi korumak ve **kontrollü erişim** sağlamak.
- C++ erişim belirleyicileri:
  - **public**: Her yerden erişilebilir.
  - **private**: Sadece sınıfın içinden erişilebilir.
  - **protected**: (kalitimda kullanılır, bu derste detay yok)
- İyi tasarımda:
  - Veri elemanları genelde **private** olur.
  - Dış dünyaya **public üye fonksiyonlar** ile erişim sağlanır (get/set fonksiyonları).

## Örnek: Banka Hesabı Sınıfı

```
class BankaHesabi {  
private:  
    double bakiye;          // dışarıdan direkt değiştirilemesin  
public:  
    void paraYatir(double miktar) {  
        bakiye += miktar;  
    }  
    void paraCek(double miktar) {  
        if (miktar <= bakiye)  
            bakiye -= miktar;  
        else  
            cout << "Yetersiz bakiye!" << endl;  
    }  
    double bakiyeGoster() {  
        return bakiye;  
    }  
};
```

# BankaHesabi Kullanımı

```
int main() {
    BankaHesabi hesap;

    hesap.paraYatir(1000);                      // bakiye = 1000
    hesap.paraCek(300);                          // bakiye = 700

    cout << "Bakiye: "
    << hesap.bakiyeGoster() << endl;

    // hesap.bakiye = -9999;      // HATA: private

    return 0;
}
```

- bakiye değişkeni dışarıya kapalıdır.
- Sadece paraYatir, paraCek, bakiyeGoster fonksiyonları ile kontrol edilebilir.

# Kurucu Fonksiyon (Constructor)

- Nesne oluşturulduğunda **otomatik çalışan** özel fonksiyon.
- İsmi, sınıf ile aynı olmalıdır.
- Geri dönüş türü yazılmalıdır (void bile değil!).
- Genellikle:
  - Veri elemanlarını başlangıç değerine ayarlamak için kullanılır.

## Örnek: Kuruculu Nokta Sınıfı

```
class Nokta {  
private:  
    int x;  
    int y;  
  
public:  
// Kurucu fonksiyon  
Nokta(int xDeger, int yDeger) {  
    x = xDeger;  
    y = yDeger;  
}  
  
void yazdir() {  
    cout << "(" << x << ", " << y << ")" << endl;  
}  
};
```

# Kurucunun Kullanılması

```
int main() {  
    Nokta p1(3, 5);      // x=3, y=5  
    Nokta p2(-1, 10);    // x=-1, y=10  
  
    p1.yazdir();         // (3, 5)  
    p2.yazdir();         // (-1, 10)  
  
    return 0;  
}
```

- Nesne oluştururken parametre vererek başlangıç değerlerini ayarladık.
- Kurucu sayesinde sonradan x ve y ataması yapmamıza gerek kalmadı.

## Bu derste neler gördük?

- Nesne yönelimli programlamaının motivasyonu.
- Sınıf ve nesne kavramları:
  - Kendi veri tipimizi tanımlama.
  - Aynı sınıfın birçok nesne oluşturma.
- Kapsülleme:
  - public ve private kullanımı.
  - Veriye kontrollü erişim (get/set, paraYatır, paraCek vb.).
- Kurucu fonksiyon:
  - Nesne oluşturulurken otomatik çalışan fonksiyon.
  - Başlangıç değerlerini ayarlama.

# Teşekkürler