Programlama Dillerinin Prensipleri

Hafta 8 – Nesne Yönelimli Programlama

Dr. Öğr. Üyesi M. Fatih ADAK

İçerik

- Nesnelerin Görünüşleri
- Nesneye Dayalı Düşünme
- Sınıf Hiyerarşisi
- this terimi
- Nitelikler
- İç içe Sınıflar
- Yıkıcı Metot
- Erişim Niteleyicileri
- Kalıtım
- Overload ve Override

Tanım

- Nesneye dayalı programlama kompleks sistemleri yapılandırma ve yönetmeye imkan sağlamaktadır.
- Günlük hayatımızda kullandığımız şeylerin taklit edilmesi esasına dayanılır.
- Nesnelerin tanımlanmasını ve kullanımını destekleyen dillere nesne yönelimli dil denir.
- Tüm nesneye yönelik diller, programlama dilleri literatüründe sınıf ve alt sınıf kavramını ilk defa tanıtan SIMULA67 diline dayanmaktadır.

Nesnelerin Görünüşleri

- Bir problemin çözümünde bir nesne herhangi bir varlık olarak ifade edilebilir.

Harici Görünüş

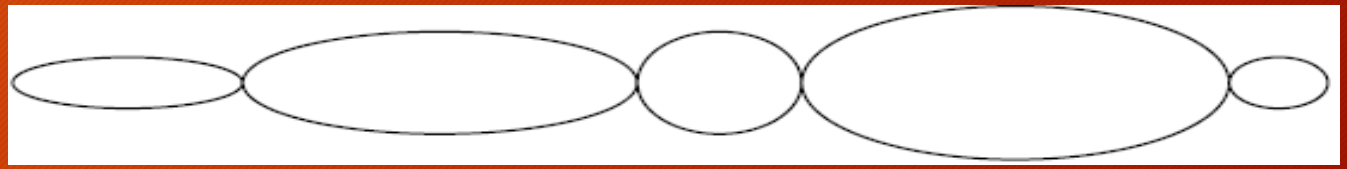
Kargo Takip Sistemi

- Kargo
- Müşteri
- Nakliye

Bu kullanılabilecek varlıklardır. Bunlar nesnelerin harici görünüşüdür.

Farklı Özelliklere Sahip Aynı Tür Nesneler

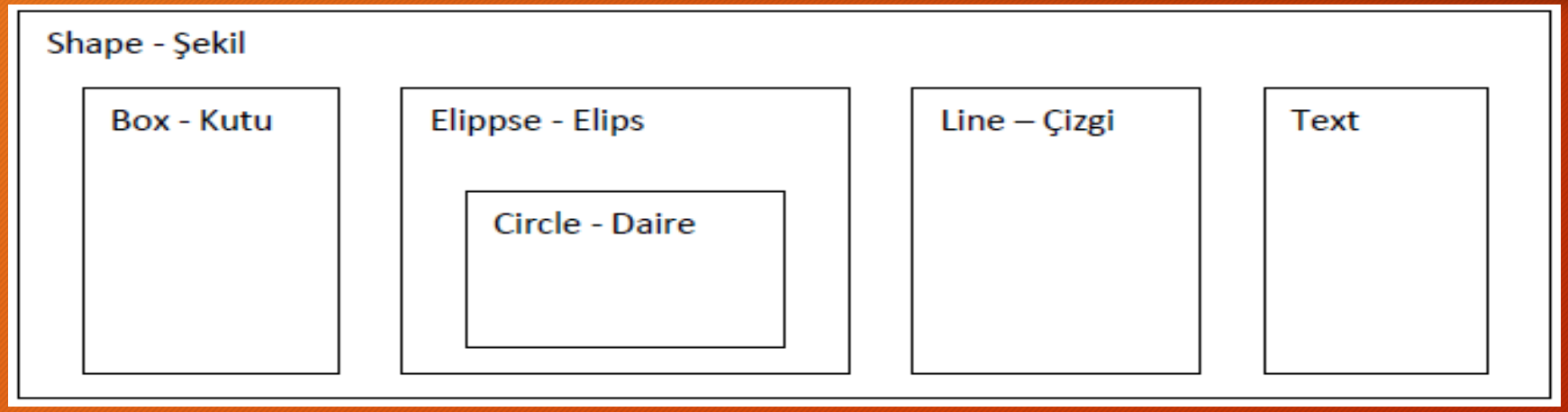
- Genişlik ve yükseklikleri farklı fakat aynı elips nesneleri



Nesneye Dayalı Düşünme

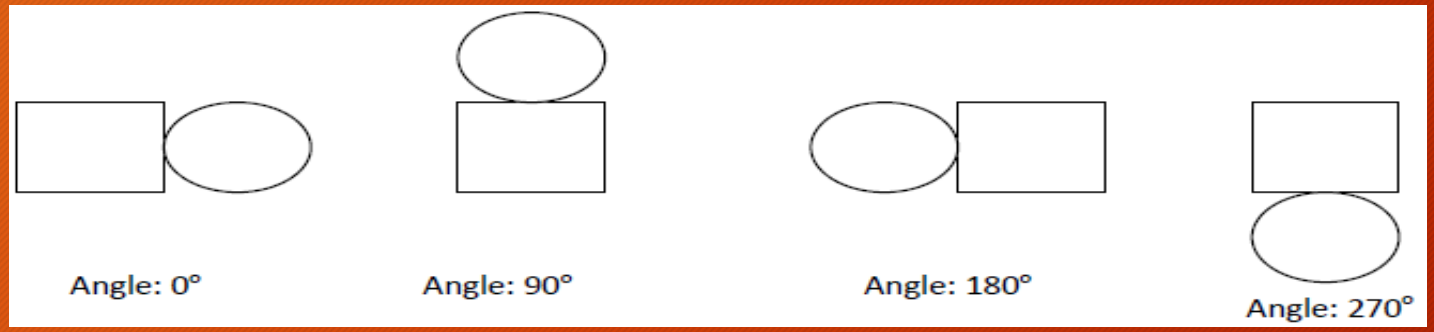
- Object (nesne) : Veriler ve işlemler topluluğu
- Class (sınıf) : Nesneler kümesinin bir tanımı
- Subclass (altsınıf) : Bir sınıfın ilave özelliklere sahip bir kümesi
- Instance (örnek) : Bir sınıfın bir nesnesi için kullanılan teknik bir terim
- Method (metod) : Bir işlemi yürüten bir altprogram içeriği
- Message (mesaj) : Bir metodu işlemek için bir istek, bir alt programın çağrılması

Sınıf Hiyerarşisi



- Bir sınıfın içine girmiş sınıf alt sınıftır. Tersi süper sınıf veya üst sınıftır.
- Bir nesneye gelen bir mesaj, bir altprogramın çağrılmasına benzer.

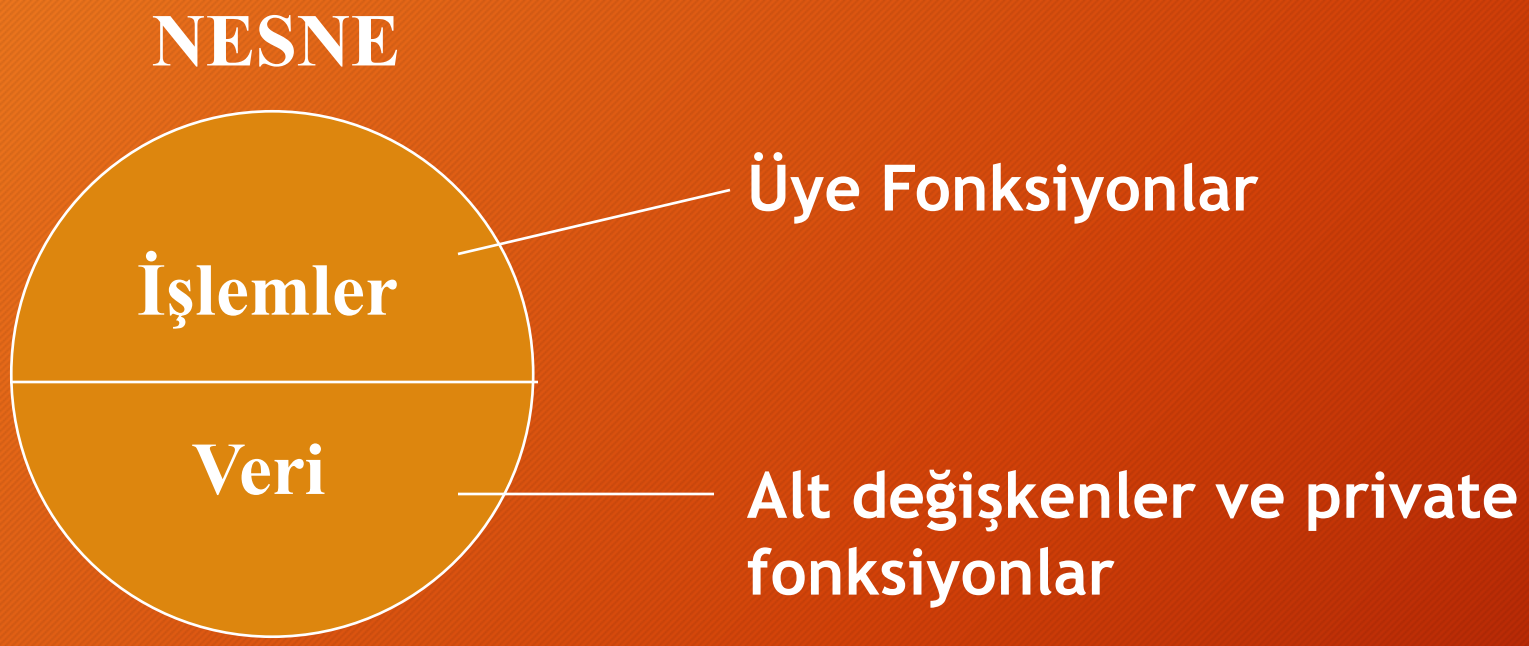
Nesne Türerilmesine Örnekler



Sınıf

- Kapsüllemeyi, veri gizlemeyi ve çok biçimliliği mümkün hale getiren yapı.
- Kullanıcı tanımlı bir türdür.
- Veri ve metotları içerir, dolayısıyla mesaj gönderilip cevap alınabilir.

Nesne Terimi



Nesnenin Oluşturulması

- Sınıftan türetilecek bir nesnede içerisinde bulunacak elemanlar ilk değerlerini almaları gerekir.
- Bu durumda nesne oluşma aşamasında bir metot çağrılmalıdır.
- Bu fonksiyon kurucu veya yapıcı (constructor) fonksiyondur.

```
// Java   Ogrenci o1 = new Ogrenci("Ahmet Almaz",159);
class Ogrenci{
      private String isim;
      private int numara;

      public Ogrenci(String ism,int nmr){
            isim=ism;
            numara = nmr;
      }
}
```

```
//C++     Ogrenci o1("Ahmet Almaz",159);
class Ogrenci{
      private:
            string isim;
            int numara;
      public:
            Ogrenci(String ism,int nmr){
              isim=ism;
              numara = nmr;
            }
};
```

this Terimi

- Sınıf hiyerarşisinde, o anda oluşturulan nesneyi ifade etmek için kullanılır.
- Bazı durumlarda kullanımı gereklidir.
- Ayrıca okunabilirliği arttırmak için de kullanılmaktadır.

this teriminin kullanımı

```
class Kisi{
public String isim;
...

public Kisi(String isim){
        isim=isim;
        yas=0;
        boy=20;
        kilo=4;
    }
...
}
```

```
// doğrusu
public Kisi(String isim){
        this.isim=isim;
        yas=0;
        boy=20;
        kilo=4;
}
```

Nitelik Tanımı (Property)

- Nitelikler bazı programlama dillerinde desteklenmektedirler.
- Amaç sınıftaki bir veya birden fazla alt alanı okuma ve yazma yönünden korumaya almak veya yönetebilmektir.
- Nitelik desteklemeyen programlama dillerinde bu metotlar yardımıyla yapılır.

```
class Kisi
   {
      private double kilo;
      private double kalori;

      public Kisi()
      {
         kilo = 4;
         kalori = 0;
      }


// C# dili
```

```
public double Kalori
      {
         get
         {
            return kalori;
         }
         set
         {
            kalori = value;
        if (kalori > 1000)
           kilo += (kalori / 1000);
         }
      }
      public double Kilo
      {
         get
         {
            return kilo;
         }
      }
   }
```

Yıkıcı Metotlar

- Çöp toplayıcı araçlarına sahip diller için dikkat edilmesine gerek olmayan metotlardır.
- Fakat söz konusu C++ gibi bir dil ise Heap bellek bölgesinde oluşturulan nesnelerin belleğe geri iadesi gerekeceğinden yıkıcı metotların önemi büyüktür.

Yıkıcı Metotlar

- C++ dilinde programcı tarafından çağrılabilen yıkıcı metot java ve C#, Python gibi dillerde çöp toplayıcı tarafından çağrılır.

```java
//Java
public class Arac{
    @Override
    protected void finalize() throws Throwable {
        // Çöp toplayıcı tarafından çağrılır.
    }
}
```

```csharp
// C#
 class Arac : IDisposable
    {
        ~Arac()
        {
// Çöp toplayıcı tarafından
        }

        public void Dispose()
        {
// Programcı çağırabilir
// Nesnenin yok edilmesini garantilemez
        }
    }
```

```python
# Python
class Arac:
    def __del__(self):
      # Çöp toplayıcı tarafından
```

Erişim Niteleyicileri

- Sınıflarda bilgi gizleme, dış kullanıma açma, kalıtım için kullanma gibi işlemler için erişim niteleyicileri kullanılır.

  Varsayılan: Bazı dillerde

  C++ (struct) public, C++(class) private

  C# private

  Java aynı pakette ise sınıf dışından erişilebilir.

  private: Sadece sınıf içerisinden erişilebilir.

  protected: Sınıf dışından sadece kalıtım yolu ile erişilebilir.

  public: Sınıf içi ve dışında erişilebilir.

Erişim Niteleyicileri

- Bilgi gizleme iki çeşit değişikliğe imkan sağlar.
  - Yürütme (implementation) Değişiklikleri
    - Eğer bir nesne ile olan bütün etkileşimler o nesnenin arayüzü üzerinden yapılıyorsa o zaman arayüzün arkasında gizlenen algoritmalar ve veri yapıları yürütme yardımıyla değiştirilebilir.
  - Kalıtım Değişiklikleri
    - Bir üst sınıfa olan bütün etkileşimler arayüzler üzerinden olursa program alt sınıfların eklenmesi ile genişletilebilir.

Kalıtım

- Birçok tür tamamen bağımsız olmayacak ve bazı özellikleri başka türlere benzeyecektir.
- Bu durumda bu özelliklerin bir üst sınıftan alınması geliştirme sürecini hızlandırır.
- Kalıtım alınan sınıfa üst sınıf denir ve programlama dillerinde yukarıya mesaj gönderme şekli;
  - Java super
  - C# base
  - C++ Sınıf adı ile

Kalıtım

- C++ gibi bazı dillerde çoklu sınıf kalıtımı desteklenir.
- Java ve C# dillerinde çoklu arayüz kalıtımı desteklenmektedir.
- Çok biçimlilik yine kalıtım ile desteklenmektedir.

Çoklu Kalıtım ve Diamond Problemi

```
class Canli{
	public:
		double kilo;
};

class Kus : public Canli{
	public:
		Kus(){
			kilo=0.25;
		}
		void YemekYe(double kalori){
			kilo += (kalori/10000);
		}
		double Kilo(){
			return kilo;
		}
};
class At : public Canli{
	public:
		At(){
			kilo=100;
		}
		void YemekYe(double kalori){
			kilo += (kalori/1000);
		}
		double Kilo(){
			return kilo;
		}
};
class Pegasus : public At, public Kus{
	public:
		Pegasus(){

		}
};
```

```
int main(){
	Pegasus *p = new Pegasus();
	p->YemekYe(1500);
	cout<<p->Kilo();
	delete p;
	return 0;
}
```

Overload (Aşırı Yükleme) ve Override (Ezme) Terimleri

- Overload bir metoda yeni anlamlar kazandırarak çok biçimliliği destekler.
- Override ise metodu yeniden tanımlamaktır. Dolayısıyla eski metodu ezmiş olacaktır.

C# dilinde operatörlerin aşırı yüklenmesi	C#'ta ilkel türe benzer kullanım

```
class Sayi
   {
      public int deger;
      public Sayi(int dgr)
      {
         deger = dgr;
      }
      public static Sayi operator +(Sayi s1, Sayi s2)
      {
         return new Sayi(s1.deger+s2.deger);
      }
      public override string ToString()
      {
         return this.deger.ToString();
      }
   }
```

```
static void Main(string[] args)
      {
         Sayi x = new Sayi(10);
         Sayi y = new Sayi(5);
         Console.WriteLine(x+y);
         Console.ReadKey();
      }
static void Main(string[] args)
      {
         int x = 10, y = 5;
         Console.WriteLine(x+y);
         Console.ReadKey();
      }
```

Kaynaklar

- Yumusak N., Adak M.F. *Programlama Dillerinin Prensipleri*. 1. Baskı, Seçkin Yayıncılık, 2018
- Sebesta, Robert W. *Concepts of programming languages*.11 ed. Pearson Education Limited, 2016.
- Sethi, Ravi. *Programming languages: concepts and constructs*. Addison Wesley Longman Publishing Co., Inc., 1996.
- Watt, David A. *Programming language design concepts*. John Wiley & Sons, 2004.
- Malik, D. S., and Robert Burton. *Java programming: guided learning with early objects*. Course Technology Press, 2008.
- Waite, Mitchell, Stephen Prata, and Donald Martin. *C primer plus*. Sams, 1987.
- Hennessey, Wade L. *Common Lisp*. McGraw-Hill, Inc., 1989.
- Liang, Y. Daniel. Introduction to Java programming: brief version. pearson prentice hall, 2009.
- Yumusak N., Adak M.F. *C/C++ ile Veri Yapıları ve Çözümlü Uygulamalar*. 2. Baskı, Seçkin Yayıncılık, 2016