# 3. Ders: JAVA ile Nesne Yönelimli Programlama
# Nesne Yönelimli Düşünme (Object-Oriented Thinking)

Fırat Üniversitesi Teknoloji Fakültesi Yazılım Mühendisliği Bölümü

YMH112 Algoritma ve Programlama-II

Dr. Öğr. Üyesi Yaman Akbulut

# JAVA ile Nesne Yönelimli Programlama

- http://www.kriptarium.com/algoritma.html (Yardımcı kaynak)

- JAVA ile Nesne Yönelimli Programlama (Ders: 11-20) video.

    - Ders 11: Java İsimlendirme Kuralları (izle)
    - Ders 12: Java Dilinde Sınıf ve Nesneler Nasıl Kodlanır? (izle)
    - Ders 13: Örnek Kodlar ile Nesne ve Sınıflar (izle)
    - Ders 14: Java Programları Yazarken Kullanılan Yaygın Yaklaşım (izle)
    - Ders 15: Nesneleri Hazırlamanın Farklı Yolları (izle)
    - Ders 16 Anonim Nesneler (Anonymous Object) (izle)
    - Ders 17: Örnek Bir Uygulama (izle)
    - Ders 18: Yapıcı Metotlar (Constructor Methods) (izle)
    - Ders 19: Java'da Constructor Aşırı Yüklemesi (izle)
    - Ders 20: Static Anahtar Sözcüğü (izle)

# Java Keywords

| | | | |
|---|---|---|---|
| abstract | double | int | super |
| assert | else | interface | switch |
| boolean | enum | long | synchronized |
| break | extends | native | this |
| byte | final | new | throw |
| case | finally | package | throws |
| catch | float | private | transient |
| char | for | protected | try |
| class | goto | public | void |
| const | if | return | volatile |
| continue | implements | short | while |
| default | import | static | |
| do | instanceof | strictfp | |

Kaynak: Introduction to JAVA Programming 10th Edition, Y. Daniel Liang

# Sınıf Soyutlama ve Kapsülleme (Class Abstraction and Encapsulation)

Sınıf soyutlaması, bir sınıf uygulamasının (implement) sınıfın kullanımından ayrılmasıdır.

Uygulamanın (Implementation) ayrıntıları kapsüllenir ve kullanıcıdan gizlenir.

Bu, sınıf kapsülleme olarak bilinir.

# Sınıf Soyutlama ve Kapsülleme (Class Abstraction and Encapsulation)

Java, birçok soyutlama düzeyi sağlar ve sınıf soyutlaması, sınıf uygulamasını (implementation) sınıfın nasıl kullanıldığından ayırır.

Bir sınıfın yaratıcısı, sınıfın işlevlerini tanımlar ve kullanıcının sınıfın nasıl kullanılabileceğini bilmesini sağlar.

Sınıfın dışından erişilebilen metotların ve alanların koleksiyonu, bu üyelerin nasıl davranmasının beklendiğinin açıklamasıyla birlikte, sınıfın sözleşmesi olarak hizmet eder.

# Sınıf Soyutlama ve Kapsülleme (Class Abstraction and Encapsulation)

Şekil 10.1'de gösterildiği gibi, sınıfın kullanıcısının sınıfın nasıl uygulandığını bilmesine gerek yoktur.

Uygulamanın ayrıntıları kapsüllenir ve kullanıcıdan gizlenir. Buna sınıf kapsülleme denir.

Örneğin, bir Cember nesnesi oluşturabilir ve alanın nasıl hesaplandığını bilmeden dairenin alanını bulabilirsiniz.

Bu nedenle, bir sınıf aynı zamanda soyut veri türü (ADT) olarak da bilinir.

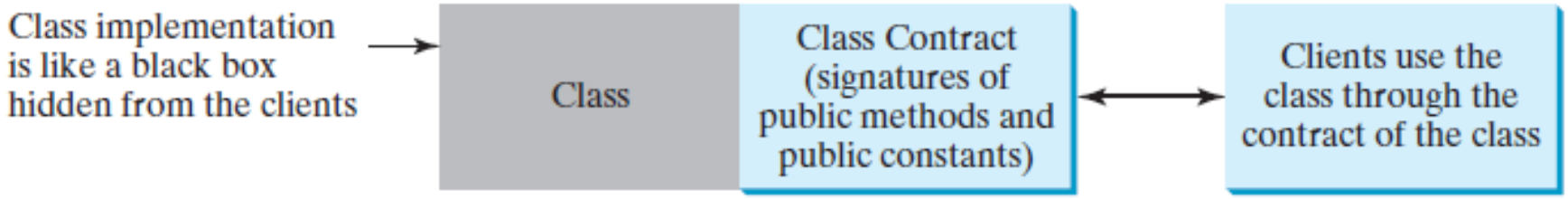



**FIGURE 10.1** Class abstraction separates class implementation from the use of the class.

# Sınıf Soyutlama ve Kapsülleme (Class Abstraction and Encapsulation)

Sınıf soyutlaması ve kapsülleme aynı madalyonun iki yüzüdür.

Pek çok gerçek hayat örneği, sınıf soyutlaması kavramını göstermektedir. Örneğin, bir bilgisayar sistemi oluşturmayı düşünelim. Kişisel bilgisayarınızda birçok bileşen vardır - bir CPU, bellek, disk, ana kart, fan vb. Her bileşen, özellikleri ve metotları olan bir nesne olarak görülebilir.

Bileşenlerin birlikte çalışmasını sağlamak için, yalnızca her bir bileşenin nasıl kullanıldığını ve diğerleriyle nasıl etkileşime girdiğini bilmeniz gerekir.

Bileşenlerin kendi içinde nasıl çalıştığını bilmenize gerek yoktur. İç uygulama (implementation) kapsüllenir ve sizden gizlenir.

Bir bileşenin nasıl uygulandığını (implement) bilmeden bir bilgisayar oluşturabilirsiniz.

# Sınıf Soyutlama ve Kapsülleme (Class Abstraction and Encapsulation)

Bilgisayar sistemi benzetmesi, nesne yönelimli yaklaşımı tam olarak yansıtır.

Her bileşen, bileşen için sınıfın bir nesnesi olarak görülebilir.

Örneğin, fan boyutu ve hızı gibi özellikler ve başlatma ve durdurma gibi metotlarla bir bilgisayarda kullanılmak üzere her türlü fanı modelleyen bir sınıfınız olabilir.

Belirlenmiş bir fan, bu sınıfın belirli özellik değerlerine sahip bir örneğidir.

# Nesnelerle Düşünme

Prosedüral programlama paradigması metotları tasarlamaya odaklanır.

Nesne yönelimli paradigma, verileri ve yöntemleri birlikte nesneler halinde birleştirir.

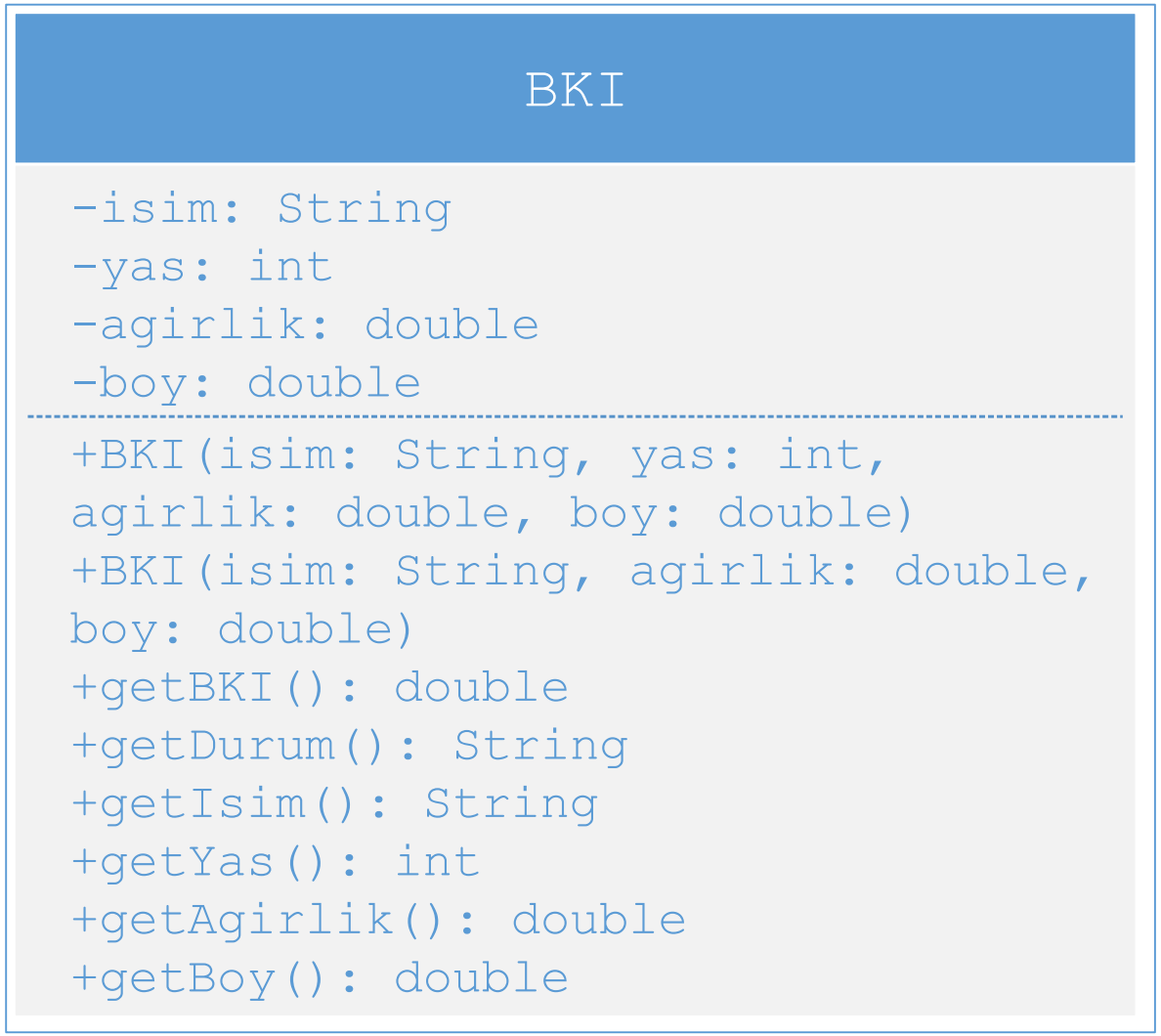
Nesneye yönelik paradigmayı kullanan yazılım tasarımı, nesnelere ve nesneler üzerindeki işlemlere odaklanır.

# BKI Hesapla ve Yorumla

```java
import java.util.Scanner;

public class BKIHesaplaveYorumla {
    public static void main(String[] args) {
        Scanner giris = new Scanner(System.in);

        System.out.print("Lutfen kilonuzu giriniz (kg): ");
        double agirlik = giris.nextDouble();
        System.out.print("Lutfen boyunuzu giriniz (m): ");
        double boy = giris.nextDouble();

        // BKI Hesapla
        double bki = agirlik /  (boy * boy);

        // Sonuçları göster
        System.out.println("BKI: " + bki);
        if (bki < 18.5)
            System.out.println("Zayif");
        else if (bki < 25)
            System.out.println("Normal");
        else if (bki < 30)
            System.out.println("Kilolu");
        else
            System.out.println("Obez");
    }
}
```

# BKI Sınıfı

BKI sınıfı, BKI bilgisini kapsüllüyor.




Kişinin ismi
Kişinin yaşı
Kişinin ağırlığı (kg)
Kişinin boyu (m)

BKI nesnesi oluştur (isim, yaş, ağırlık ve boy)
BKI nesnesi oluştur (isim, ağırlık ve boy)
BKI döndür.
Durum döndür.
İsim döndür.
Yaş döndür.
Ağırlık döndür.
Boy döndür.

# BKI Sınıfı kullanımı

```
1  public class BKISinifKullanimi {
2      public static void main(String[] args) {
3          BKI bki1 = new BKI("Yaman Akbulut", 43, 84, 1.78);
4          System.out.println(bki1.getIsim() + " icin BKI: "
5          + bki1.getBKI() + " " + bki1.getDurum());
6
7          BKI bki2 = new BKI("Diger kisi", 60, 1.70);
8          System.out.println(bki2.getIsim() + " icin BKI: "
9          + bki2.getBKI() + " " + bki2.getDurum());
10     }
11 }
```

```
C:\Program Files\Java\jdk-15.0.1\bin\yeni2>javac BKISinifKullanimi.java

C:\Program Files\Java\jdk-15.0.1\bin\yeni2>java BKISinifKullanimi
Yaman Akbulut icin BKI: 26.51 Kilolu
Diger kisi icin BKI: 20.76 Normal
```

# BKI Sınıfı implement

```java
public class BKI {
    private String isim;
    private int yas;
    private double agirlik; // kg
    private double boy; // m

    public BKI(String isim, int yas, double agirlik, double boy) {
        this.isim = isim;
        this.yas = yas;
        this.agirlik = agirlik;
        this.boy = boy;
    }

    public BKI(String isim, double agirlik, double boy) {
        this(isim, 20, agirlik, boy);
    }

    public double getBKI() {
        double bki = agirlik / (boy * boy);
        return Math.round(bki * 100) / 100.0;
    }

    public String getDurum() {
        double bki = getBKI();
        if (bki < 18.5)
            return "Zayif";
        else if (bki < 25)
            return "Normal";
        else if (bki < 30)
            return "Kilolu";
        else
            return "Obez";
    }

    public String getIsim() {
        return isim;
    }

    public int getYas() {
        return yas;
    }

    public double getAgirlik() {
        return agirlik;
    }

    public double getBoy() {
        return boy;
    }
}
```

# Prosedür vs Nesne Yönelimli

Bu örnek, nesne yönelimli paradigmanın prosedüral paradigmaya göre avantajlarını göstermektedir.

Prosedüral paradigma, metotların tasarlanmasına odaklanır.

Nesne yönelimli paradigma, verileri ve metotları birlikte nesneler halinde birleştirir.

Nesneye yönelik paradigmayı kullanan yazılım tasarımı, nesnelere ve nesneler üzerindeki işlemlere odaklanır.

Nesne yönelimli yaklaşım, prosedüral paradigmanın gücünü, işlemlerle verileri nesnelere entegre eden ek bir boyutla birleştirir.

# Prosedür vs Nesne Yönelimli

Prosedürel programlamada, veriler üzerindeki veriler ve işlemler ayrıdır ve bu metodoloji verilerin metotlara aktarılmasını gerektirir.

Nesneye yönelik programlama, verileri ve bunlarla ilgili işlemleri bir nesneye yerleştirir.

Bu yaklaşım, prosedürel programlamanın doğasında bulunan birçok sorunu çözer.

Nesne yönelimli programlama yaklaşımı programları, tüm nesnelerin hem özniteliklerle hem de etkinliklerle ilişkilendirildiği gerçek dünyayı yansıtacak şekilde düzenler.

# Prosedür vs Nesne Yönelimli

Nesnelerin kullanılması, yazılımın yeniden kullanılabilirliğini artırır ve programların geliştirilmesini ve bakımını kolaylaştırır.

Java'da programlama nesneler açısından düşünmeyi içerir; bir Java programı, işbirliği yapan nesnelerin bir koleksiyonu olarak görülebilir.

# Örnek 1a:

```java
public class TestOgrenci {
    public static void main(String args[]){
        Ogrenci nesne1 = new Ogrenci(111,"Yaman Akbulut",5000f);
        Ogrenci nesne2 = new Ogrenci(222,"Fatih Ozkaynak",6000f);
        nesne1.bilgileriGoster();
        nesne2.bilgileriGoster();
    }
}
```

```java
public class Ogrenci {
    int ogrenciNo;
    String isim;
    float burs;

    Ogrenci(int ogrenciNo, String isim, float burs){
        ogrenciNo = ogrenciNo;
        isim = isim;
        burs = burs;
    }

    void bilgileriGoster(){
        System.out.println(ogrenciNo + " " + isim
        + " " + burs);
    }
}
```

```
C:\Program Files\Java\jdk-15.0.1\bin\yeni2>javac Ogrenci.java

C:\Program Files\Java\jdk-15.0.1\bin\yeni2>javac TestOgrenci.java

C:\Program Files\Java\jdk-15.0.1\bin\yeni2>java TestOgrenci
0 null 0.0
0 null 0.0
```

# Örnek 1b:

```java
public class TestOgrenci {
    public static void main(String args[]){
        Ogrenci nesne1 = new Ogrenci(111,"Yaman Akbulut",5000f);
        Ogrenci nesne2 = new Ogrenci(222,"Fatih Ozkaynak",6000f);
        nesne1.bilgileriGoster();
        nesne2.bilgileriGoster();
    }
}
```

```java
public class Ogrenci {
    int ogrenciNo;
    String isim;
    float burs;

    Ogrenci(int ogrenciNo, String isim, float burs){
        this.ogrenciNo = ogrenciNo;
        this.isim = isim;
        this.burs = burs;
    }

    void bilgileriGoster(){
        System.out.println(ogrenciNo + " " + isim
        + " " + burs);
    }
}
```

```
C:\Program Files\Java\jdk-15.0.1\bin\yeni2>javac Ogrenci.java

C:\Program Files\Java\jdk-15.0.1\bin\yeni2>java TestOgrenci
111 Yaman Akbulut 5000.0
222 Fatih Ozkaynak 6000.0
```

# Örnek 1c:

```java
public class TestOgrenci {
    public static void main(String args[]){
        Ogrenci nesne1 = new Ogrenci(111,"Yaman Akbulut",5000f);
        Ogrenci nesne2 = new Ogrenci(222,"Fatih Ozkaynak",6000f);
        nesne1.bilgileriGoster();
        nesne2.bilgileriGoster();
    }
}
```

```java
public class Ogrenci {
    int ogrenciNo;
    String isim;
    float burs;

    Ogrenci(int o, String i, float b){
        ogrenciNo = o;
        isim = i;
        burs = b;
    }

    void bilgileriGoster(){
        System.out.println(ogrenciNo + " " + isim
        + " " + burs);
    }
}
```

```
C:\Program Files\Java\jdk-15.0.1\bin\yeni2>javac Ogrenci.java

C:\Program Files\Java\jdk-15.0.1\bin\yeni2>java TestOgrenci
111 Yaman Akbulut 5000.0
222 Fatih Ozkaynak 6000.0
```

# Ödev 1:

en, boy ve yukseklik özelliklerine sahip Kutu adında bir sınıf tasarlayınız.

Daha sonra TestKutu adında bir uygulama ile bir kutu örneğinin hacmini hesaplayınız.

# Ödev 2:

ogrenciNo, isim, burs ve ders özelliklerine sahip Ogrenci adında bir sınıf tasarlayınız. Sınıf içinde 2 tane yapıcı (kurucu) metot yazınız. Birisi 3 parametreli (ogrenciNo, isim, burs) diğeri 4 parametreli (ogrenciNo, isim, burs, ders) olsun. Ayrıca Ogrenci sınıfı bilgileriGoster() adında bir metota sahip olsun. Ogrenci sınıfına ait UML diyagramı çiziniz ve bu sınıfı java dilinde kodlayınız.

Daha sonra TestOgrenci adında bir sınıf ile (main metotu olan) Ogrenci sınıfını test ediniz. Ogrenci sınıfından 2 nesne oluşturunuz ve nesne1 ile 3 parametreli, nesne2 ile 4 parametreli yapıcıları kullanınız. Çalışan kod sonucunda ekran çıktılarını inceleyiniz.