



Bölüm 4: Sarmalama

JAVA ile Nesne Yönelimli Programlama



Sarmalama (Encapsulation)





Bilgi Saklama (Information Hiding)

- Soyutlama olarak da adlandırılır.
- Tasarım, **ne** ve **nasıl**ı birbirinden ayırmalıdır.
- Bir metot detaylarına hakim olunmadan da kullanılabilmelidir.
- Sadece metodun ne işe yaradığını bilmek yeterli olmalıdır.
- Arabanın fren, gaz pedalı, direksiyonunu görürüz ve kullanırız.
- Ancak bu parçaların iç mekanik detaylarını bilmeyiz.



Bilgi Saklama (Information Hiding)

```
public class VeriIslemleri {  
  
    private VeritabaniBaglantisi baglanti;  
    // Kullanıcı bu metodun iç yapısını bilmese de kullanabilir.  
    public void veriEkle(String veri) {  
        baglanti = new VeritabaniBaglantisi();  
        baglanti.baglan();  
        baglanti.veriEkle(veri);  
        baglanti.baglantiKapat();  
    }  
}
```



Ön Koşullar (Preconditions)

- Bir metodun çağrılmadan önce doğru olduğu varsayılan koşullardır.

```
public class VeriIslemleri {  
    // Ön koşul: veri != null  
    public void veriEkle(String veri) {  
        // verinin null olup olmadığını kontrol et  
        // ...  
    }  
}
```



Son Koşullar (Postconditions)

- Metot çağırıldıktan sonra, ön koşullar karşılandığında beklenen durumlar.

```
public class VeriIslemleri {  
    // Son koşul: veri tabanına veri eklendiyse true döner  
    public boolean veriEkle(String veri) {  
        // veriyi ekleyip eklemediğini kontrol et  
        // ...  
        return true;  
    }  
}
```



Erişim Belirleyiciler

- **public:** Diğer sınıflar tarafından doğrudan erişilebilir.
- **private:** Sadece sınıfın içinden erişilebilir, dışarıdan erişim engellenir.
- Sınıflar genellikle public olarak tanımlanır.
- Sınıfın nitelikleri genellikle private olarak tanımlanır.
- Sınıfın niteliklerine erişmek ve değiştirmek için metotlar kullanılır.



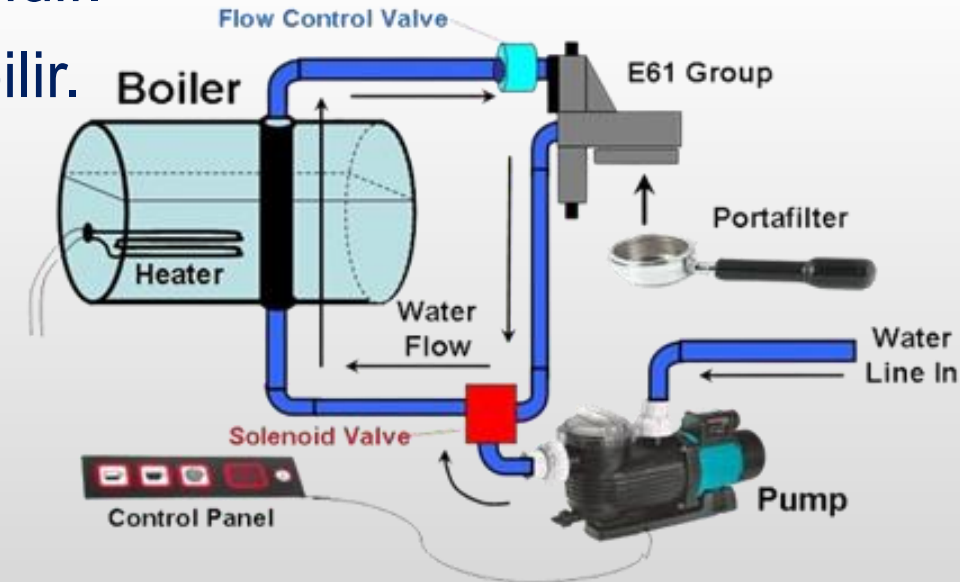
DijitalSaat Sınıfı

```
public class DijitalSaat {  
    private int saat = 0;  
    private int dakika = 0;  
    private int saniye = 0;  
  
    public int getSaniye() {  
        return this.saniye;  
    }  
  
    public void setSaniye(int saniye) {  
        if(saniye >= 0 && saniye < 60) {  
            this.saniye = saniye;  
        }  
    }  
}
```




Sınıf Tasarımı

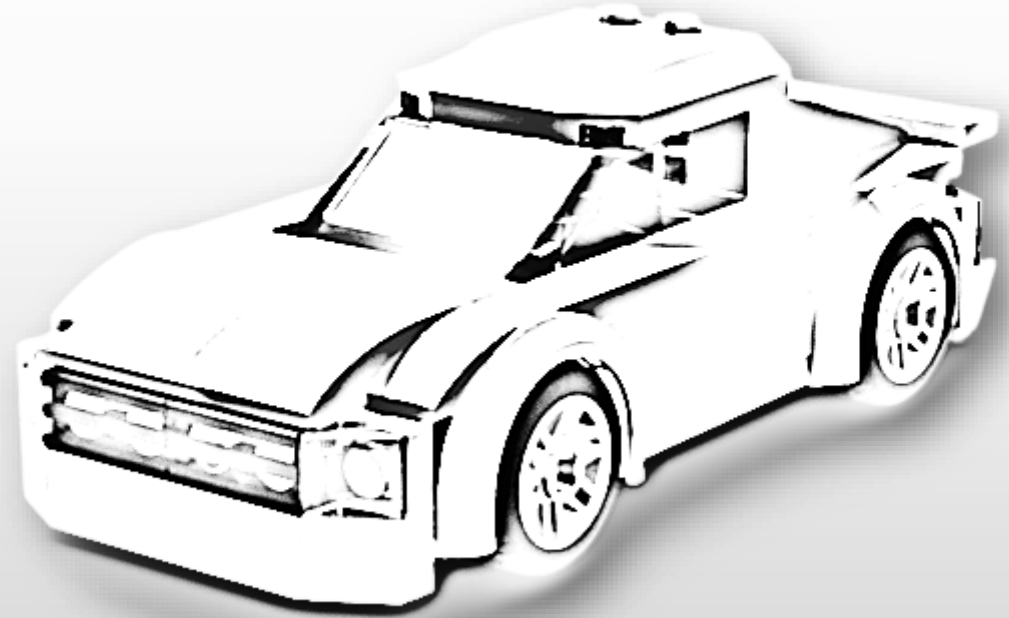
- Sınıf tanımı; **arayüz** ve **gerçekleştirim** olmak üzere iki kısma ayrılır.
- Arayüz, sınıfın ne yaptığını ve nasıl kullanılacağını belirtir.
- Gerçekleştirim, sınıfın iç mekanik detaylarını ve özel niteliklerini içerir.
- Sınıfın iç mekanikleri sadece sınıf içinde önemlidir.
- Arayüz değişse bile gerçekleştirim değişmeyebilir.
- Öz nitelikler sınıfın dışarısına gizlidir.





Arayüz

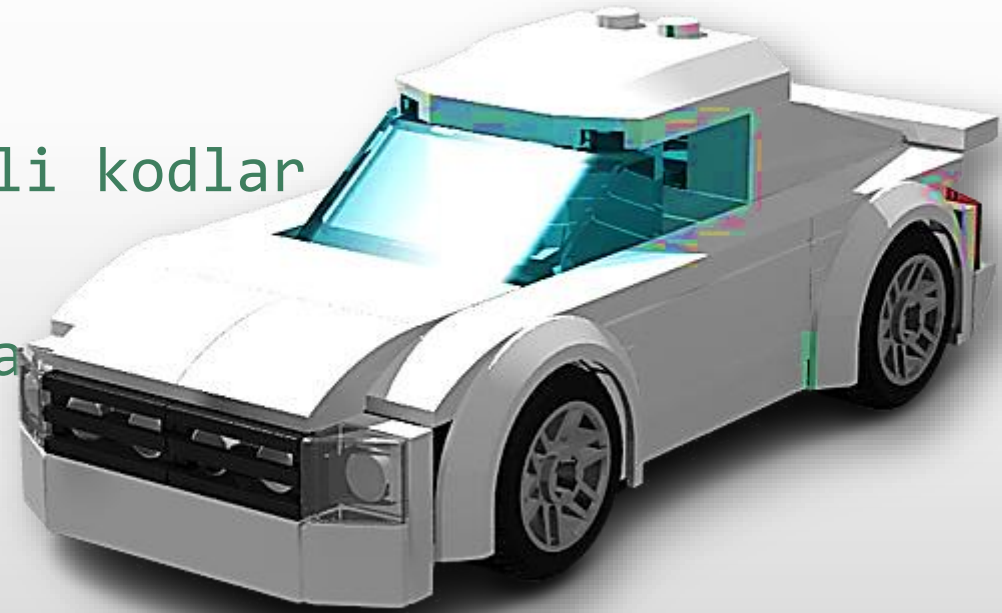
```
public interface Araba {  
    // Sınıf arayüzü  
    public void hareketEt();  
  
    public void dur();  
    // ...  
}
```





Sınıf Gerçekleştirimi

```
public class Araba {  
    // Sınıf implementasyonu  
    private String marka;  
    private int hiz;  
  
    public void hareketEt() {  
        // Hareket etmek için gerekli kodlar  
    }  
    public void dur() {  
        // Durmak için gerekli kodlar  
    }  
    // ...  
}
```





Tavsiyeler

- Sınıf içindeki tüm değişkenler private tanımlanmalı.
- Değişkenlere erişim sağlamak için public erişim metotları sağlanmalı.
- Her metodun nasıl kullanılacağını belirten yorum satırları eklenmeli.
- Sadece sınıf içerisinde kullanılan yardımcı metotlar private tanımlanmalı.



Sarmalama Temel Faydaları

- Nesne ve kullanıcısı arasındaki ilişkiyi soyutlar.
- Kullanıcı tarafından istenmeyen erişimlere karşı nesneyi korur.
- Sınıfın içsel yapısını değiştirmeden sadece arayüz değiştirilebilir.
- Nesnelerin durumunun belirli koşullara uymasını sağlar.
 - Sadece pozitif koordinatlara sahip noktaları kabul etme.
 - Saniye ve dakika değerlerinin 0 ile 59 arasında olması
 - Saat değerinin negatif olamaması gibi



Gözlemler

- Arayüzler, gerçekleştirimden (implementation) daha nadir değişir.
 - Kodun kararlı olmasını sağlar.
- Gerçekleştirim değiştiğinde, buna bağımlı kodlar da değişir.
- Gerçekleştirmeyi gizlemek, diğer sınıfların bağımlılığını azaltır.



Paketler

- Bir klasör içinde gruplanmış sınıfların oluşturduğu bir koleksiyon.
- Klasörün adı aynı zamanda paketin de adıdır.
- Her sınıf, ayrı bir dosyada tanımlanır.
- Dosyanın başında ait olduğu paket ile ilgili bir satır yer alır.
 - `package Paket_Adı;`
- Derleyiciye, paketa ait sınıfların bulunduğu dizinin yolunu (path) söyler.
- İlgili paket araması, sınıfın bulunduğu dizinde başlar.
- Paket ismi, iç içe klasörler için / veya \ yerine nokta (.) kullanır.



İçe Aktarma (Import)

- Başka bir paket içinde yer alan sınıfları kullanmak için kullanır.
 - `import Paket_Adı.Sınıf_Adı;`



Metod ve Sınıf Adı Çakışmaları

- Aynı sınıf içinde iki veya daha fazla metod aynı ismi taşıyor olabilir.
- Aynı isimli metodlar, parametre sayısı ve türlerine göre ayırt edilir.
- Metod çağırısı, bir tanım ile eşleşmezse, tip dönüşümleri yapılmaya çalışılır.
- Benzer şekilde, iki sınıf aynı adı taşıyabilir.
- Bu belirsizliği gidermek için paket adları kullanılır.



SON