# Bölüm 4: Kalıtım

## JAVA ile Nesne Yönelimli Programlama

# Kalıtım (Inheritance)

# Kalıtım

- Mevcut bir sınıfın metot ve niteliklerini miras alarak, yeni veya değiştirilmiş yeteneklerle zenginleştirerek yeni bir sınıf oluşturulmasıdır.
- Mevcut sınıf, "üst sınıf (superclass)" olarak adlandırılır.
- Yeni sınıf, "alt sınıf (subclass)" olarak adlandırılır.
- Alt sınıflara yeni nitelik ve metotlar eklenebilir, ancak miras alınan nitelikler tekrar tanımlanmamalıdır.

# Kalıtım

```
//Temel sınıf (üst sınıf)
public class Sekil {
// Temel sınıfa ait öznitelik ve metotlar
// ...
}

//Yeni sınıf, Sekil sınıfından türetilir (alt sınıf)
public class Dikdortgen extends Sekil {
// Yeni sınıfa ait öznitelik ve metotlar
// ...
}
```

# Alt ve Üst Sınıf

- Alt sınıf, üst sınıfın davranışlarını sergiler.
- Alt sınıf, üst sınıfa ek olarak kendine özgü davranışları sergiler.
- Kalıtım, "özelleştirme" olarak da adlandırılır.
- Alt sınıf, daha spesifik grubu temsil eder.
- Java sınıf hiyerarşisi, *java.lang* paketindeki *Object* sınıfı ile başlar.
- Her sınıf, doğrudan veya dolaylı olarak *Object* sınıfından miras alır.

# Kalıtımın Avantajları

- Kalıtım mevcut sınıfın metot ve niteliklerinin yeni tanımlanan sınıfta kullanılabilmesini sağlar.
- Kalıtım, mevcut bir sınıfı genişleterek yeni özellikler eklenebilmesini sağlar.
- Mevcut sınıfın metotları aşırı yüklenerek (overloading) değiştirilebilir.

```
//Yeni sınıf, Sekil sınıfından türer (alt sınıf)
public class Dikdortgen extends Sekil {
  // Yeni sınıfa ait metot ve nitelikler
  public void ozelMetod() { /* ... */ }
}
```

# Nesnenin Görünümünü Kısıtlama

- Bir nesne bir üst sınıf türüne atanarak, sadece üst sınıfa ait metotlarına erişim izni verilmesidir.
- Nesnenin gerçek sınıfına bakılmaksızın, dışarıya karşı sadece üst sınıf aracılığıyla erişilebilen kısımlar görünür.

```
Parca p = new Lastik("Petlas", "XYZ-123", "225/45R17", true);
System.out.println(p.seriNumarasi()); // izin verilir
// p.kislikMi() // izin verilmez
```

# Sınıflar Arasındaki İlişkiler

- **Is-a İlişkisi (Kalıtım, Inheritance):**
  - Bir sınıfın başka bir sınıftan türetilmesidir.
  - Türetilen sınıf, üst sınıfın bir türüdür.
  - Alt sınıf nesnesinin üst sınıfın nesnesi gibi de kullanılmasını sağlar.
- **Has-a İlişkisi (Oluşum, Composition):**
  - Bir sınıfın başka bir sınıf türünde öznitelik içermesidir.
  - Bir nesnenin içinde diğer nesnelere referanslar içerir.

# Sınıflar Arasındaki İlişkiler

```
//Is-a İlişkisi (Kalıtım)
public class Araba extends Arac{

   //Has-a İlişkisi (Oluşum)
   private Motor motor;
   // ...
}

public class Motor {
   // Motor sınıfı içeriği
}
```

# protected Erişim Seviyesi

- **public**: Sınıfın üyelerine her yerden erişilebilir.
- **private**: Sınıfın üyelerine sadece o sınıf içinden erişilebilir.
- **protected**: public ve private erişim seviyeleri arasında bir düzeydir. Sınıfın protected üyelerine, kendi içinden, alt sınıflarından ve aynı paketteki diğer sınıflar tarafından erişilebilir.
- Bir sınıfın private tanımlanmış niteliklerine, alt sınıflarından sadece üst sınıftan miras alınan public veya protected metotlar aracılığıyla erişilebilir.

# super

- Üst sınıfın metoduna **super** anahtar kelimesi ve nokta (.) ile erişilebilir.
- Her sınıf sadece kendi verilerini yönetmelidir.
- super kelimesi kullanılarak, alt sınıfın yapıcı metodu içinde üst sınıfın yapıcı metodu çağrılarak niteliklere ilk değerleri atanabilir.
- Genel metotlar, toString ve equals, aşırı yükleme ile geçersiz kılınmalıdır.
- Alt sınıfların ortak kavramları, soyut (abstract) sınıflar ile temsil edilmelidir.

# final

- final anahtar kelimesi, kalıtımı sınırlamak için kullanılabilir.
- Bir metot final tanımlanırsa, hiçbir alt sınıfta geçersiz kılınamaz (override).
- Bir sınıf final tanımlanırsa, bu sınıftan hiçbir alt sınıf türetilemez.

# Object Sınıfı

- Her sınıf, dolaylı olarak Object sınıfından türer.
- **toString**:
  - public String toString() metodu, nesneyi yazdırmak için kullanılır.
  - Sınıf adı ve nesnenin hashCode değerini içeren bir dize döner.
- **equals**:
  - public boolean equals(Object other), nesne karşılaştırmak için kullanılır.
  - == operatörü referansları karşılaştırır, bu yüzden equals kullanılmalıdır.

# equals Metodu

- Varsayılan olarak sadece referansları karşılaştırır.
- Aşırı yüklenmez ise, == operatörü gibi davranır.
- Aşırı yüklenerek davranışı değiştirilebilir.
- Metot, iki nesnenin durumunu karşılaştırmalı ve eşitse true dönmelidir.
- Hatalı durumları engellemek için tür kontrolü yapmalıdır.

# instanceof

- Nesnenin belirli bir sınıfa veya arayüze uyup uymadığını kontrol eder.
- `nesne instanceof Tür` şeklinde kullanılır.
- true veya false döner.
- null değeri bir nesne olarak kabul edilmez.
- instanceof kullanılırken null kontrolü yapılmalıdır.

# equals Metodu

```
public boolean equals(Object nesne) {

  if (nesne instanceof Nokta) {
    // nesne bir Nokta nesnesidir; dönüştürüp karşılaştır
    Nokta diger = (Nokta) nesne;
    return x == diger.x && y == diger.y;
  } else {
    // o bir Nokta değildir; eşit olamaz
    return false;
  }
}
```

# equals ve hashCode

- *If you override equals you must also override hashCode.*

```
public class Nokta {
  private double x, y;
  public Nokta(double x, double y) {this.x = x; this.y = y; }
  @Override public boolean equals(Object o) {
    return (o instanceof Nokta n) && x == n.x && y == n.y;
  }
  @Override public int hashCode() {
    return Objects.hash(x, y);
  }
  @Override public String toString() {
    return "Nokta[x=" + x + ", y=" + y + "]";
  }
}
```

# @Override Etiketi (Annotation)

- Üst sınıftan miras alınan bir metodu geçersiz kılar.
- Derleyiciye, metodun üst sınıftan miras alındığını kontrol etmesini söyler.
- Kodun daha okunabilir olmasını sağlar.
- Diğer geliştiricilere, metodun bir üst sınıftan miras alındığını belirtir.

```
public class Dikdortgen extends Sekil {
  // üst sınıftan miras alınan metodu geçersiz kıl
  @Override
  public void ciz() {
    System.out.println("Dikdörtgen çiziliyor.");
  }
}
```

# Çoklu Kalıtım

- Java'da doğrudan çoklu kalıtım desteklenmez.
- Bir sınıf, yalnızca bir sınıftan türeyebilir.
- Birden fazla sınıfın özelliklerini kullanma ihtiyacı durumunda,
  - Oluşum (composition) ve
  - Arayüzler (interface) kullanılabilir.

SON