Java ile Nesne Merkezli ve Fonksiyonel Programlama

3. Bölüm
Soyut Sınıflar
(Abstract Classes)

Akın Kaldıroğlu

www.javaturk.org

Kasım 2015

Küçük Ama Önemli Bir Konu

- Bu dosya ve beraberindeki tüm, dosya, kod, vb. eğitim malzemelerinin tüm hakları Akın Kaldıroğlu'na aittir.
- Bu eğitim malzemelerini kişisel bilgilenme ve gelişiminiz amacıyla kullanabilirsiniz ve isteyenleri http://www.javaturk.org adresine yönlendirip, bu malzemelerin en güncel hallerini almalarını sağlayabilirsiniz.
- Yukarıda bahsedilen amaç dışında, bu eğitim malzemelerinin, ticari olsun/olmasın herhangi bir şekilde, toplu bir eğitim faaliyetinde kullanılması, bu amaca yönelik olsun/olmasın basılması, dağıtılması, gerçek ya da sanal/Internet ortamlarında yayınlanması yasaktır. Böyle bir ihtiyaç halinde lütfen benimle, akin@javaturk.org adresinden iletişime geçin.
- Bu ve benzeri eğitim malzemelerine katkıda bulunmak ya da düzeltme ve eleştirilerinizi bana iletmek isterseniz çok sevinirim.
- Bol Java'lı günler dilerim.

İçerik

- Bu bölüme soyut sınıflar (abstract classes) ele alacaktır.
- Java'nın tipleri arasındaki mümkün dönüşümler incelenecektir.

Soyut Sınıf (Abstract Class)

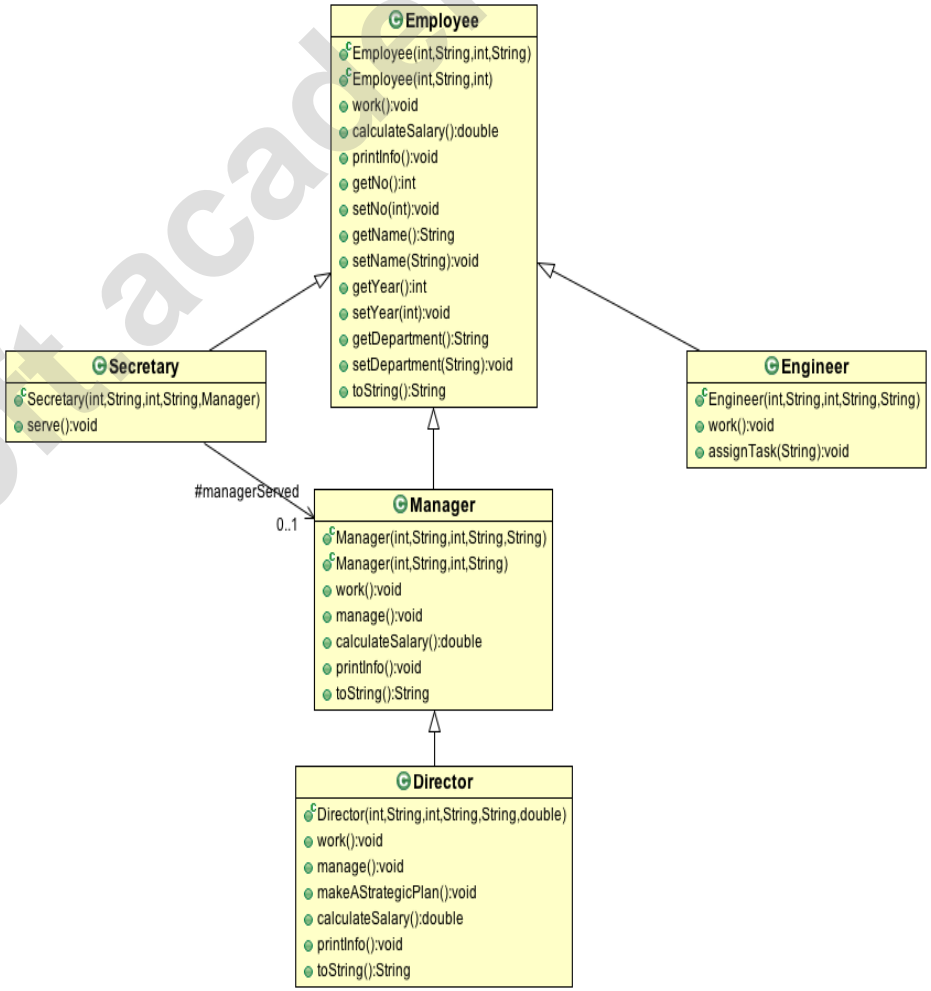
Soyut Sınıf (Abstract Class)

- Soyut sınıflara (abstract classes) iki türlü yaklaşılabilir:
  - İlki nesneler üzerindendir,
  - İkincisi ise soyut metotlar üzerindendir.

- Bu bölümde her iki yönden ele alacağız.

Soyut Sınıf – Nesne Yaklaşımı - I

- Yandaki nesne hiyerarşisini düşünün.
- Bu hiyerarşideki Employee sınıfının hiç nesnesi olmayabilir.
  - Çünkü fabrikada çalışan herkes özel bir tip Employee'dir.
  - Kimse sadece Employee değildir.



Soyut Sınıf – Nesne Yaklaşımı - II

- Bu durumda neden Employee sınıfına ihtiyaç vardır?
- Çünkü Employee sınıfı, alt sınıfları için bir kalıptır, şablondur.
  - Alt sınıfların ortak özelliklerini ve davranışlarını toplar,
  - Polymorphism için gerekli olan üst tipi temsil eder.

- Bu gibi durumlarda sınıflar, nesnesinin yaratılması önlenecek şekilde kurgulanır.

- Nesnesi oluşturulmayan sınıflara **soyut sınıf** (**abstract class**) denir.

Soyut Sınıf – Nesne Yaklaşımı - III

- Bu sınıflar soyutturlar, çünkü nesneleri yoktur, oluşturulamaz da.
  - Soyut sınıfları, singleton ile karıştırmayın!
- Bu sınıfların nesnesi oluşturulamadığı için sadece kavramsal olarak ve hiyerarşinin en tepesinde, alt sınıfları için bir kalıp oluşturmak üzere vardırlar.
- Bir sınıfı soyut yapmak için tanımlarken **abstract** anahtar kelimesi kullanılır.

```
public abstract class Employee{
    ...
}
```

Employee.java

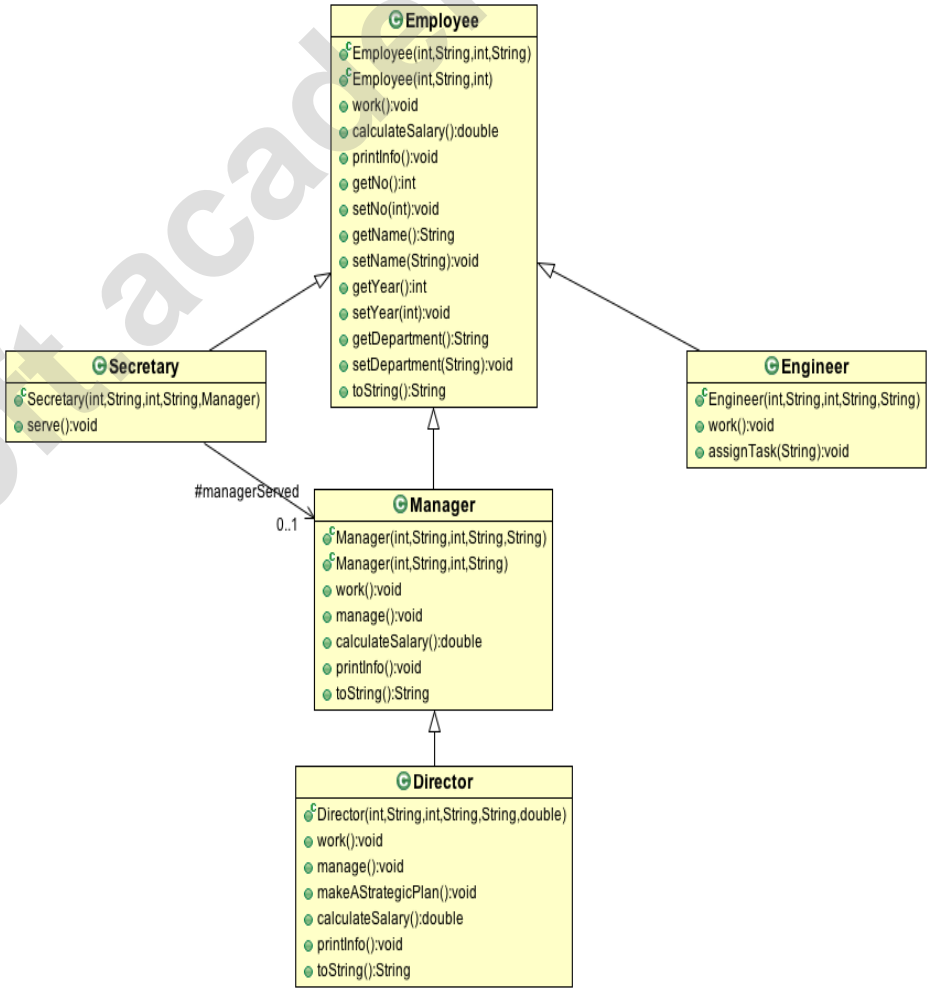
- Soyut olan Employee sınıfının nesnesinin yaratılamayacağını HR.java sınıfındaki getAnEmployee() metoduna bakarak gözlemleyin.

Soyut Metot (Abstract Method)

Soyut Sınıf – Metot Yaklaşımı - I

- Yandaki nesne hiyerarşisini ve bu hiyerarşideki soyut olan Employee sınıfının üzerindeki work() metodunu düşünün.

- Büyük bir ihtimalle, Employee sınıfının bütün alt sınıfları work() metodunu override etmişlerdir.



Soyut Sınıf – Metot Yaklaşımı - II

- Soyut bir sınıfın bir metodu, o sınıfın tüm alt sınıfları tarafından override ediliyorsa bu durumda o metot sadece arayüz sağlıyor, gerçekleştirme sağlamıyor demektir.
  - Çünkü, sınıf soyut olduğundan o sınıfın hiç bir nesnesi oluşturulmuyor dolayısıyla da metodun bu sınıftaki gerçekleştirmesi hiç bir şekilde çağrılmıyor demektir.

- Tüm alt sınıflar tarafından override edilen böyle metotlar, aslen bir kalıp ya da şablon olarak rol alırlar yani sadece arayüz sağlarlar.
  - Bu tür metotların davranışının olmasına da gerek yoktur.

Soyut Sınıf – Metot Yaklaşımı - III

- Arayüzü olan ama gerçekleştirmesi olmayan metoda **soyut metot (abstract method)** denir.

- Soyut bir metot, **abstract** anahtar kelimesiyle nitelenir.

- Soyut bir metot, kodu yani gerçekleştirmesi olmadığından, tanımında "{}" kullanmadan, sadece arayüz olarak ifade edilir ve ";" ile sonlandırılır.

```
public abstract class Employee{
   ...
   public abstract void work();
}
```

Soyut Sınıf – Soyut Metot

- Soyut metotlar ancak soyut sınıflarda bulunur.
- Bir sınıfın soyut olması için soyut metoda sahip olmasına gerek yoktur.
  - Ama bir tane bile soyut metoda sahip olan bir sınıf, soyut olarak tanımlanmak zorundadır.
  - Tipik olarak bir sınıfın soyut olmasının sebebi, en az bir tane de olsa soyut metoda sahip olmasıdır.

- Hiç bir metodu soyut olmayan bir sınıfı soyut olarak tanımlamanın ne gibi bir sebebi olabilir?

AbstractClassExample.java

Soyut – Somut Sınıf ve Miras

Soyut Sınıf ve Miras

- Alt sınıflar, soyut sınıflardan “extends” anahtar kelimesiyle miras devralırlar.
- Soyut sınıfların devralınabilecek tüm özellikleri ve davranışları alt sınıfları tarafından devralınır.
- Soyut bir sınıftan miras devralan alt sınıfın, devraldığı soyut metotlara gerçekleştirilme sağlaması yani onları implement etmesi gereklidir.
  - Aksi taktirde alt sınıf da soyut olmalıdır.

Employee.java ve Manager.java

- Manager sınıfının, soyut olan Employee sınıfından nasıl miras devraldığını ve soyut olan work() metoduna nasıl gerçekleştirme verdiğini gözlemleyin.
- Employee sınıfının alt sınıflarının soyut olarak devraldıkları work() metodunu override etmediklerinde oluşan durumu gözlemleyin.
- Manager'in, devraldığı soyut olan work() metoduna bir gerçekleştirme verdiğinde, alt sınıfı Director'un artık Manager'den devraldığı work() metodunu override etmeden kullanabileceğini de gözlemleyin.

Soyut Sınıf – Somut Sınıf - Farklılıklar

- Nesneleri oluşturulabilen sınıflara **somut** (**concrete**) sınıf denir.
- Soyut sınıfları, somut sınıflardan ayıran en temel üç özellik şunlardır:
  - Nesneleri oluşturulamaz,
  - Soyut metotlara sahip olabilirler,
  - Devralındığında soyut metotlarına davranış sağlanması gereklidir.
- Dolayısıyla soyut sınıf varsa, ondan miras devralıp, soyut metotlarına gerçekleştirme sağlayan alt sınıfları da var demektir.
- Somut sınıfların ise bir alt sınıfa ihtiyaçları yoktur.

Soyut Sınıf – Somut Sınıf - Aynılıklar

- Soyut sınıflar, somut olanlar gibi devralınmak üzere bir duruma sahip olabilirler.
- Soyut sınıflar, devralınmak üzere somut olan, soyut olan ve olmayan (somut) metotlara sahiptirler.
- Soyut sınıfların, somut olanlar gibi bir ya da daha fazla sayıda kurucu metotları vardır.
  - Eğer hiç kurucu sağlanmazsa derleyici bir tane varsayılan (no-arg constructor) kurucu sağlar.
  - Soyut sınıfların nesneleri oluşturulmasa bile, alt sınıflarının nesneleri oluşturulurken, soyut olan üst sınıfın kurucusuna ihtiyaç duyacaklardır.

Soyutluk, final ve private

- Soyut sınıflar final olamazlar.
- Soyut metotlar da private veya final tanımlanamazlar.

- Bir soyut metot sadece ve sadece public ya da protected erişim niteleyicilerini (access modifier) alabilir.
  - Soyut metotlarda static, synchronized vb. niteleyiciler de kullanılamaz.

- Bu durumlar, soyut sınıflar ve soyut metotların varlık sebeplerine aykırı bir durumdur.

- Öte taraftan bir soyut sınıf main metot içerebilir.

AbstractProblem.java

Uygulama - I

- Shape sınıfının en tepede olduğu hiyerarşiyi düşünün.
  - Shape'in üzerindeki draw(), erase(), calculateArea() ve calculateCircumference() metotlarının soyut olmalarını nasıl karşılarsınız?
  - Shape üzerindeki metotları soyut yaparak hşeyrarşiyi tekrardan düzenleyin.
- Circle, Rectangle, Square ve Triangle ise Shape'in alt sınıflarıdır ve bu metotları override ederler.
  - Metotları override ederken mümkünse "super"i kullanın.

Uygulama - II

- ShapeFactory isimli bir başka sınıfın üzerindeki createShape() isimli metodun da random olarak Shape sınıfının alt sınıflarının bir nesnesini yaratıp döndürmesini sağlayın.

- Test sınıfında da random Shape nesneleri üretirken artık Shape sınıfının nesnesini üretemeyeceğinizi de gözlemleyin.

Neden Soyut Sınıf?

Somut Sınıf

- Somut (concrete) sınıflar, alt sınıflarına hem arayüz hem de gerçekleştirme sağlarlar,
  - Çünkü bütün metotları somuttur, somut metotların hem arayüzü hem de gerçekleştirmesi vardır.

- Somut bir sınıftan miras devralan alt sınıfların önünde iki seçenek vardır:
  - Hem arayüzü hem de gerçekleştirmeyi devralmak: Bu durumda alt sınıf, devraldığı metotları override etmez.
  - Sadece arayüzü devralmak: Bu durumda alt sınıf, devraldığı metotları override eder.

Soyut Sınıf

- Soyut sınıfların soyut metotları ise gerçekleştirme sağlamadan sadece arayüz sağlarlar,

- Dolayısıyla soyut sınıfların, soyut metotlarından dolayı sadece arayüz sağlamaları da söz konusudur.

- Soyut sınıfın hangi oranda sadece arayüz sağladığı hangi oranda hem arayüz hem de gerçekleştirme sağladığı, sahip olduğu soyut-somut metot dengesiyle ilgilidir.

Neden Soyut Sınıf?

- Şu iki durumda soyut sınıf kullanmak anlamlıdır:
  - Sınıfın nesnesinin oluşturulması istenmediğinde.
    - Bu durumda hiç bir metodu soyut olmadan bir sınıf soyut olabilir.
    - Bu daha nadir görülen bir durumdur.
  - Daha yaygın halde, sınıftaki bazı metotlara gerçekleştirme verilmek istenmediğinde.
    - Bunun iki sebebi olabilir:
      - Ya baştan bazı metotlar için, tüm alt sınıflara uygun bir gerçekleştirme bulunamaz.
      - Ya da başta bazı metotlar için, tüm alt sınıflara uygun bir gerçekleştirme bulunduğu düşünülse bile bir müddet sonra tüm alt sınıfların bu metodu override ettiği görülür.

Java API'sindeki Soyut Sınıflar

- Java API'sinde pek çok soyut sınıf vardır;
  - java.io paketinde InputStream, OutputStream, Reader ve Writer
  - java.util paketinde Calendar ve AbstractList

Özet

- Bu bölümde, çok şekillilik (polymorphism) ele alındır.
- Tipler arasındaki çevrimler (conversion) ele alındı.

Ödevler

Ödevler I

- 2. Bölüm uygulamalarında kurguladığınız yandaki hiyerarşide Student sınıfını ve üzerindeki study() ve register() metotlarını abstract yapın.

- Bu metotları alt sınıflarda gerçekleştirin.