Java ile Nesne Merkezli Programlamaya Giriş

2. Bölüm
Nesne Merkezli Programlamanın Temelleri

Akın Kaldıroğlu

www.javaturk.org

Aralık 2016

Küçük Ama Önemli Bir Konu

- Bu dosya ve beraberindeki tüm, dosya, kod, vb. eğitim malzemelerinin tüm hakları **Selsoft Yazılım, Danışmanlık, Eğitim ve Tic. Ltd. Şti**.'ne aittir.
- Bu eğitim malzemelerini kişisel bilgilenme ve gelişiminiz amacıyla kullanabilirsiniz ve isteyenleri **http://www.selsoft.academy** 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.kaldiroglu@selsoft.academy** 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 bölümde şu konular ele alınacaktır:
  - Soyutlama,
  - Yazılım Mühendisliği ve Modelleri,
  - Sınıf ve Nesne kavramları,
  - Java'da Sınıf Tanımlama ve Kod Yapısı,
  - Bir Soyutlama Örneği.

Soyutlama

Soyutlama I

- **Soyutlama**, bir şeyin, sahip olunan bakış açısı itibariyle, en önemli özelliklerini ön plana çıkarırken, önemli olmayan özelliklerini bastırmaktır, görmezden gelmektir:
  - Önemli olan özellikler, genel olarak o şeyi diğer şeylerden ayırt eden unsurlardır ya da ana, asli özelliklerdir,
  - Ayırt edici olmayanlara ise ikincil ya da arizi özellikler denir.
- Zihnimiz karşılaştığı her nesneyi tek tek algılamak yerine, nesneleri, karekteristik özellikleriyle algılar, sonra da ya zihinde var olan kavramsal bir kategoriyle örtüştürür ya da böyle bir kategori yoksa, bu nesneden yola çıkarak yeni bir kategori oluşturur.

Soyutlama II

- **soyutlama** *İng.* **abstraction** (Lat. abstractio < abstrahere = çekip çıkarmak, soymak, ayırmak): 1. Gerçekte ayrılamaz olanı düşüncede ayırma eylemi. (Ör. Biçimi, rengi, boyutları özdekten ayırıp düşünme.) 2. Geneli ve öz olanı arınmış bir biçimde elde etmek için özle ilgili olmayanı bir yana bırakma. Bir tasarımın ya da bir kavramın nitelik ve bağıntı gibi öğelerini göz önüne almayarak dikkati doğrudan doğruya kavrama çeken düşünme eylemi. (Ör. Üçgen kavramında; üçgenin büyük, küçük, eşkenarlı ya da dik açılı oluşunu göz önüne almamak gibi.)
- Bilim ve Sanat Terimleri Sözlüğü, http://www.tdk.gov.tr/

Soyutlama III

- Soyutlamanın İngilizce karşılığı **abstraction** kelimesidir.
- Britannica'da şöyle tarif edilmektedir:

  The cognitive process of isolating, or "abstracting," a common feature or relationship observed in a number of things, or the product of such a process.

- Soyutlama, bu anlamda *kavramsallaştırma* ile aynı anlamdadır.

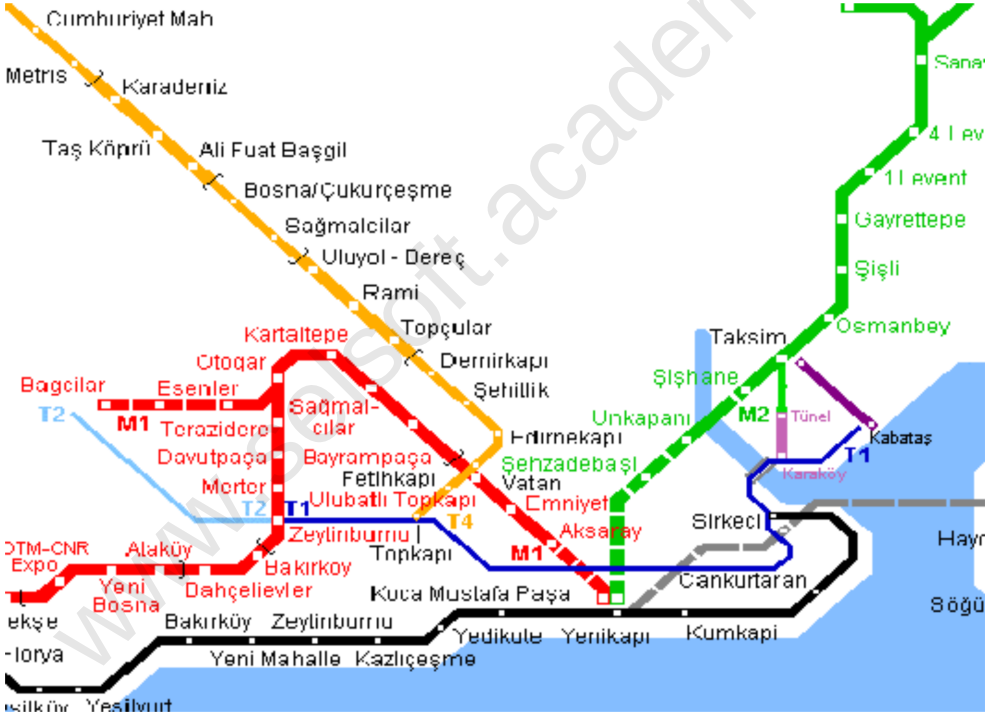
Neden Soyutlama?

- Çünkü zihnimiz bu haliyle soyutlama yapmadan bir nesneyi tüm yönleriyle kavrayamaz.
- Dolayısıyla soyutlama, bir indirgemedir, basitleştirmedir ya da genellemedir.
- Soyutlamalar, bir bebeğin dünyayı algılama şeklinde apaçık görülür:
  - Köpek: 4 ayaklı, havlayan ve kuyruğunu sallayan canlı

Soyutlama Örneği: Haritalar

- Haritalar, soyutlamaya mükemmel bir örnektirler:
  - Farklı türdeki haritalar, aynı gerçekliğin farklı yönlerini betimlerle amaçları dışındaki özellikleri göstermezler.
- Dolayısıyla zihin de soyutlamaları bir amaç doğrultusunda yapar.
- Amaç ya da ilgi alanı, aynı gerçekliğin farklı soyutlamalarının yapılmasını sağlar:
  - Otel zincirlerinin ya da benzin istasyonlarının haritaları,
  - Aynı binaya bakan farklı meslekteki kişilerin farklı soyutlamaları.

İstanbul Metro Haritası



"Ne"lik ve "Nasıl"lık

- Soyutlamalar, bir şeyin "ne"liği üzerine yoğunlaşır, "nasıl"lık üzerine eğilmez.

- Yani soyutlamalar, şeylerin temel özelliklerini, o özelliklerin oluşumundan, nasıl meydana geldiğinden bağımsız olarak ele alır:
  - Araba'yı algılamak için, motor yapısını bilmeye gerek yoktur,
  - Ya da bir insanla ortaklık kurmak için onun DNA dizilerini bilmeye de gerek yoktur.

Soyutlama ve Gerçeklik

- Soyutlama, sonsuz karmaşıklıktaki gerçekliği basitleştirerek anlama çabasıdır.
- Aslında bütün soyutlamalar birer kurgudur, insanın gerçekliğe giydirdiği bir kisvedir, çoğu gerçeklikte ya yoktur ya da bizim algıladığımız şekilde değildir:
  - Banka ya da banka hesabı ya da bilgisayarda dosya sistemi tamamen insan kurgusudur
    - Dosya sistemine karşın hard diskin fiziksel yapısı ve dosya kopyalamayı düşünün.
  - Metre ya da gram ise gerçeklikte bizim algıladığımız şekilde değildir.
- Dolayısıyla insan zihni, gerçekliği algılamak için olduğu kadar kendi gerçekliğini oluşturmak için soyutlamalar yapar.

Model I

- Bir şey ile ilgili farklı açılardan ya da ilgi alanlarından yapılan soyutlamaların bütününe **model** denir.
- Modeller, bir neseyi ya da olguyu daha bütünsel olarak ifade ederler, çünkü birden fazla soyutlama barındırırlar.
  - Bu nesne ve olguların çoğu zaman bir tek açıdan yapılan soyutlamayla anlaşılamayacak kadar farklı vechelere sahip olmasındandır.
- Dolayısıyla modellerle biz gerçeklikteki şeylerin, zihnimizin daha rahat algılaması amacıyla, birden fazla bakış açısıyla elde edilmiş tanımlarını yapmış oluruz .

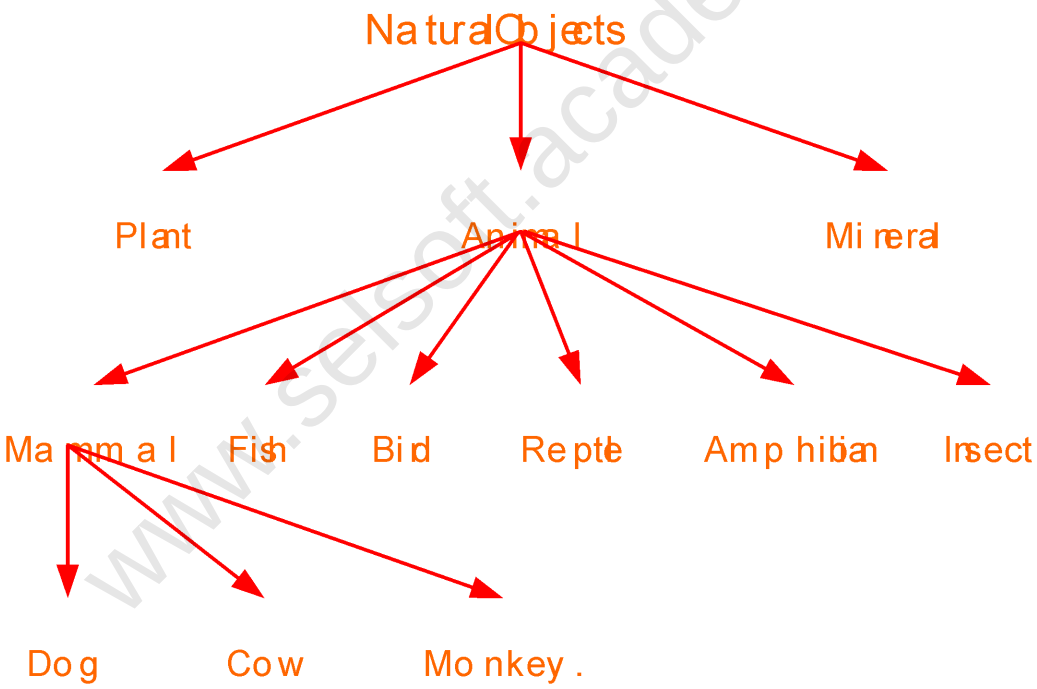
Model II

- Ya da başka bir deyişle soyutlamaları da sınıflandırırız:
  - Bir binanın statik, dinamik açılardan özelliklerini ve davranışlarını farklı soyutlamalarla ortaya koyarız.
  - Aynı binanın elektrik donanımı, ısıtma/soğutma/havalandırma düzeni ve ısı yalıtımı vs., hepsi birer soyutlamadır.
  - Bir binayı ancak bu soyutlamaların bütünü temsil edebilir ki bu durumda *binanın modeli* elde edilmiş olur.

Sınıflandırma

- Zihin, sonsuz sayıda soyutlama ile uğraşır,
- Ve bu soyutlamalarla elde edilen genellemeler birer kategori oluşturur,
- Soyutlamayı yönlendiren ilgi alanları ya da kriterler çerçevesinde kategori oluşturmaya **sınıflandırma** (**classification**) denir.
- Böylece nesneler, olgular, duygular vs. hepsi belli sınıflara ait hale gelirler.
  - Sınıflar, kavramsal genellemeleri, nesneler ise sınıfların gerçeklikt var olan örneklerini oluşturur.

Sınıflandırma Örneği



Sınıf ve Nesne

- Konuşmalarımızda "Köpek dediğin sadık olmalı." ya da "Köpekler çok sadık hayvanlardır." diyorsak, komşunun köpeğinden ya da sokakta az önce karşılaştığımız köpekten değil de kavramsal olarak *köpek sınıfından* bahsediyoruz demektir ve söylediklerimiz, var olmuş ve olacak bütün köpekler için geçerlidir.
- "Komşunun köpeği çok sadık." dediğimizde ise köpek sınıfının bir örneği ya da nesnesi olan, soyut olmayıp somut olan bir canlıdan bahsediyoruz demektir.
- İlk durumda "köpek" bir sınıfı, "komşunun köpeği" nde ise köpek bir nesneyi (object ya da instance) temsil eder.

Soyutlama ve Programlama

- Peki bütün bunların yazılım ve programlamayla ne alakası var?
- Yazılım geliştirme ve programlama da bir dizi soyutlamayı içerir.
- Programlama dilleri de birer soyutlamadır hatta modeldir:
  - Değişkenler ve veri tipleri
  - Operatörler ve kontrol yapıları
  - Fonksiyonlar
  - Sınıflar, modüller vs.

Yazılım Mühendisliği ve Modelleri

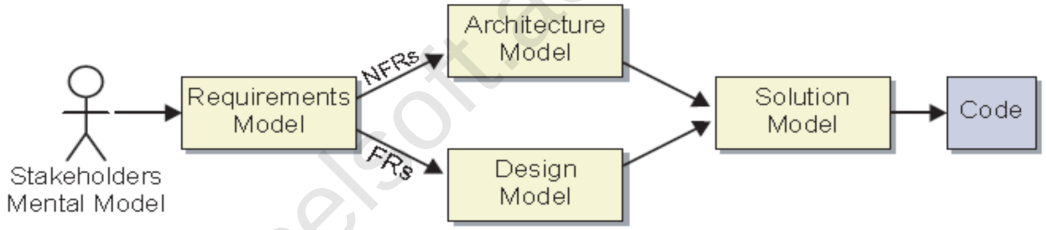
Yazılım Mühendisliği ve Modeller

- Çok basit olanlar dışında hemen hiç bir iş problemi, problemi ifade eden kişinin ağzından çıkar çıkmaz programlanıp, bir yazılım haline getirilemez.
- Dolayısıyla yazılım geliştirme sürecinde, ihtiyacın alınmasından gerçekleştirilmesine kadar geçen süreçte pek çok modelin kurgulaması, karmaşık yazılımları geliştirilebilir kılar.
- Ancak bu modeller yardımıyla karmaşık ihtiyaçları ve yazılmları basitleştirir ve anlaşılır hale getiririz.

Yazılım Mühendisliği Modelleri

- Yazlım geliştirme süreci en temelde üç modelden oluşur:
  - **Analiz** (**Analysis**): Bu ihtiyacın modelidir.
  - **Tasarım** (**Design**): Bu ihtiyacın çözümünün modelidir.
  - **Gerçekleştirme** (**Implementation**): Bu da çalışan modeldir.
- Analiz modeli, ihtiyacın **ne** olduğunu anlamaya yöneliktir.
- Tasarım modeli, ihtiyacın **nasıl** yerine getirileceğini ortaya koyar. Mimari tasarım da bu modelin içindedir.
- Gerçekleştirme de, analizde detaylandırılan ihtiyaçların tasarımda ortaya konan çözüm modelini bir programlama dili ile kodlanır.

Modeller



Modellemenin Zorlukları

- Sınırı yoktur, sonsuz sayıda ve derinlikte modeller oluşturulabilir.
  - Detay miktarı, amaç ve imkanlarla belirlenir.
- Mutlak doğru model ancak gerçekliğin kendisidir.
  - Doğruluk ya da yanlışlık, ancak gerçekliğe uygunlukla ölçülebilir,
  - Gerekli özellikleri alıp gereksizleri bırakmak, soyutlamanın dolayısıyla da modelin doğruluğunu belirler.
- Modellerin testini yapmak zordur.
  - En iyi test, yazılımın müşteriyi ne kadar memnun ettiğidir.

Başarılı Yazılım

- En iyi, başarılı yazılım, ihtiyaçlarını tam ve doğru olarak yerine getirendir.
- Bu sebeple doğru modeller için modellenen iş alanı çok iyi bilinmeli ve doğru soyutlamalar yapılmalıdır.
  - Bu bilgi, tecrübe, yaratıcılık ve yoğun ve sağlıklı bir iletişim gerektirir.
- Ayrıca yazılım geliştirmenin prensipleri, yazılım modelleri de iyi bilinmeli ve uygulanmalıdır.

Sınıf ve Nesne

Nesne (Object)

- **Nesne**, insan zihninin algıladığı herhangi bir kavramsal ya da fiziksel şeydir:
  - Öğrenciler, derslere devam ediyorlar.
  - Öğretmen, sınıfta öğrencileri dinliyor.
  - Dersler yarın başlıyor.
- Nesnelerin özellikleri vardır ve bu özellikler, nesnelerin **durumlarını** *(state)* ve **davranışlarını** *(behavior)* ifade eder:
  - Sarı boyalı sınıfta öğrenci şiir okuyor.
  - Kırmızı top suya yuvarlandı.

Sınıf I

- **Sınıf**, benzer nesnelerin kategorisidir.
- Sınıf, nesneler için bir kalıptır, şablondur, yani kendisinden üretilecek olan nesnelerin sahip olacağı özellikler ile davranışları tarif eder.
- Sınıf, kendisinden türetilecek olan nesnelerinin özelliklerini değişik tiplerde değişkenlerle (ya da bir başka deyişle veri yapılarıyla (data structures)), davranışlarını ise metotlarla (method) (fonksiyon (function)/prosedür (procedure)) ifade eder.
- Nesnenin özelliklerinin bütününe **durum** (**state**), metotların bütününe de **arayüz** (**interface**) denir.

Sınıf II

- Böylece, aynı sınıftan üretilen nesneler aynı tipte olurlar, yani:
  - Aynı özelliklere sahiptir ama özelliklerin değerleri değişebilir,
  - Aynı davranışlara sahiptir,
    - Davranışlar genelde duruma bağlı olduğundan, farklı durumdaki nesnele davranışları da farklı olur.

Yazılımın Nesnesi

- Yazılımın nesnesi ise gerçek dünyadaki, kavramsal ya da fiziksel, bir nesneyi temsil etmek üzere, onun özelliklerini ve davranışlarını ifade eden yapıdır:
  - Yazılımın nesnesi, temsil ettiği gerçek dünyadaki nesnenin durumunu, sınıfında tanımlanan değişkenlerle, davranışlarını da metotlarla yerine getirir.
  - Yani nesne, sınıfının ifade ettiği soyut yapının hayat bulmuş, gerçekleşmiş halidir.

Durum

- Nesnelerin durumu ile daha çok durağan (static), görülen ve hissedilen, özellikleri kastedilir ve programlama dillerinde farklı tiplerde bir grup değişken ile ifade edilir.
- Nesnenin durumunu oluşturan her bir ayrık bilgiye ise **özellik** (**attribute**) denir:
  - Öğrenci
    - No, isim, soy isim, doğum tarihi, cinsiyet, adres, bölüm, aldığı dersler, vs
  - Ders
    - No, isim, bölüm, veren kişi, kredi sayısı, vs.

Davranış

- Nesneler davranırlar, hareket ederler, belli işleri yerine getirirler.
- Yazılım nesnelerinin davranışlarına, yerine getirdiği **sorumluluk (responsibility)**, verdiği **hizmet (service)** ya da aldığı **mesaj** (**message**) olarak bakmak, işimizi kolaylaştırır:
  - Öğrenci
    - Kayıt olur
    - Ders alır
    - Sınava girer, vs.
  - Ders
    - Öğrencinin kaydolmasına/bırakmasına izin verir
    - Ön şart dersleri hakkında bilgi verir, vs.

Mesajlaşma

- Bir nesne şu 4 şekilden biri ile bir sorumluluk yerine getirir:
  - Nesne, kendi durumu hakkında bilgi verir,
  - Nesne, kendi durumunu değiştirir,
  - Nesne, bir işi ya da faaliyeti yerine getirir ve zorunlu olmamakla birlikte bu faaliyet sonucunda bize birşeyler geri verir ,
  - Nesne, bizim ondan istediğimizi, bir başka nesneye havale eder:
    - Havale edilen nesne bu sefer yukarıdaki 3 durumdan birisiyle istenileni yerine getirir.
- Nesnelerin birbirlerinden bir sorumluluk/hizmet yerine getirmesini istemeye **mesajlaşma** (**messaging**) denir.

Nesne-Merkezli Yazılım

- Nesne-merkezli yazılım sistemi ise, birbirleriyle mesajlaşan ve bu şekilde iş süreçlerini yerine getiren bir grup nesneden başka birşey değildir.

- Nesneler, yazılım sisteminin yerine getireceği sorumlulukları paylaşırlar öyle ki her bir nesne, temsil ettiği kavramla ilgili sorumlulukları yerine getirir.

Sınıf Tanımlama ve Kod Yapısı

Java'da Sınıf Tanımlama I

- Java'da sınıf tanımlamak için **class** anahtar kelimesi kullanılır:

```
<niteleyici>* class <SınıfİSmi>{

}
```

- Sınıfın tanımı, Java'da en geniş blok olan sınıf blokuyla yapılır.
  - Java'da bloklar daima "**{ }**" ile oluşturulur.
- Sınıfın bir ya da daha fazla **niteleyicisi** (**modifier**) olabilir.
- Sınıfın geçerli ve anlamlı bir ismi olmalıdır.

Java'da Sınıf Tanımlama II

- Zorunlu olmamakla birlikte sınıfın öğeleri, özellikler, kurucular ve metotlar olarak sıralanır.
  - Özellikler farklı tiplerde olan **nesne değişkenleridir** (**instance variables**).
  - Metotlar ise **nesne fonksiyonlarıdır** (**instance methods**).

```
<niteleyici>* class <SınıfİsmI>{
    <özellik>*
    <metot>*
}
```

Özellik Tanımı

- Sınıfta özellikle şöyle tanımlanır:

```
<niteleyici>* <tip> <değişkenİsmi> = [ilk değer]
```

- Değişkenler için bir grup niteleyici vardır,
- Java'da her değişkenin bir tipi olmak zorundadır,
- Değişkenlere geçerli ve anlamlı bir isim verilmelidir,
- Değişkene, isteğe bağlı olarak, ilk değer de verilebilir.

Metot Tanımı

- Sınıfta metot şöyle tanımlanır:

```
<niteleyici>* <dönüş tipi> <metotİsmi>(<Arguman>*){
    <ifade>*
}
```

- Zorunlu olmamakla birlikte metotlar da farklı niteleyiciler alır,
- Metotlar, hiç birisi zorunlu olmamakla birlikte,
  - dışarıdan argümanlar alır,
  - argümanları, ifadeleri ile işler yani bir iş mantığını yerine getirir
  - ve çağrıldıkları ortama bilgi geri döndürürler,
- Dolayısıyla metota geçilen ve metottan dönen bilgi, metotun çağrıldığı ortam ile metot arasındaki iletişimin öğelerindendir.

Java Kaynak Kodunun Yapısı

- Java'nın kaynak kodu "**java**" uzantılı dosyalarda tanımlanır.
- Sınıfın bütün öğeleri sınıf bloku içinde tanımlanır,
  - Sınıfın, sadece paket (**package**) bilgisi ile koduna dahil edeceği (**import**) diğer yapıların ifadeleri sınıf dışında tanımlanır.
- **public** sınıf niteleyicisi sınıfa her yerden erişilebileceğini ifade eder.
- Her public sınıf (arayüz ya da enum), *kendi ismini taşıyan* ve .java uzantısına sahip olan bir dosyada yer almalıdır.
  - Dolayısıyla her .java kaynak kodu dosyasında *en fazla bir tane* public sınıf olabilir.

```
<package cümlesi>

<import cümlesi>*


public class <MyClass>{
    ...
}

class <SınıfİsmiN>{
    ...
}
...
...
class <SınıfİsmiN>{
...
}
```

Kaynak kodun ismi: **MyClass.java**

Nesne Yaratma

- Java'da nesne yaratmak için **new** anahtar kelimesi kullanılır,
- new'den sonra da kurucu metot çağrısı gelmelidir,
- Yaratılan nesneyi, kendisine ulaşmakta kullanılacak olan bir referansa atamak gereklidir.
- Bunların hepsini bir satırda yapabiliriz:

```
Student studentObject = new Student();
```

- Yukarıdaki satırda, atamanın sağ tarafında oluşturulan nesne, sol tarafındaki *Student* tipinden olan *studentObject* isimli bir referansa atanmaktadır.

Araba

- Bir "Araba" soyutlaması yapın.
  - Soyutlamada bulunması gereken davranışlar nelerdir?
    - Gitmek
    - Durmak
    - Hızlanmak
    - vs.
  - Soyutlamada bulunması gereken durum bilgileri nelerdir?
    - Marka
    - Model
    - Yıl
    - Hız

```
public class Car {
    String make;
    String model;
    String year;
    int speed;
    int distance;

    public void go(int newDistance){
        distance += newDistance;
    }

    public void accelerate(int newSpeed){
        speed = newSpeed;
    }

    public void stop(){
        speed = 0;
    }

    public String getInfo(){
        return "Car Info: " + year + " " + make +...
    }
}
```

Main Metot

- Java'da pek çok sınıfınız olsa bile en az bir tanesi **main** isimli özel bir metota sahip olmalıdır.
- main metota sahip olan sınıf JVM'e geçilerek çalıştırılabilir.
  - main metot, sistemin çalışmaya başladığı yerdir.
  - Nerede tanımlandığının pek önemi yoktur.
- Dolayısıyla diğer sınıfların nesneleri bu metotta oluşturulur ve üzerindeki metotlar burada çağrılarak sistem çalışmaya başlar.
- main metotun *arayüzü (interface)* aşağıdaki gibidir:

```
public static void main(String[] args)
```

Test Sınıfı

- Şimdi de oluşturduğumuz sınıfın, *test* ya da *istemci (client)* sınıfını oluşturup, main metot içerisinde bir kaç tane araba nesnesi yaratın.
- Daha sonra bu nesnenin durumunu değiştirin ve üzerindeki metotları çağırın.
- Yaratılan nesne üzerindeki değişken ve metotlara "." notasyonuyla ulaşılır:

```
Car arabam = new Car();
int hiz = arabam.speed;
arabam.go(50);
```

```
public class CarTest {
    public static void main(String[] args) {
    Car car1 = new Car();
        car1.make = "Mercedes";
        car1.model = "C200";
        car1.year = "2016";
        car1.distance = 0;
        car1.speed = 0;

        String infoCar1 = car1.getInfo();
        System.out.println(infoCar1);

        car1.accelerate(80);
        car1.go(10);

        infoCar1 = car1.getInfo();
        System.out.println(infoCar1);

        car1.accelerate(120);
        car1.go(20);

        infoCar1 = car1.getInfo();
        System.out.println(infoCar1);
    }
}
```

Uygulama

- Car sınıfını kullanarak 10.000 KM'de, saatte 220 km/saat hızla giden 2016 model bir Porche Cayman'ı nasıl oluşturursunuz?

Diğer Soyutlamalar

- Aşağıdaki açılardan baktığınızda soyutlamanızda neler değişir?
  - Sürücü
    - Akaryakıt kullanımı, maksimum hız
  - Satıcı
    - Fiyatı
  - Araba tasarımcısı
    - Üstü açılabilir mi?
  - Tamirci
    - Motor özellikleri
  - Sigortacı
    - Nerede ve kim kullanıyor, hasarı var mı?

Özet

- Bu bölümde nesne merkezli programlamaya giriş yaptık.
- Bu amaçla,
  - Soyutlama kavramını,
  - "Ne"lik ve "nasıl"lık ayırımı,
  - Sınıf ve nesne kavramlarını,
  - Nesnenin durumu ve davranışlarını,
  - Model kavramını ve Yazılım Mühendisliği modellerini,
  - Java sınıf tanımı ve kaynak kodunun yapısını,
  - Ve örnek bir soyutlama ile soyutlamanın sınıf olarak gerçekleştirilmesini

  işledik.

Ödevler

Ödevler

- İki farklı açıdan oluşturulmuş iki farklı Driver (Sürücü) soyutlaması yapın.
  - Car örneğinde olduğu gibi iki ayrı soyutlama için iki ayrı sınıf oluşturup uygun isimler verin.
  - Soyutlamalar arasındaki farkları özellik ve davranış açısından kıyaslayın.
  - Sınıflarınızın, main metoda sahip test sınıflarını oluşturun, main metotta nesneleri oluşturup özelliklerine ulaşın ve metotlarını çağırın.
- Karmaşık sayı (complex number) soyutlamasını nasıl yapardınız?