

BLM5504 NESNEYE DAYALI KAVRAMLAR VE PROGRAMLAMA

Eylül 2016

Yrd.Doç.Dr. Yunus Emre SELÇUK

GENEL BİLGİLER

BAŞARIM DEĞERLENDİRME

- 1. Ara Sınav: %20, 11 Kasım 2016 Çarşamba
- 2. Ara Sınav: %20, 16 Aralık 2016 Çarşamba
- Ara sınav telafisi: 30 Aralık 2016 Çarşamba
- Final Sınavı: %40,
- Proje ödevi: %10,
- Kısa ödevler: %10
- Vize haftaları diğer derslerinizin durumlarına göre değiştirilebilir.
- Verilemeyen ödev olursa yüzdesi vizelere dağıtılır.

KAYNAKLAR:

- Java Programlama (Nesne Yönelimli):
 - Java How to Program, Deitel & Deitel, Prentice-Hall. (≥ 9th ed. Early Objects Version)
 - Core Java 2 Vol. I, Horstmann & Cornell, Prentice-Hall. (≥ 7th ed.)
- Java Programlama (Yapısal):
 - Algoritma Geliştirme ve Programlamaya Giriş, Fahri Vatansever, Seçkin Y.
- UML:
 - UML Distilled, 3rd ed. (2003), Martin Fowler, Addison-Wesley.

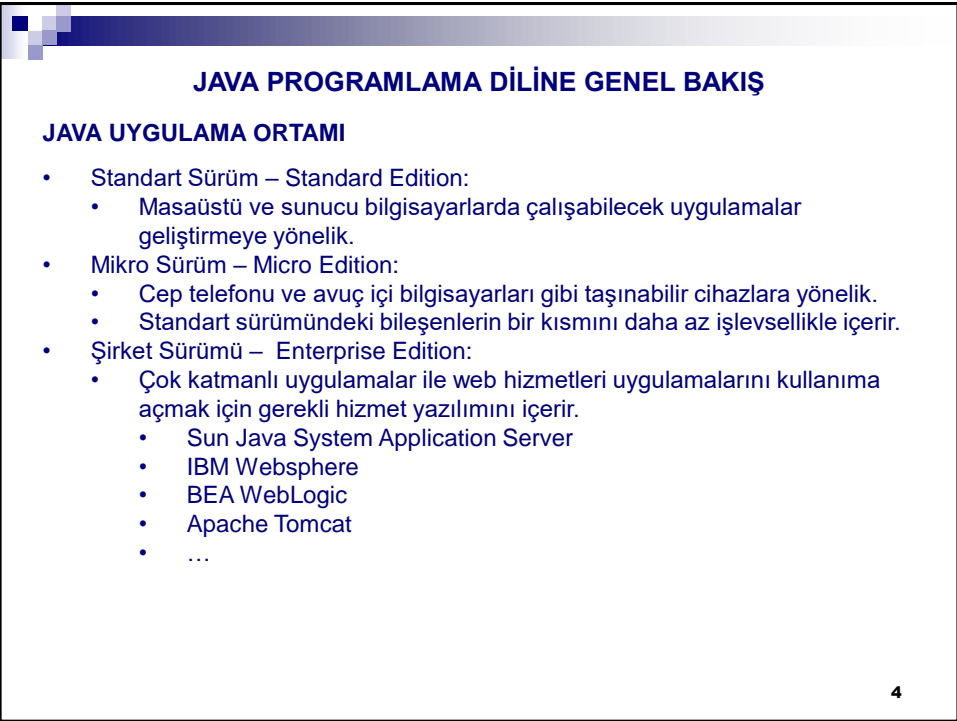
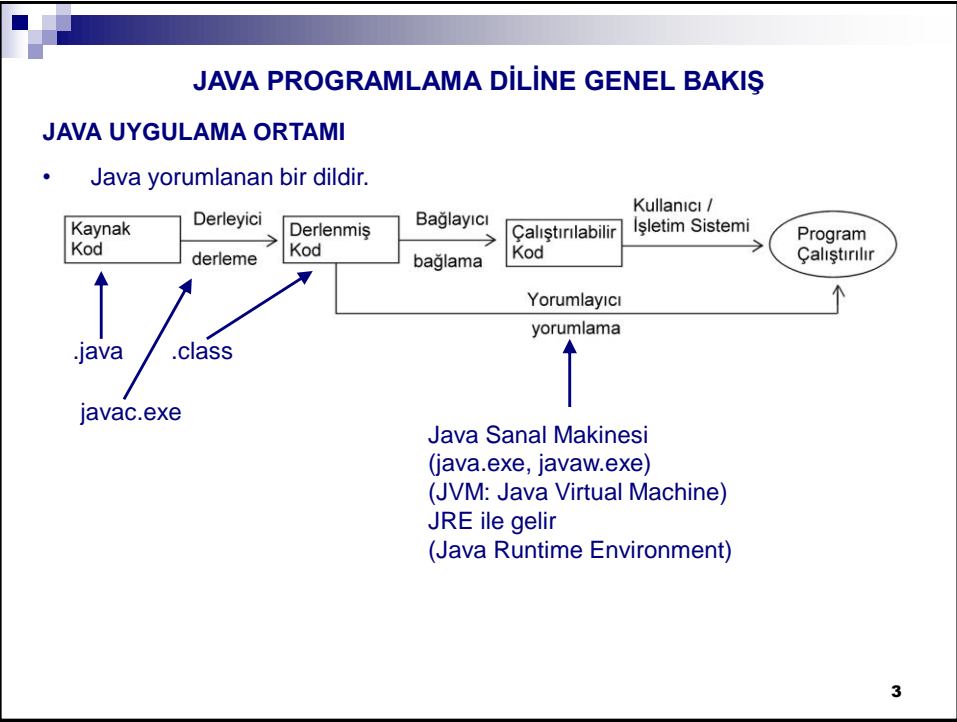
1

GENEL BİLGİLER

DERS İÇERİĞİ

- Temel içerik:
 - Java programlama diline genel bakış
 - Nesne ve Sınıf Kavramları
 - UML Sınıf Şemaları
 - Nesne Davranışı ve Metotlar
 - Nesne ve Sınıfların Etkileşimleri ve İlişkileri
 - UML Etkileşim (Sıralama) Şemaları
 - Kalıtım ve Soyut Sınıflar
 - Nesne Arayüzleri ve Çoklu Kalıtım
 - Çokbüçümlilik, Metotların Yeniden Tanımlanması ve Çoklu Tanımlanması

2



JAVA PROGRAMLAMA DİLİNE GENEL BAKIŞ

JAVA SÜRÜMLERİ

- Eski ve yeni sürümlendirme:

Eski Sürüm (Developer Version)	Yeni Sürüm (Product Version)
Java 1.0	
Java 1.1	
Java 1.2	Java 2 Platform
Java 1.3	Java 2 SE 3 (J2SE3)
Java 1.4	J2SE4
Java 1.5	J2SE5
Java 1.6 {Sun}	Java Platform Standard Edition, version 6 (JSE6)
Java 1.7 {Oracle}	Java Platform Standard Edition, version 7 (JSE7)
Java 1.8	Java Platform Standard Edition, version 8 (JSE8)


5

JAVA PROGRAMLAMA DİLİNE GENEL BAKIŞ

JAVA SÜRÜMLERİ

- Eski sürümlendirme ayrıntıları:
 - JDK 1.8.0.20:
 - Java 2, Version 8.0, update 20.
 - Update:
 - Hata düzeltme, daha iyi başarımlar ve güvenlik nedenleriyle güncellemeler.
 - Birkaç aylık aralıklarla.
- Nereden indirmeli?
 - Oracle.com/java
 - Dokümantasyonu da ayrıca indirip açınız.

6



JAVA PROGRAMLAMA DİLİNE GENEL BAKIŞ


ÜCRETSİZ JAVA GELİŞTİRME ORTAMLARI

- Eclipse: <http://www.eclipse.org>
 - Ayrıca indirilir.
 - UML için eUML2 plug-in'i kurulmalı.
 - GUI için ayrı plug-in kurulmalı.
 - Yönetici olarak kurulum gerekmiyor, unzip yetiyor.
- NetBeans:
 - JSE dağıtımı ile birlikte (seçimlik)
 - UML için ayrı plug-in gerek.
 - Kuran bana da isim söylesin.
 - Dahili GUI editörü var.
 - Yönetici olarak kurulum gerektiriyor.

ÜCRETSİZ UML MODELLEME ORTAMLARI

- Violet UML: Hafif sıklet, bizim için yeterli
- Argo UML

7



BLM5504 NESNEYE DAYALI KAVRAMLAR VE PROGRAMLAMA

Yrd. Doç. Dr. Yunus Emre SELÇUK

DERS NOTLARI:

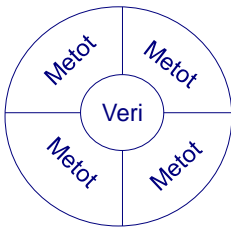
A. NESNEYE YÖNELİMİN TEMELLERİ

8

SINIFLAR, NESNELER VE ÜYELER

NESNE

- Nesne: Nitelikler ve tanımlı eylemler içeren, temel programlama birimi.
 - Nesne \approx bir gerçek dünya varlığına denk gelir.
 - Nesneler değişkenlere de benzetilebilir
 - Süpermen de sıradan insanlara benzetilebilir!
 - Nesnenin nitelikleri \approx Nesne ile ilgili veriler.
 - Eylemler \approx Nesnenin kendi verisi üzerinde(*) yapılan işlemler \approx Nesnenin kendi verileri ile çalışan metotlar (fonksiyonlar).
 - * Çeşitli kurallar çerçevesinde aksi belirtilmedikçe.
 - Metotlara parametre(ler) de verilebilir.
 - Sarma (Encapsulation): Veri ve eylemlerin birlikteliği.
 - Verilere, eylemler üzerinden erişilir.



9

SINIFLAR, NESNELER VE ÜYELER

SINIF

- Sınıf: Nesneleri tanımlayan şablonlar.
 - Şablon = Program kodu.

```
class myClass {  
    /*  
        program kodu  
    */  
}
```

10

SINIFLAR, NESNELER VE ÜYELER

NESNELER VE SINIFLAR

- Örnek nesne: Bir otomobil.
 - Nitelikler: Modeli, plaka numarası, rengi, vb.
 - Niteliklerden birinin tekil tanımlayıcı olması sorgulama işlerimizi kolaylaştıracaktır.
 - Eylemler: Hareket etmek, plaka numarasını öğrenmek, satmak, vb.
- Örnek sınıf: Taşıt aracı.
 - Nitelikleri ve eylemleri tanımlayan program kodu.
- Gerçek dünya benzetimi:
 - Nesne: Bir varlık olarak bir otomobil.
 - Sınıf: Dilbilgisi açısından bir genel isim olarak taşıt aracı.
- Bir nesneye yönelik program içerisinde, istenildiği zaman herhangi bir sınıftan olan bir nesne oluşturulabilir.
- Aynı anda aynı sınıftan birden fazla nesne etkin olabilir.

11

SINIFLAR, NESNELER VE ÜYELER

NESNELER VE SINIFLAR

- Bir nesnenin nitelikleri iki (!) çeşit olabilir:
 - Tamsayı, karakter gibi tek bir birim bilgi içeren 'ilkel' veriler,
 - Aynı veya başka sınıftan olan nesneler.
 - Sonsuz sayıda farklı sınıf oluşturulabileceği için, 'iki çeşit' deyimini çok da doğru değil aslında.
- Nesnenin niteliklerinin bir kısmı ilkel, bir kısmı da başka nesneler olabilir.

12

SINIFLAR, NESNELER VE ÜYELER

TERMİNOLOJİ VE GÖSTERİM

- NYP Terminolojisi:
 - Veri: Üye alan (Member field) = Nitelik (attribute)
 - Durum bilgisi: Nesnenin belli bir andaki niteliklerinin durumları
 - Eylem: Metot (Member method)
 - Nesnenin (Sınıfın) üyeleri = Üye alanlar + üye metotlar
 - Sınıf = tür = tip.
 - S sınıfından oluşturulan n nesnesi = n nesnesi S sınıfının bir örneğidir (instance)
- UML Gösterimi:



Sınıf: Sınıf şemasında



Nesne: Etkileşim şemasında

SINIFLAR, NESNELER VE ÜYELER

TERMİNOLOJİ VE GÖSTERİM

- İki tür UML etkileşim şeması (interaction diagram) vardır:
 1. Sıralama şeması (Sequence diagram)
 2. İşbirliği şeması (Collaboration diagrams)
- Bu derste sıralama şemaları çizeceğiz.
 - "Etkileşim" bu tür şemaların özünü çok iyi tarif ediyor. O yüzden "sıralama" ve "etkileşim" terimlerini birbirlerinin yerine kullanabilirim.

SINIFLAR, NESNELER VE ÜYELER

HER NESNE FARKLI BİR BİREYDİR!

- Aynı türden nesneler bile birbirinden farklıdır:
 - Aynı tür niteliklere sahip olsalar da, söz konusu nesnelerin nitelikleri birbirinden farklıdır = Durum bilgileri birbirinden farklıdır.
 - Durum bilgileri aynı olsa bile, bilgisayarın belleğinde bu iki nesne farklı nesneler olarak ele alınacaktır.
 - Bu farklılığı sağlamak üzere, her nesne programcının ulaşamadığı bir tekil tanımlayıcıya (UID: unique identifier) sahiptir.
 - Hiçbir nesnenin tanımlayıcısı birbiri ile aynı olmayacaktır.
 - UID'yi JVM kotarır.
- Örnek: Sokaktaki arabalar.
 - Örnek nitelikler: Modeli, rengi.
 - Modelleri ve renkleri farklı olacaktır.
 - Aynı renk ve model iki araba görseniz bile, plakaları farklı olacaktır.
 - Plaka sahteciliği sonucu aynı plakaya sahip olsalar bile, bu iki araba birbirlerinden farklı varlıklardır.

15

SINIFLAR, NESNELER VE ÜYELER

HER NESNE FARKLI BİR BİREYDİR! (devam)

- Aynı tür bile olsa, her nesnenin durum bilgisi farklı olduğu için, aynı türden iki nesne bile aynı mesaja farklı yanıt verebilir.
 - Örnek: Bana adımı sorsanız "Yunus" derim, sizin aranızda kaç Yunus var?
 - Kaldı ki, nesneye mesaj gönderirken farklı parametreler de verebilirsiniz. Aynı nesneye aynı mesaj farklı parametre ile giderse, geri dönen yanıtlar da farklı olacaktır.
- Terminoloji: Aynı türden iki nesne, aynı mesaja farklı yanıtlar verir.

16

SINIFLAR, NESNELER VE ÜYELER

NESNELERE MESAJ GÖNDERME

- Bir nesneye neden mesaj gönderilir?
 - Ona bir iş yaptırmak için
 - Bir üyesine erişmek için.

ÜYELERE ERİŞİM

- Üye alana erişim:
 - Üyenin değerini değiştirmek (setting)
 - Üyenin değerini okumak (getting)
 - (Değiştirmeden herhangi bir işlemde kullanmak)
- Üye metoda erişim:
 - Bir eylemler sürecini varsa kendine özgü çalışma parametreleri ile yürütmek.
 - Fonksiyon çağırmak gibi, ama unutmayın: Aksi belirtilmedikçe metod, üyesi olduğu nesnenin üyeleri ile çalışır.
 - Aksinin nasıl belirtileceğini ileride nesneler arasındaki ilişkileri öğrenince göreceksiniz.

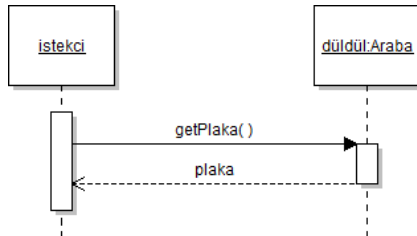
17

SINIFLAR, NESNELER VE ÜYELER

TERMİNOLOJİ VE GÖSTERİM

- NYP Terminolojisi:
 - Bir nesnenin diğer bir nesnenin bir üyesine erişmesi, bir nesnenin diğerine bir mesaj göndermesi olarak da tanımlanır.
 - Bir nesneye yönelik program, nesneler arasındaki mesaj akışları şeklinde yürür.

UML Gösterimi:



Kod Gösterimi:

```
döldul.getPlaka();  
//Nesne adını Türkçe veremiyoruz.
```

Anlamı:

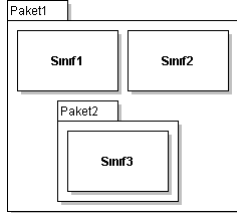
- istekçi adlı bir nesne vardır.
- istekçi nesnenin sınıfı belli değil.
- döldül adlı bir nesne vardır.
- döldül nesnesinin sınıfı Araba'dır.
- Araba sınıfının plakaniVer adlı bir metodu vardır.
- istekçi nesne döldül nesnesine plakaniVer mesajı gönderir.
- döldül nesnesi bu mesaja yanıt olarak kendi plakasını döndürür.

18

SINIFLAR, NESNELER VE ÜYELER

PAKETLER

- Sınıflar paket (package) adı verilen mantıksal kümelere ayrılabilir.
- Amaç: Kendi içerisinde anlam bütünlüğü olan ve belli bir amaca yönelik olarak birlikte kullanılabilecek sınıfları bir araya toplamaktır.



- Bir paketteki sınıfları koda ekleme:

```
import paket1.Sınıf1;
import paket1.*;
import paket1.Paket2.*;
```

- paket1 eklenince alt paketi olan paket2 içindeki sınıflar eklenmiş olmaz.
- Paketler, aynı adlı sınıfların birbirine karışmamasını da önler:
 - Sınıf adı aynı bile olsa farklı paketlerde bulunan sınıflar, belirsizlik oluşturmaz.
`java.io.File`
`com.fileWizard.File`

- Paket hiyerarşisi, aynı zamanda dosya hiyerarşisidir.

```
com.fileWizard.File -> com\fileWizard\File.java
```

19

SINIFLAR, NESNELER VE ÜYELER

GÖRÜNEBİLİRLİK KURALLARI VE VERİ GİZLEME

- Bir nesne, kendi sınıfından olan diğer nesnelerin ve bizzat kendisinin bütün üyelerine erişebilir,
- Ancak bir nesnenin üyelerine diğer sınıflardan olan nesnelerin erişmesi engellenebilir.
- Veri Gizleme İlkesi (information hiding):
 - İlke olarak, bir sınıfın içsel çalışması ile ilgili üyeler diğerlerinden gizlenir.
 - Böylece bir nesne diğerini kullanmak için, kullanacağı nesnenin ait olduğu sınıfın iç yapısını bilmek zorunda kalmaz.
- Örnek: TV çalıştırmak için uzaktan kumandalardaki ses ayarı, kanal değiştirme ve güç düğmelerinin evrensel işaretlerini tanımak yeterlidir;
 - Televizyonun içinde katot tüpü adlı bir cihaz olduğunu bilmek gerekmez.
 - Böylece LCD, plazma gibi yeni teknolojiler kullanıcıyı yeniden eğitmeye gerek kalmadan televizyonlarda kullanılabilir.
- Örnek: Arkadaşınız sizden borç para istedi.
 - Borç verirsiniz ya da vermezsiniz.
 - Arkadaşınızın sizin aylık gelirinizi, ATM kartınızın şifresini, vb. bilmesi gerekmez.

20

SINIFLAR, NESNELER VE ÜYELER

GÖRÜNEBİLİRLİK KURALLARI VE VERİ GİZLEME

- Genel Görünebilirlik Kuralları (Visibility rules):
 - Public: Bu tip üyelere erişimde hiç bir kısıtlama yoktur.
 - Private: Bu tip üyelere başka sınıflardan nesneler erişemez, yalnız kendisi ve aynı türden olan diğer nesneler erişebilir.
- UML Gösterimi:

ClassName
- aPrivateField : TypeOfField
+ aPublicVoidMethod()
+ aPublicMethod() : ReturnType
+ aMethodWithOneParameter(param1 : Param1Type)
+ manyParameteredMethod(param1 : P1Type, param2 : P2Type)

- Ayrıca (derste sorumlu değilsiniz):
 - protected: #
 - Kalıtım ile ilgili (paketteki diğer sınıflara ve alt sınıflarına açıktır)
 - package: ~
 - Paketteki diğer sınıflara açıktır
 - Java'da varsayılan kural

21

SINIFLAR, NESNELER VE ÜYELER

GÖRÜNEBİLİRLİK KURALLARI VE VERİ GİZLEME

- Veri gizleme ilkesi her zaman için mükemmel olarak uygulanamaz.
 - Bir sınıftaki değişiklik sadece o sınıfı etkilemekle kalmaz, ilişkide bulunduğu başka sınıfları da etkileyebilir.
 - Veri gizleme ilkesine ne kadar sıkı uyulursa, değişikliğin diğer sınıflara yayılması olasılığı veya değişiklikten etkilenen sınıf sayısı da o kadar azalır.
- Veri gizleme ilkesine uyulmasını sağlamak için:
 - Üye alanlar private olarak tanımlanır, ve:
 - Değer atayıcı ve değer okuyucu metotlar kullanılır.
 - Bu ilkeye uymazsanız gitti en az 5 puan!
- Değer atayıcı ve değer okuyucu metotlar (erişim metotları):
 - Değer atayıcı (Setter) metot: Bir nesnenin bir üye alanına değer atamaya yarar.
 - Değer okuyucu (Getter) metot: Bir nesnenin bir üye alanının değerini öğrenmeye yarayan metot.
 - Adlandırma: getUye, setUye

22

SINIFLAR, NESNELER VE ÜYELER

GÖRÜNEBİLİRLİK KURALLARI VE VERİ GİZLEME

- Örnek:

Araba
- plaka: String
+ getPlaka(): String
+ setPlaka(String)

- Üyelere erişim kurallarında istenen değişiklikler kolaylıkla yerine getirilebilir.
 - Örneğin plaka içeriğinin okunması serbest olmakla birlikte bu alana değer atanmasının sadece ilgili paket içerisindeki sınıflar tarafından yapılması gerektiğinde, getPlaka metodu public olarak bırakılıp setPlaka metodu paket düzeyi görünebilirliğe alınır


23

SINIFLAR, NESNELER VE ÜYELER

ÜYELERİN ÖZEL DURUMLARI

- Statik üye alanlar:
 - Aynı türden nesnelerin bile durum bilgisi farklıdır (gördük), ancak:
 - Kimi zaman aynı tipten tüm nesnelerin ortak bir üye alanı paylaşması istenilebilir.
 - Bu durumda üye alan static olarak tanımlanır.
 - SınıfAdı.üyeAdı** şeklinde sınıf üzerinden kullanılırlar, nesneler üzerinden kullanılmazlar.
 - Örnek: Her binek otomobilin 4 tekerleği vardır.
- Statik üye metotlar:
 - Aynı türden iki nesne, aynı mesaja farklı yanıtlar verir (gördük), ancak:
 - Kimi zaman aynı tipten tüm nesnelerin aynı mesajın aynı şekilde çalışması istenilebilir.
 - Bu durumda üye metot static olarak tanımlanır.
 - Statik metot içerisinde yalnız statik üyeler kullanılabilir.
 - Statik üye alana erişim metotları da statik tanımlanır.
 - SınıfAdı.üyeAdı()** şeklinde kullanılırlar.

24



SINIFLAR, NESNELER VE ÜYELER


ÜYELERİN ÖZEL DURUMLARI

- Final üye alanlar:
 - Bir alanın değerinin sürekli olarak aynı kalması istenebilir.
 - Bu durumda üye alan **final** olarak tanımlanır.
 - Final üyelere yalnız bir kez değer atanabilir.
 - Örnek: Bir arabanın şasi numarası o araba fabrikadan çıkar çıkmaz verilir ve bir daha değiştirilemez.
- Final üye metotlar:
 - Sınırlı kullanım alanı: Kalıtım ile aktarılamazlar (ileride).

DİKKAT EDİLECEK NOKTALAR

- Bir üye, hem final hem de static olabilir.
- Tanımları ve adları gereği final ve static kavramları birbiriyle karıştırılabilir:
 - Final: Bir kez değer atama
 - Static: Ortak kullanım. Sözlük karşılığı durağan, ancak siz ortak diye düşünün.

25



SINIFLAR, NESNELER VE ÜYELER

KURUCULAR VE SONLANDIRICILAR

- Kurucu Metot (Constructor):
 - Bir nesne oluşturulacağı zaman sınıfın kurucu adı verilen metodu çalıştırılır.
 - Nesnenin üyelerine ilk değerlerinin atanmasına yarar.
 - Bu yüzden ilklendirici metot olarak da adlandırılırlar.
 - Kurucu metotlara bu derste özel önem gösterilecektir.
- Sonlandırıcı metot:
 - Nesne yok edildiğinde JVM tarafından çalıştırılır.
 - Adı finalize'dir, parametre almaz, geri değer döndürmez.
 - C/C++ aksine, Java programcısının bellek yönetimi ile uğraşmasına gerek yoktur.
 - JVM için ayrılan bellek azalmaya başlamadıkça nesneler yok edilmez.
 - Bu yüzden bir nesnenin finalize metodunu çalıştırmak için çok çabalamanız gerekiyor!
 - Özetle:
 - Bu derste bu konu üzerinde daha fazla durulmayacaktır.

26

SINIFLAR, NESNELER VE ÜYELER

KURUCULAR

- Kurucu Metot kuralları:
 - Public görünürlüğe sahip olmalıdır.
 - Kurucu metodun adı, sınıfın adı ile aynı olmalıdır.
 - Bir kurucu metodun geriye o sınıftan bir nesne döndürmesine rağmen,
 - Metot imzasında bir geri dönüş tipi belirtilmez,
 - Metot gövdesinde bir sonuç geri döndürme (return) komutu bulunmaz.
 - Final üyelere değer atamak için uygun bir yerdir.
 - Alternatif: Final üyeye tanımlandığı yerde değer atanması
 - Kod içerisinde bir nesne oluşturulacağı zaman ise, kurucu metod **new** anahtar kelimesi ile birlikte kullanılır.

```
arabam = new Araba();
```

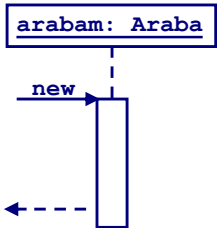
27

SINIFLAR, NESNELER VE ÜYELER

KURUCULAR

- Bir nesneyi kullanmak için onu tanımlamak yetmez, kurucusunu da çalıştırmak suretiyle onu ilklendirmek (*initialize, instantiate*) gerekir.

- UML Gösterimi:



- Kod gösterimi 1: Üye alan olarak kullanım

```
public class AClass {
    private Araba arabam;

    someMethod( ) {
        arabam = new Araba();
    }
}
```

- Kod gösterimi 2: Geçici değişken olarak kullanım

```
public class AnotherClass {
    someMethod( ) {
        Araba arabam = new Araba();
    }
}
```

28

SINIFLAR, NESNELER VE ÜYELER

KURUCULAR

- Varsayılan kurucu (default constructor):
 - Parametre almayan kurucudur.
 - Programcı tanımlamazsa, JVM (C++: Derleyici) tanımlar.
- Parametrelili kurucular:
 - Üye alanlara parametreler ile alınan ilk değerleri atamak için kullanılır.
 - Bir tane bile parametrelili kurucu tanımlanırsa, buna rağmen varsayılan kurucu tanımlanmamışsa, varsayılan kurucu kullanılamaz.
- Bir sınıfta birden fazla kurucu olabilir, ancak varsayılan kurucu bir tanedir.
 - Aynı üye aynı sınıf içinde birden fazla tanımlanamaz.
 - Aynı adı paylaşan ancak imzaları farklı olan birden fazla metot tanımlanabilir.
 - Bu tür metotlara **adaş metotlar**, bu yapılan işe ise adaş metot tanımlama (**overloading**) denir.

29

BİR NESNEYE DAYALI PROGRAMIN OLUŞTURULMASI

DENETİM AKIŞI

- Denetim akışı: Kodların yürütüldüğü sıra.
 - En alt düzeyde ele alındığı zaman bir bilgisayar programı, çeşitli komutların belli bir sıra ile yürütülmesinden oluşur.
 - Komutların peş peşe çalışması bir nehrin akışına benzetilebilir.
 - Komutların kod içerisinde verilmiş sırası ile bu komutların yürütüldüğü sıra aynı olmayabilir.
 - Belli bir komut yürütülmeye başlandığı zaman ise o komut için denetimi ele almış denilebilir.
 - Bu benzetmelerden yola çıkarak, kodların yürütüldüğü sıraya denetim akışı adı verilebilir.

30

BİR NESNEYE DAYALI PROGRAMIN OLUŞTURULMASI

DENETİM AKIŞININ BAŞLANGICI

- Denetim akışının bir başlangıcının olması gereklidir.
 - Akışın hangi sınıftan başlatılacağını programcı belirler.
 - Java'da denetim akışının başlangıcı: Main komutu.
 - `public static void main(String[] args)`
 - `static`: Henüz bir nesne türetilmedi!
 - `args` dizisi: Programa komut satırından ilk parametreleri aktarmak için
 - Main metodunun görevi, gerekli ilk bir/birkaç nesneyi oluşturup programın çalışmasını başlatmaktır.
 - Hatırlayın, bir nesneye yönelik programın nesneler arasındaki mesajlar ile yürüdüğünü söylemiştik.
 - Bir sınıfın main metodunun olması, her zaman o metodun çalışacağı anlamına gelmez.
- Blok: Birden fazla komut içeren kod parçası.
 - Kıvrık parantez çifti içerisinde: { ve }

31

BİR NESNEYE DAYALI PROGRAMIN OLUŞTURULMASI

KENDİ SINIFLARINIZI OLUŞTURMAK VE KENDİ NESNELERİNİZİ TÜRETMEK

- UML gösterimi (sınıf şeması)

Araba
- plaka : String
+ Araba(plakaNo : String)
+ getPlaka() : String
+ setPlaka(String)
+ kendiniTanit()
+ main(String[])

- Önce UML sınıf şemasını çiz.
- Sonra kodda neresinin şemada nereye denk geldiğini işaretle.
- Pretty printing, camel casing ...

- Kaynak kod (gerçekleme)

```
package ndk01;
public class Araba {
    private String plaka;

    public Araba( String plakaNo ) {
        plaka = plakaNo;
    }
    public String getPlaka( ) {
        return plaka;
    }
    public void setPlaka( String plaka ) {
        this.plaka = plaka;
    }
    public void kendiniTanit( ) {
        System.out.println( "Plakam: " + getPlaka() );
    }

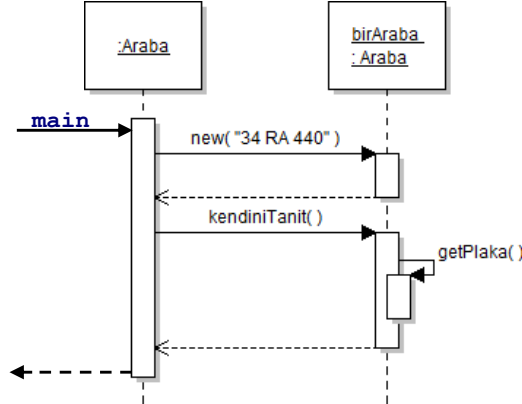
    public static void main( String[] args ) {
        Araba birAraba;
        birAraba = new Araba( "34 RA 440" );
        birAraba.kendiniTanit( );
    }
}
```

32

BİR NESNEYE DAYALI PROGRAMIN OLUŞTURULMASI

UML ŞEMASI İLE GÖSTERİM

- Örnek koddaki main metodunun, Etkileşim (Interaction) şeması türü olan sıralama şeması (sequence) ile gösterimi.
 - Okların düşeydeki sıralamasına azami dikkat ediniz !



33

BİR NESNEYE DAYALI PROGRAMIN OLUŞTURULMASI

KENDİ SINIFLARINIZI OLUŞTURMAK VE KENDİ NESNELERİNİZİ TÜRETMEK

- Araba sınıfının başka bir versiyonu:

Car
- plate : String
- chassisNR : String
+ Car(String, String)
+ getPlate() : String
+ setPlate(String)
+ getChassisNR() : String

```
package ndk01;
public class Car {
    private String plate;
    private String chassisNR;
    public Car( String plateNr, String chassisNR ) {
        plate = plateNr;
        this.chassisNR = chassisNR;
    }
    public String getPlate() {
        return plate;
    }
    public void setPlate(String plate) {
        this.plate = plate;
    }
    public String getChassisNR( ) {
        return chassisNR;
    }
}
```

- Araba sınıfının bu versiyonunda bir main metodu yoktur. Bu nedenle doğrudan çalıştırılıp sıranamaz.
- Bu amaçla main metoduna sahip başka bir sınıf kodlamalı ve Car sınıfını oradan test etmeliyiz (ileride gösterilecek).

34

TEMEL VERİ TEMSİLİ VE İŞLEMLERİ

İLKEL VERİ TİPLERİ

- İlkeller ile işlemler:
 - Aritmetik: + - * / %.
 - İşlem önceliği
 - ++, --, (Bir arttırma ve bir azaltma)
 - ++i ile i++ farkı
 - Atama ve işlem: += -= *= /= %=
 - Anlaşılabilirlik için işi sade tutun, abartmayın
 - Mantıksal: & | ! vb.
- Özetle, önceki programlama derslerinizden öğrendiğiniz gibi.

37

TEMEL VERİ TEMSİLİ VE İŞLEMLERİ

STRING SINIFI

- String sınıfı metotları
 - int length()
 - int compareTo(String anotherString)
 - int compareToIgnoreCase(String str)
- System.out.println(String)
 - print / println
- Örnek:

```
package ndk01;
public class StringOps01 {
    public static void main( String args[] ) {
        String strA, strB;
        strA = "A string!";
        strB = "This is another one.";
        System.out.println(strA.compareTo(strB));
    }
}
```

- Örneğin çıktısı: -19

38

TEMEL VERİ TEMSİLİ VE İŞLEMLERİ

STRING SINIFI (devam)

- String sınıfı metotları (devam)
 - boolean contains(String anotherString)
 - String toUpperCase()
 - String toLowerCase()
 - Dikkat: toUpper/LowerCase metotları çağırılan nesnenin kendisini değiştirmiyor.
 - Bu bilgi doğrultusunda, sonraki yansıdaki kodun çıktısı acaba ne olacaktır?

39

TEMEL VERİ TEMSİLİ VE İŞLEMLERİ

STRING SINIFI (devam)

```
package ndk01;
public class StringOps02 {
    public static void main( String args[] ) {
        String strA = "İstanbul", strB = "Yıldız";
        System.out.println(strA.contains(strB));
        strB = "tan";
        System.out.println(strA.contains(strB));
        strB.toUpperCase();
        System.out.println(strB);
        System.out.println(strA.contains(strB));
        strB = strB.toUpperCase();
        System.out.println(strB);
        System.out.println(strA.contains(strB));
    }
}
```

- Örneğin çıktısı: ???

40

TEMEL VERİ TEMSİLİ VE İŞLEMLERİ

KOMUT SATIRI ÜZERİNDEN G/Ç

- System.out nesnesi ile çıktı işlemleri:
 - System sınıfının out üyesi public ve statiktir
 - Bu yüzden out nesnesi doğrudan kullanılabilir.
 - Komut satırına çıktı almak için metotlar:
 - println, print: Gördük
 - printf: C kullanıcılarının alıştığı şekilde kullanım.

41

TEMEL VERİ TEMSİLİ VE İŞLEMLERİ

KOMUT SATIRI ÜZERİNDEN G/Ç

- java.util.Scanner sınıfı ile giriş işlemleri: JDK 5.0 ile!
 - Oluşturma: Scanner in = new Scanner(System.in);
 - System.in : java.io.InputStream türünden public static üye.
 - Tek tek bilgi girişi için metotlar:
 - String nextLine()
 - int nextInt()
 - float nextFloat()
 - ...

```
package ndk01;
import java.util.Scanner;
public class ConsoleIOv1 {
    public static void main(String[] args) {
        Scanner in = new Scanner(System.in);
        System.out.print("What is your name? ");
        String name = in.nextLine();
        System.out.print("How old are you? ");
        int age = in.nextInt();
        System.out.println("Hello, " + name +
            ". Next year, you'll be " + (age + 1) + ".");
    }
}
```

42

TEMEL VERİ TEMSİLİ VE İŞLEMLERİ

KOMUT SATIRI ÜZERİNDEN G/Ç

- Scanner sınıfındaki bir hata:
 - nextInt, nextFloat, vb. ilkel okuma komutundan sonra bir karakter katarı okumak için nextLine kullanırsan sorun çıkıyor.
 - Çözmek için ilkel okumadan sonra bir boş nextLine ver.

```
package ndk01;
import java.util.Scanner;
public class ConsoleIOv2 {
    public static void main(String[] args) {
        Scanner in = new Scanner(System.in);
        System.out.print("How old are you? ");
        int age = in.nextInt();
        in.nextLine(); //workaround for the bug
        System.out.print("What is your name? ");
        String name = in.nextLine();
        System.out.println("Hello, " + name +
            ". Next year, you'll be " + (age + 1) + ".");
    }
}
```

43

TEMEL VERİ TEMSİLİ VE İŞLEMLERİ

KOMUT SATIRI ÜZERİNDEN G/Ç

- Araba sınıfının main metodunu, araç plakasını kullanıcıdan alacak şekilde değiştirelim:

Araba
- plaka : String
+ Araba(plakaNo : String)
+ getPlaka() : String
+ setPlaka(String)
+ kendiniTanit()
+ main(String[])

```
package ndk01;
import java.util.Scanner;
public class ArabaV2 {
    private String plaka;
    public Araba( String plakaNo ) {
        plaka = plakaNo;
    }
    public String getPlaka( ) {
        return plaka;
    }
    public void setPlaka( String plaka ) {
        this.plaka = plaka;
    }
    public void kendiniTanit( ) {
        System.out.println( "Plakam: " + getPlaka() );
    }

    public static void main( String[] args ) {
        Araba birAraba;
        Scanner input = new Scanner( System.in );
        System.out.print("Enter a license plate: ");
        String plakaNr = input.nextLine();
        birAraba = new Araba( plakaNr );
        birAraba.kendiniTanit();
    }
}
```

44

DENETİM AKIŞI

- Yapısal programlamadan bildiğiniz kurallar Java dili için de aynen geçerlidir, çünkü nesneye yönelim yapısal programlamanın üst kümesidir.
 - Yapısal programlamada eksik olanlar için notların sonraki bölümü eklenmiştir.
 - Ek alıştırmalar için bkz. Fahri Vatansever, “Algoritma Geliştirme ve Programlamaya Giriş”, Seçkin Yayıncılık.

KARAR VERME İŞLEMLERİ – IF DEYİMİ

```
if (koşul) {...} else if (koşul) {...} ... else (koşul) {...}
```

- Koşul kısmı hakkında:
 - Karşılaştırma: < > <= >= == !=
 - Mantıksal işlemlerde çift işleç kullanılır: && ||

DÖNGÜLER

```
for( baslangicIfadesi; devamIfadesi; artimIfadesi ) { ... }  
while( kosul ) { ... }  
do { ... } while( kosul );  
switch / case ...
```

45

DENETİM AKIŞI

KARAR VERME İŞLEMLERİ – SWITCH DEYİMİ

- Anahtar tamsayı, tek karakter veya enum olabilir:

```
public class SwitchCase {  
    enum Yon { YUKARI, ASAGI, SAG, SOL };  
    void deneme( ) {
```

```
        switch (anahtar1) {  
            case 1:  
                break;  
            default:  
                break;  
        }
```

```
        switch (anahtar2) {  
            case YUKARI:  
                break;  
            default:  
                break;  
        }
```

```
        switch (anahtar3) {  
            case 'Y':  
                break;  
            default:  
                break;  
        }
```

```
    }  
}
```

46