Temel Kavramlar

BBS-515 Nesneye Yönelik Programlama

Ders #2 (2 Kasım 2009)

İçerik

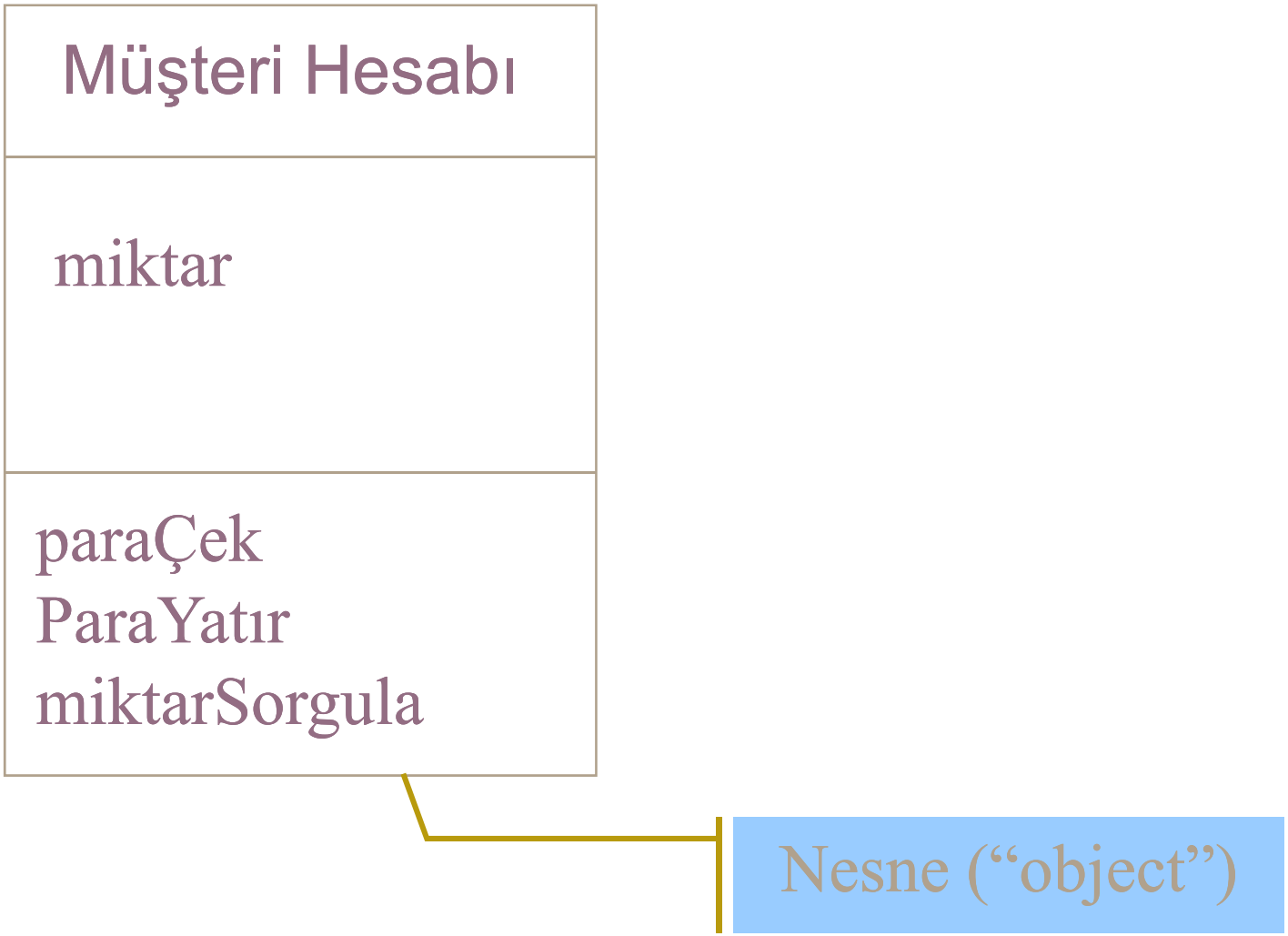
- Geçen hafta:
  - Nesneye yönelik programlama
    - Ne demektir, nasıl ortaya çıkmıştır?
    - Nesneye yönelik dil olarak JAVA ve ilişkili kavramlar
- Bu hafta:
  - “Nesne” ve “Sınıf” Kavramları
  - “Nesne” ve “Sınıf”ın Java’da gerçekleştirilmesi

“Nesne” ve “Sınıf” Kavramları

“Nesne” Nedir? – 1

- İyi tanımlı bir kapsamı ve kimliği olan, belirli bir durum ve davranışı içeren, soyut veya somut varlıktır.
  - *“A discrete entity with a well-defined boundary and identity that encapsulates state and behavior”*
  - Nesne gerçek dünyadaki somut bir varlığı temsil edebilir.
    - Televizyon, motor, vb.
  - Nesne tamamen kavramsal bir varlığı temsil edebilir.
    - Banka hesabı, vb.

“Nesne”: Örnek


Müşteri Hesabı
miktar
paraÇek
ParaYatır
miktarSorgula
Nesne (“object”)

“Nesne” Nedir? – 2

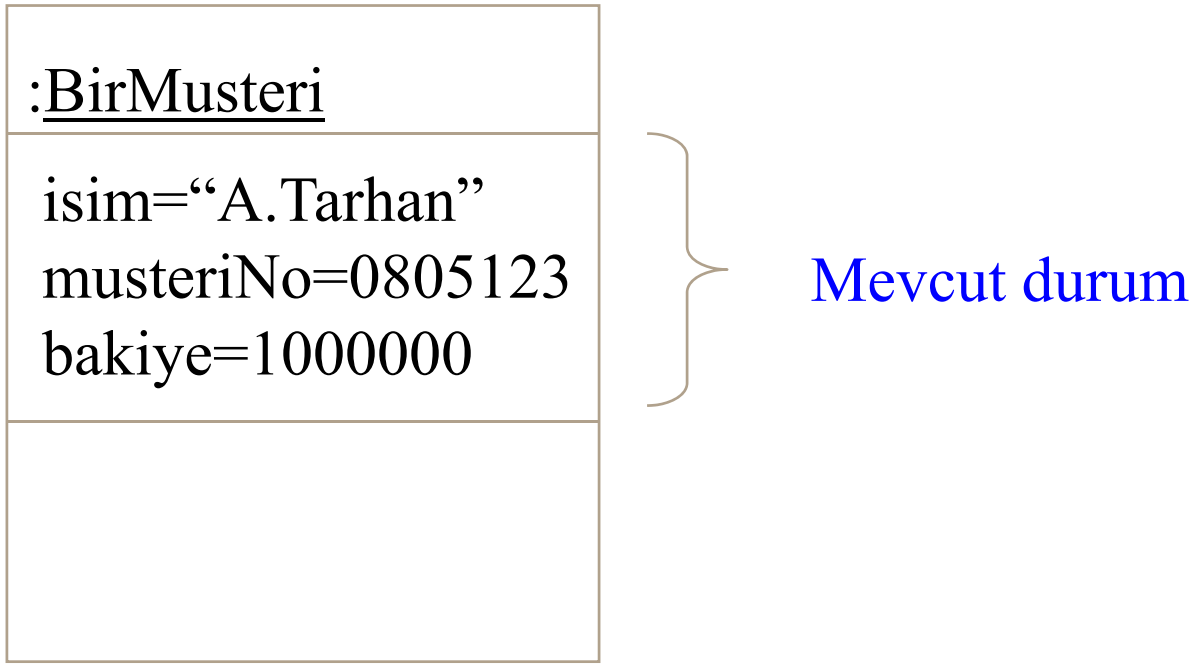
- Her nesne aşağıdakilere sahiptir:
  - Kişilik (“identity”)
  - Özellik (“attribute”)
  - Durum (“state”)
  - Davranış (“behavior”)
  - Sorumluluk (“responsibility”)

Özellikler

- Her nesnenin kendine ait bir dizi özelliği vardır.
  - Özellikler nesneye ait verileri taşır.

Durum - 1

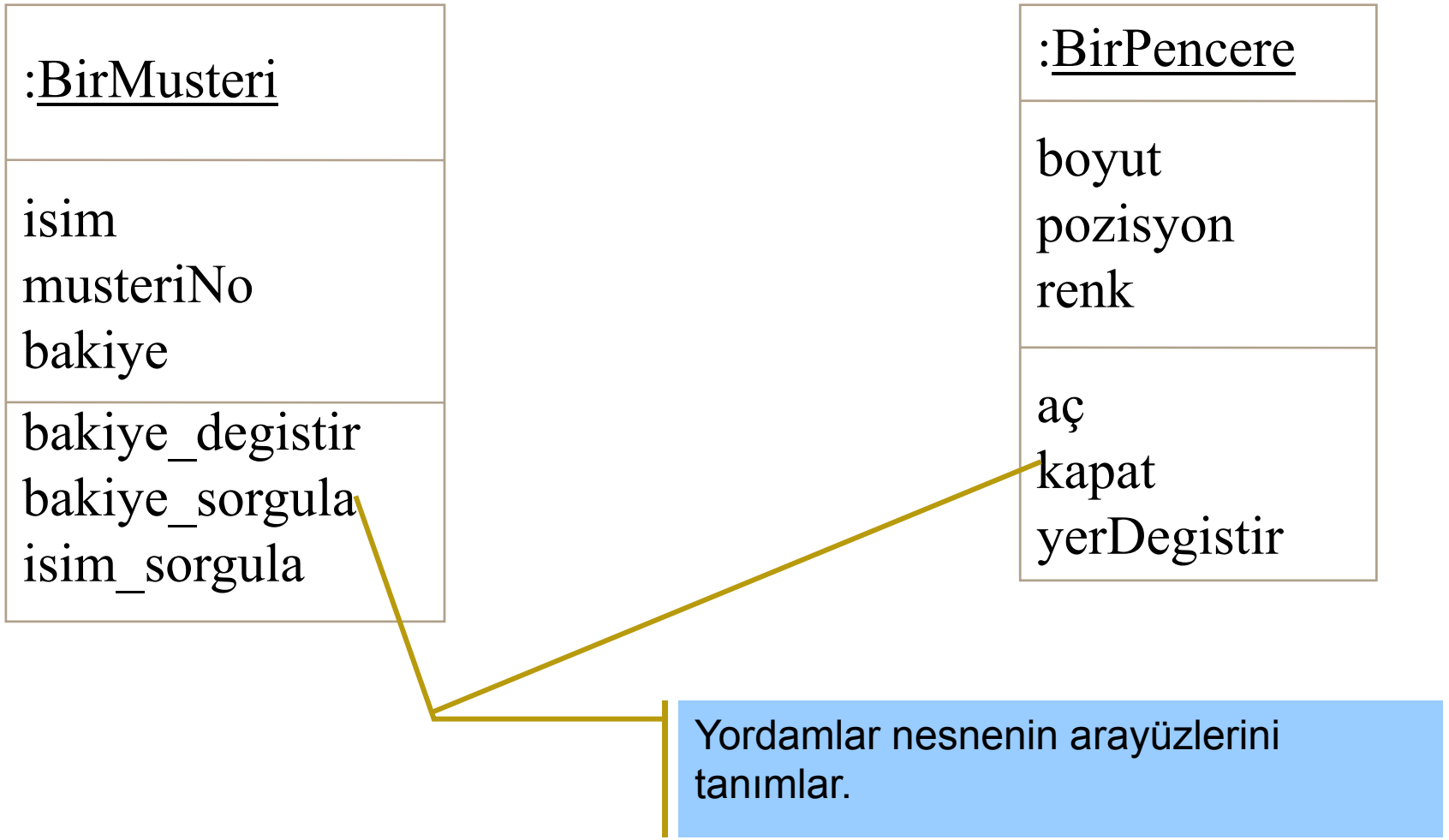
- Nesnenin tüm özellikleri ve bu özelliklerin o anki değerleri, nesnenin durumunu oluşturur.



Davranış – 1

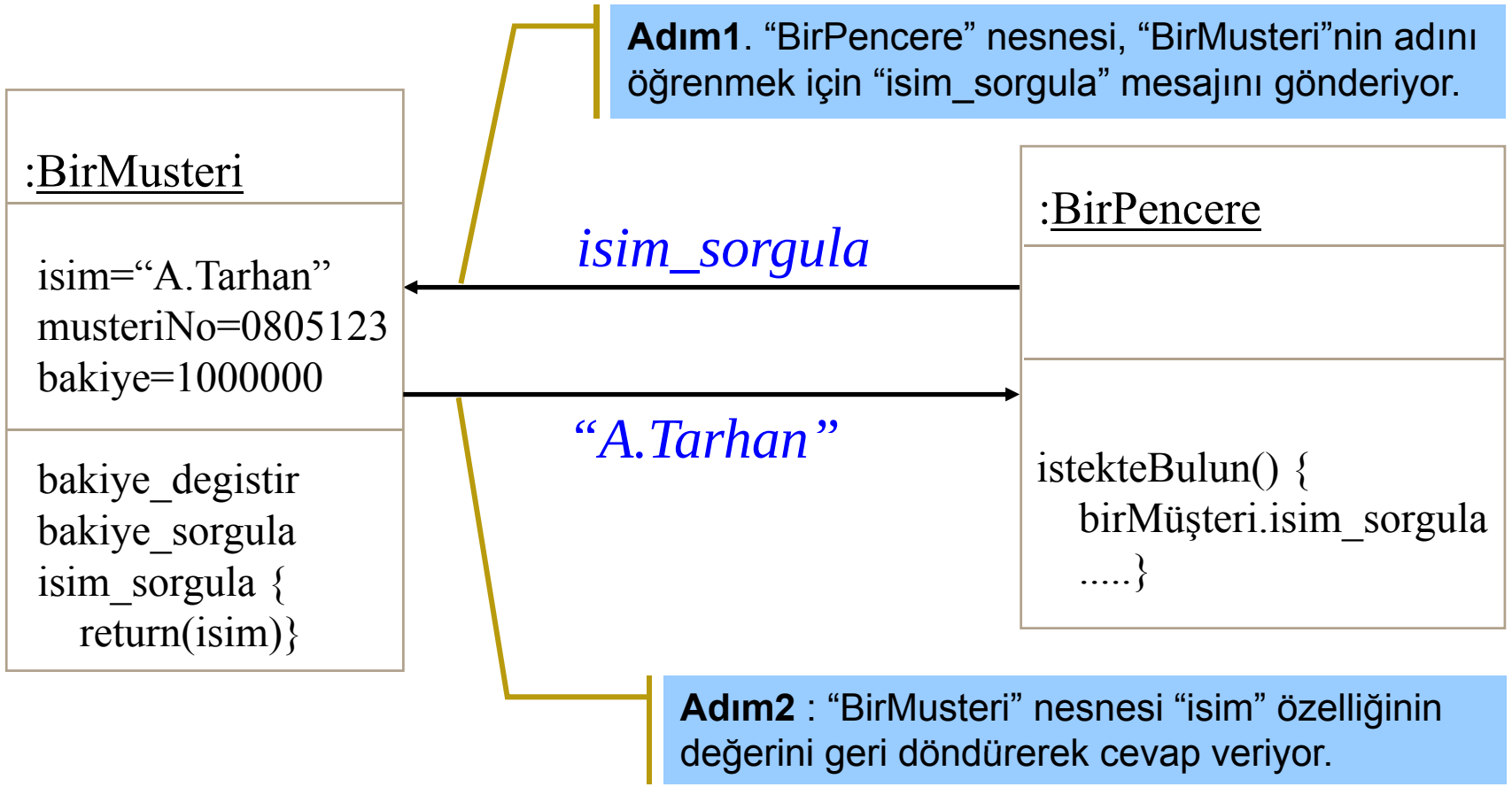
- Her nesnenin iş yapabilmeyi sağlayan tanımlı bir davranış şekli vardır.
  - Nasıl davranır, nasıl tepki verir ? (“act”/“react”)
    - Nesnenin operasyonları
    - Görülebilir aktivite
    - Diğer nesneler tarafından kullanılan arayüz (“public interface-operations”)
  - Davranışın gerçekleştirilmesi bilgisayar kodu ile yapılır.
    - Kodlanan nesne yordamları, davranışı gerçekleştirir.
    - Gizli gerçekleştirme (“encapsulated implementation”)

Davranış – 2


:BirMusteri
isim
musteriNo
bakiye
bakiye_degistir
bakiye_sorgula
isim_sorgula
:BirPencere
boyut
pozisyon
renk
aç
kapat
yerDegistir
Yordamlar nesnenin arayüzlerini tanımlar.

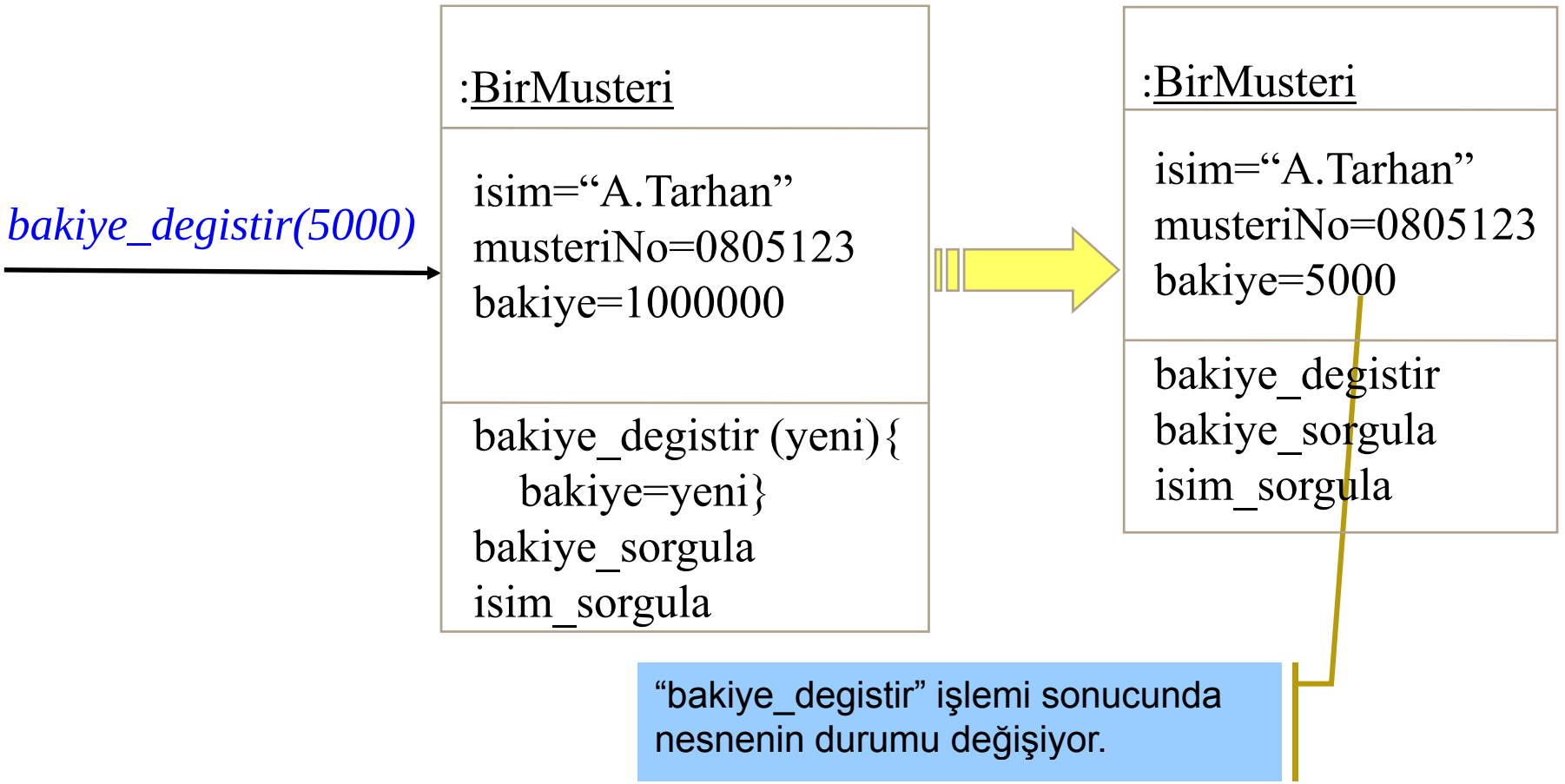
Davranış – 3

- İstek mesaj olarak iletilir.



Durum – 2

- İşlemler sonucunda nesnenin durumu değişir.



Sorumluluk

- Nesnenin sorumluluğu tüm sistemin işlevselliğine nasıl katkıda bulunacağını tanımlar.
  - Nesnenin sistemde oynayacağı rol
  - Neden böyle bir nesneye gereksinim var ?
  - Özellikler ve davranışlar, birlikte nesnenin sorumluluklarını yerine getirmesini sağlar.

İlişki (“Relationship”)

- Nesneler arasındaki fiziksel veya kavramsal bağlantı
- En yaygın ilişki: Bir nesne diğer nesnenin sunduğu servislerden yararlanır.
  - Mesaj iletimi
  - Müşteri/tedarikçi ilişkisi
- Sistem işlevselliğini sağlamak amacıyla, nesneler birbirleriyle ilişkiler aracılığı ile işbirliği yaparlar.
  - Nesne yönelimli programlama modeli

Mesajla İletişimin Getirileri

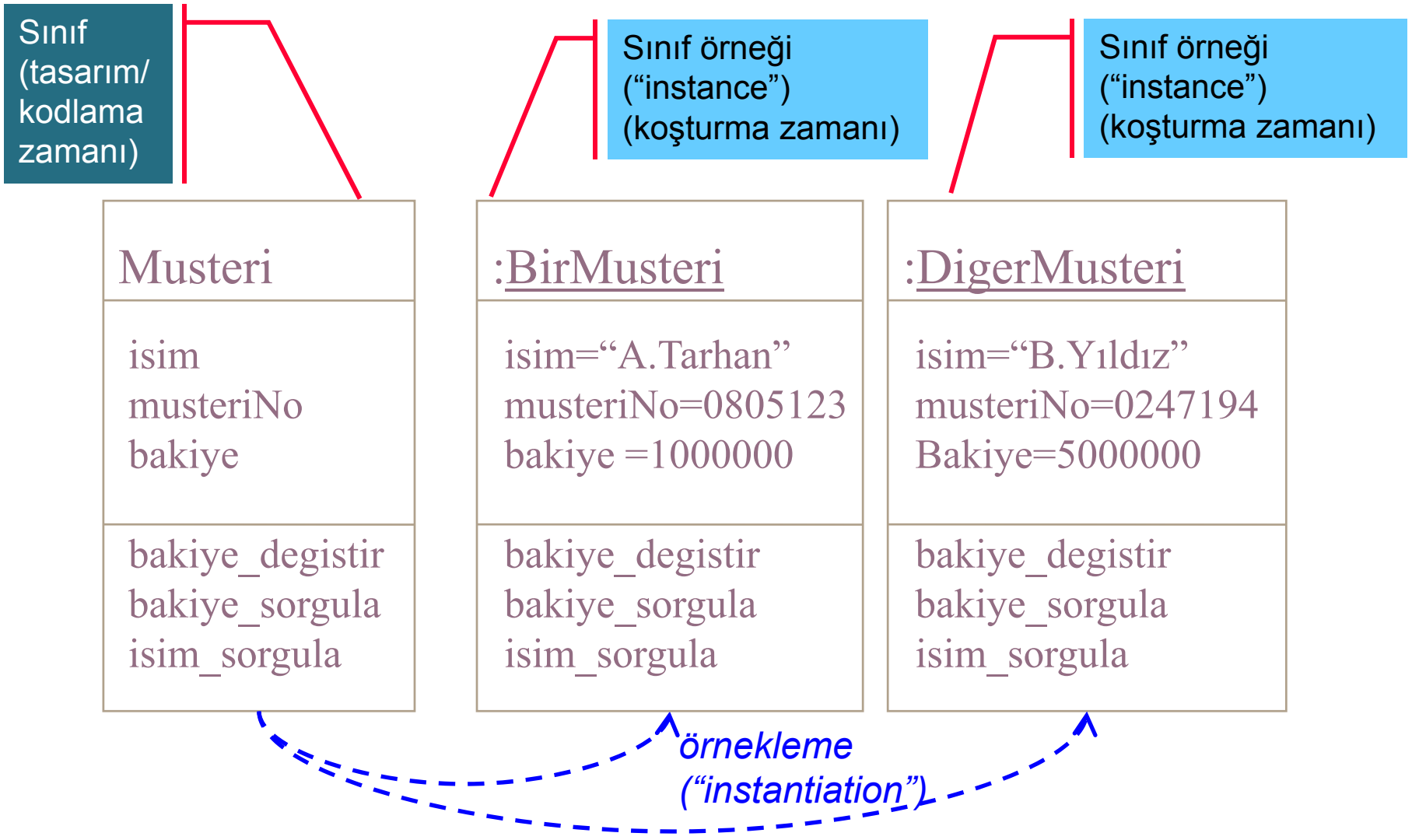
- Bir nesnenin davranışı yordamları aracılığıyla gerçekleştirilir. Mesaj gönderme nesneler arasındaki olası tüm iletişimi destekler.

- Nesneler aynı uygulama veya aynı makine içinde olmasalar bile, birbirlerine mesaj gönderip yanıt alabilirler.

“Sınıf” Kavramı

- Yapısal ve/veya davranışsal olarak aynı özelliklere sahip nesneler SINIF altında gruplanır.
  - Her nesne bir sınıfın örneğidir (“instance”).
  - Sınıfları tanımlar ve nesneleri sınıf tanımından örnekleriz (“instantiation”).
  - Her nesne ait olduğu sınıfı bilir.
- Sınıf, nesneler için şablon tanımdır.
  - Özellikler ve yordamlar sınıf için yalnızca bir kez tanımlanır.

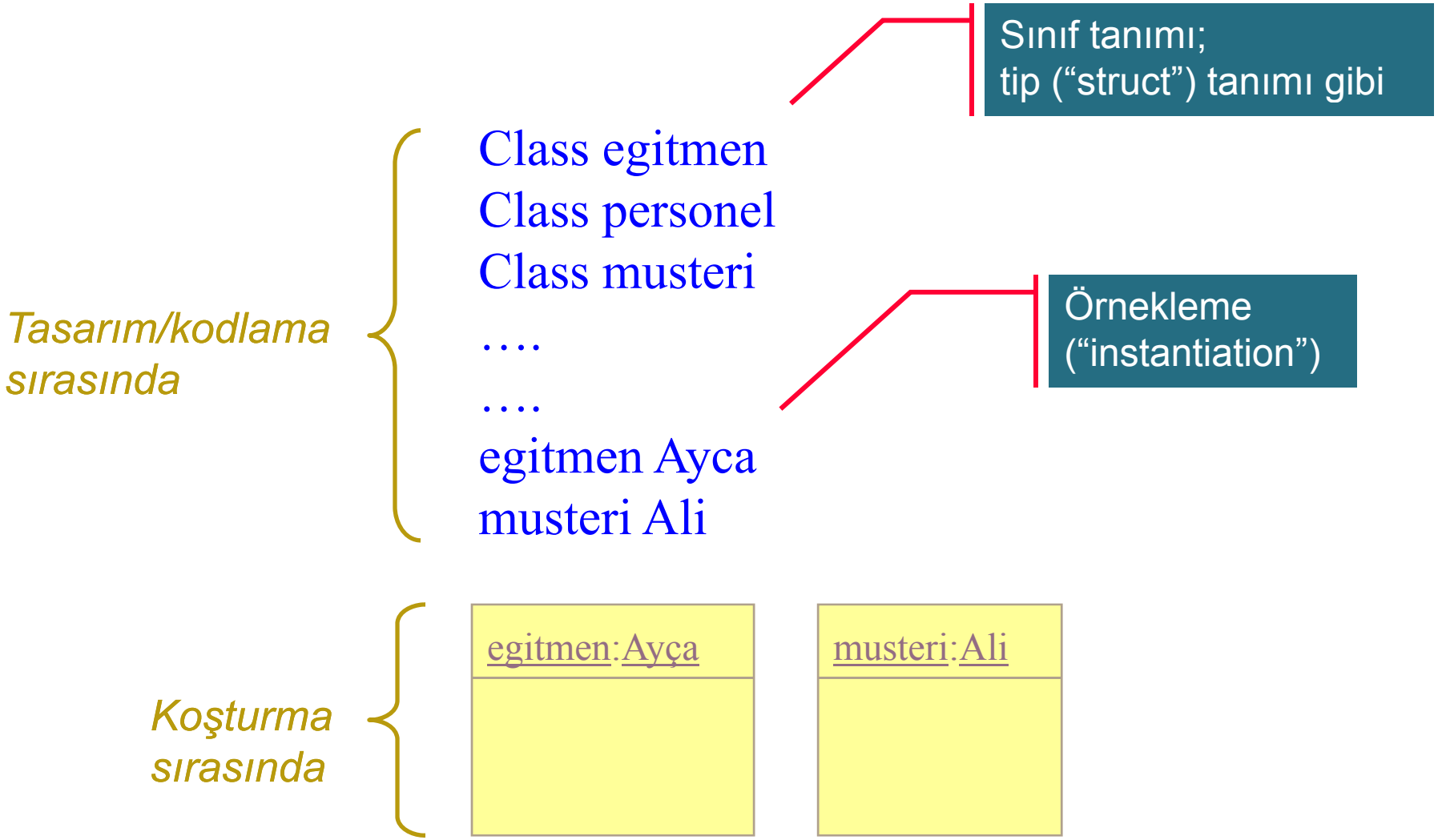
“Sınıf”: Örnek


Sınıf (tasarım/ kodlama zamanı)
Sınıf örneği (“instance”) (koşturma zamanı)
Sınıf örneği (“instance”) (koşturma zamanı)
Musteri
isim
musteriNo
bakiye
bakiye_degistir
bakiye_sorgula
isim_sorgula
:BirMusteri
isim=“A.Tarhan”
musteriNo=0805123
bakiye =1000000
bakiye_degistir
bakiye_sorgula
isim_sorgula
:DigerMusteri
isim=“B.Yıldız”
musteriNo=0247194
Bakiye=5000000
bakiye_degistir
bakiye_sorgula
isim_sorgula
örnekleme
(“instantiation”)

“Sınıf” ve “Nesne”

- Her sınıfın sıfır veya daha fazla örneği vardır.
- Sınıf statik, nesne dinamiktir.
  - Sınıfın varlığı, semantiği ve ilişkileri program koşturulmadan önce sabit olarak belirlenmiştir.
  - Nesneler program koşturulduğunda sınıf tanımından dinamik olarak yaratılırlar (“construction”).
  - Nesneler sorumluluklarını tamamladıklarında ortadan kaldırılırlar (“destruction”).
- Nesnenin sınıfı sabittir ve nesne bir kez yaratıldıktan sonra değiştirilemez.

“Sınıf” ve “Nesne”: Örnek



“Nesne” ve “Sınıf”ın Java’da Gerçekleştirilmesi

Temel Sınıf Tanımı

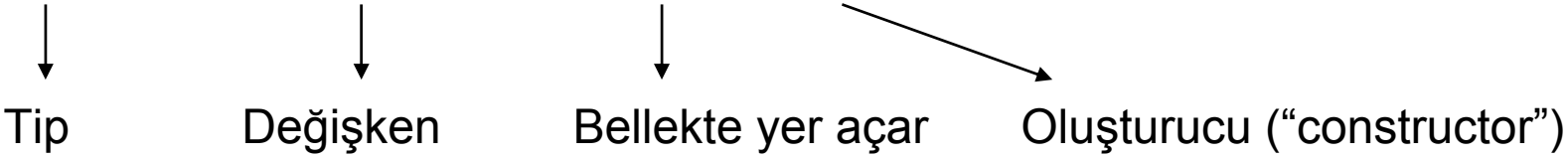
```
class   sinif_adi  {
      degisken_tipi  d1;
      degisken_tipi  d2;
      ...
      donen_tip  yordam_adi ( // parametre listesi
                              parametre_tipi  p1, parametre_tipi  p2, …) {
         …
         }
      donen_tip  yordam_adi ( // parametre listesi
                              parametre_tipi  p1, parametre_tipi  p2, …) {
         …
         }
      }
```

Temel Sınıf Tanımı: Örnek - 1

- Kutu ve KutuDemo sınıf tanımları

“Sınıf” Tipinde “Nesne” Oluşturmak - 1

- *Sınıf-adı nesne-değişkeni* = new *Sınıf-adı ();*



- Java’da nesneler sınıf tanımından oluşturulur.
  - Nesne (“Object”): Sınıf Örneği (“Class Instance”)

“Sınıf” Tipinde “Nesne” Oluşturmak - 2

- Örnek:
  - Kutu kutuX = new Kutu();
    - “Kutu” bir tip olarak belirtildiğinden sınıf tanımıdır.
    - “kutuX”, “Kutu” sınıfının bir örneğidir.
    - “kutuX” nesnesi, “Kutu” sınıfına aittir.

- Sınıf tanımından nesne oluşturulurken 3 temel işlev gerçekleştirilir.
  - Tanımlama (“declaration”)
  - Örnekleme (“instantiation”)
  - İlklendirme (“initialization”)

“Sınıf” Tipinde “Nesne” Oluşturmak - 3

- Tanımlama: Nesne için değişken tanımlama
  - Kutu kutuX;
- Örnekleme: Sınıf tipinde bir nesne oluşturma
  - kutuX = new Kutu ();
- İlklendirme: Nesnenin özelliklerine değer atama
  - kutuX = new Kutu (10, 20, 15);

Hepsi birarada:

```
Kutu kutuX = new Kutu (10, 20, 15);
```

Kullanılmayan Nesnelerin Temizlenmesi

- JRE “çöp toplama” (“garbage collection”) özelliğiyle, kullanılmayan nesneler için periyodik olarak belleği boşaltır.
  - Çöp toplama, programdan gelen bellek atama taleplerine göre, bellekte yer olmadığı değerlendirilirse taleple senkronize olarak gerçekleştirilir.
  - ***System.gc()*** yordamı çağrılır.

- Sonlandırma: Bazen nesne yok edilmeden önce bazı işlemler yapması istenebilir. Bu işlemler *finalize()* yordamı içine yazılır ve nesne bellekten atılmadan hemen önce gerçekleştirilir.

```
protected  void  finalize () {
    // sonlandırma kodu
    }
```

Örnek

- Adınızı okuyan ve adınızı “Merhaba” ile birlikte ekrana yazan program

Değişken Tanımlama

Değişken Tipleri

- Java'da kullanılan tüm değişkenlerin bir veri tipi olmak zorundadır.
  - İki tür veri tipi vardır: Temel ("primitive"), Referans ("reference")
- Temel tipler, tek bir değeri tutabilir.
  - Tamsayı ("integer")
  - Kayan noktalı ("floating point")
  - Karakter ("character")
  - İkili ("boolean")
  - vb.
- Referans tipler, değişken tarafından temsil edilen gerçek değere veya değer kümesine referans eder.
  - Dizi ("array")
  - Sınıf ("class")
  - Arayüz ("interface")
  - vb.

Temel Veri Tipleri (“Primitive Data Types”)

Tip	İçerik	Varsayılan Değer	Büyüklük	Minimum Değer Maksimum Değer
boolean	Doğru (“true”) veya Yanlış (“false”)	Yanlış (“false”)	1 bit	N.A N.A
char	Unicode karakter	\u0000	16 bit	\u0000 \uFFFF
byte	İşaretli tamsayı (“signed integer”)	0	8 bit	-128 127
short	İşaretli tamsayı (“signed integer”)	0	16 bit	-32768 32767
int	İşaretli tamsayı (“signed integer”)	0	32 bit	-2147483648 2147483647
long	İşaretli tamsayı (“signed integer”)	0	64 bit	-9223372036854775808 9223372036854775807
float	IEEE 754 kayan-noktalı (“floating-point”)	0.0	32 bit	± 3.40282347E+38 ± 1.40239846E-45
double	IEEE 754 kayan-noktalı (“floating-point”)	0.0	64 bit	± 1.79769313486231570E+308 ± 4.94065645841246544E-324

Temel Veri Tipleri için Değerler (“Literals”)

- boolean: true / false
- int: 89, -945, 37865
- long: 89L, -945L, 5123567876L
- float: 89.5f, -32.5f
- double: 89.5, -32.5, 87.6E45
- char: 'c', '9', 't'
- String: “Bu bir dizgidir."

Örnekler

```
class Degiskenler {
   public static void main (String args[]) {
      boolean b = true;
      short kucuk = 1;
      int orta = 23;
      long buyuk = 76L;
      float pi = 3.1415292f;
      double d = 2.71828;
      String s = “Merhaba Dünya!";
      System.out.println(“boolean b = ”+ b);
      System.out.println(“short kucuk= ”+ kucuk);
      System.out.println(“int orta= ”+ orta);
      System.out.println(“long buyuk= ”+ buyuk);
      System.out.println(“float pi= ”+ pi);
      System.out.println(“double d= ”+ d);
      System.out.println(“String s= ”+ s);
   }
}
```

Konsol çıktısı:

```
boolean b= true
short kucuk= 1
int orta= 23
long buyuk= 76
float pi= 3.1415292
double d= 2.71828
String s= Merhaba Dünya!
```

Değişken Tanımlama Kuralları - 1

- Her değişken tanımlama ifadesi bir ya da birden fazla değişkeni tanımlar.
  - Aynı türe sahip olan değişkenler tek bir tanımlama ifadesi ile tanımlanabilirler.
  - Eğer birden fazla değişken tanımlanacaksa, değişkenlerin adları virgül ile ayrılır.
  - Örnek: `int kucuk, buyuk, orta;`
- Her değişkenin adı bir tanımlayıcıdır.
  - Tanımlayıcılar; harfleri, rakamları, '_' ya da '$' karakterlerini içeren fakat rakamlar ile başlamayan sözcüklerdir.
- Java büyük harf küçük harf ayrımına duyarlı olduğundan, aynı harfin büyük harfi küçük harfinden farklıdır.
  - Örnek: `int orta, ortA;` `// iki farklı değişken tanımlar`

Değişken Tanımlama Kuralları - 2

- Anahtar sözcük (“keyword”) veya hazır bilgi (“literal”) olmamalıdır.
  - Örnek: true, false, if, then, else, …
- Aynı kapsamda tanımlı bir başka değişkenle aynı olmamalıdır.
- Küçük harfle başlamalıdır. Birleşik sözcük ise ilk sözcük küçük harfle, diğer sözcükler büyük harfle başlar.
  - Örnek: int sayi, asalSayi;

Kapsam (“Scope”)

- Bir değişkenin kapsamı, değişkenin ulaşılabilir olduğu kod bloğunu ifade eder.

- Bir değişken 4 farklı kapsamda tanımlanabilir:
  - Üye değişken (“member variable”)
  - Yerel değişken (“local variable”)
  - Yordam parametresi (“method parameter”)
  - Kural dışı durum işleme parametresi (“exception handler parameter”)

Üye Değişken (“Member Variable”)

- Bir sınıf veya nesnenin üyesidir ve sınıf tanımı içinde yer alır.

- Örnek:

```
class TamsayiSinifi {
        int  sayi;
        …
        // metotları tanımla
        …
        }
```

Yerel Değişken ("Local Variable")

- Sınıf yordamları içinde veya yordam içindeki kod bloğunda tanımlanır.
- Örnek:

```
public class Faktoriyel {
        long faktoriyel (long sayi) {
                long sonuc = 1;
                if ((sayi < 0) || (sayi > 20))
                        sonuc = 0;
                else
                        for (long i=1; i <= sayi; i++)
                                sonuc = sonuc * i;
                return (sonuc);
                }
        }
```

Yordam Parametresi ("Method Parameter")

- Yordam veya oluşturuculara ("constructors") değer geçirmek için kullanılan parametrelerdir.

- Örnek:

```
public class Faktoriyel {
        long faktoriyel (long sayi) {
                long sonuc = 1;
                if ((sayi < 0) || (sayi > 20))
                        sonuc = 0;
                else
                        for (long i=1; i <= sayi; i++)
                                sonuc = sonuc * i;
                return (sonuc);
                }
        }
```

Kural Dışı Durum İşleme Parametresi ("Exception Handler Parameter")

- Yordam parametrelerine benzer; ancak yordam veya oluşturucu yerine, kural dışı durum işleyiciye parametre geçirir.
- "try/catch" komut dizisi, belli bir kod bloğunu korur ve oluşmaları halinde tanımlanan tipteki kural dışı durumları yakalar.
  - Örnek:

```
try {
        dosyadanOku ("tmp");
        …
        }
catch (Exception e) {
        System.out.println ("Okuma işleminde hata: " + e);
        …
        }
```

Ödev-2

1. Kendi adınızı taşıyan bir Java sınıfı oluşturun.
2. (1) adımında oluşturduğunuz sınıfın içinde, bu derste öğrendiğiniz temel veri tiplerini örnekleyen özellikler tanımlayın.
3. (2) adımında tanımladığınız her özellik için, bu özelliğe değer atayan ve değeri ekrana yazdıran birer metot tanımlayın.
4. *"main()"* metodu içinden, (3) adımında tanımladığınız metotları çağırarak özellik değerlerinin ekranda yazılmasını sağlayın.

- Oluşan .java ve .class dosyalarını zipleyerek, dersin eğitmenine e-posta ile gönderin.
  - Teslim için son tarih: 10.Kasım.2009 Salı 24:00
  - E-posta adresi: atarhan@hacettepe.edu.tr