

# Java Programlama, Veri Yapıları ve Bellek Yönetimi

3. Hafta

Dr. Öğr. Üyesi BÜŞRA ÖZDENİZCİ KÖSE

İşletme Bölümü

İşletme Fakültesi


# Çalışma Soruları

- Carpim.java – İki integer (tamsayı) değişkenin çarpımını hesaplayan bir program oluşturunuz; değişkenlerin değerlerini 20 ve 40 olarak atayabilirsiniz. *Algoritma tasarımı yapmayı unutmayınız.*

01	20 ile 40 değerlerinin çarpım sonucu 800
----	--

- Ortalama.java – Üç integer (tamsayı) değişkenin ortalamasını hesaplayan bir program oluşturunuz; değişkenlerin değerlerini 10, 20, 30 olarak atayabilirsiniz. *Algoritma tasarımı yapmayı unutmayınız.*

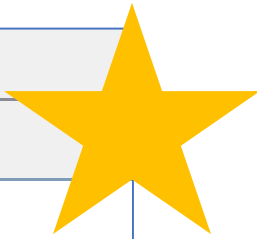
01	10, 20 ve 30 değerlerinin ortalaması 20.0
----	---




Start Page X Carpim.java X

Source History

```
1  ...5 lines
6  package carpim;
7  /**...4 lines */
11 public class Carpim {
12     /**...3 lines */
15     public static void main(String[] args) {
16
17         int sayi1 = 20;
18         int sayi2 = 40;
19
20         int sonuc;
21         sonuc = sayi1*sayi2 ;
22
23         System.out.println (sayi1 + " ile " + sayi2 + " çarpımının sonucu "+sonuc);
24     }
25
26 }
27
```



```
Start Page x Carpim.java x
Source History
1 + ...5 lines
6 package carpim;
7 + /**...4 lines */
11 public class Carpim {
12 + /**...3 lines */
15 - public static void main(String[] args) {
16
17     int sayi1;
18     int sayi2;
19     int sonuc;
20
21     sayi1 = 20;
22     sayi2 = 40;
23
24     sonuc = sayi1*sayi2 ;
25
26     System.out.println (sayi1 + " ile " + sayi2 + " çapımının sonucu "+sonuc);
27 }
28
29 }
30
```



Start Page X Carpin.java X

Source History

```
1  ...5 lines
6  package carpin;
7  /**...4 lines */
11 public class Carpin {
12     /**...3 lines */
15     public static void main(String[] args) {
16
17         int sayi1, sayi2, sonuc;
18
19         sayi1 = 20; sayi2 = 40;
20
21         sonuc = sayi1*sayi2 ;
22
23         System.out.println (sayi1 + " ile " + sayi2 + " çarpımının sonucu "+sonuc);
24     }
25
26 }
27
```



```
run:
20 ile 40 çapımının sonucu 800
BUILD SUCCESSFUL (total time: 0 seconds)

package carpim;

/**...4 lines */

public class Carpim {
    /**...3 lines */

    public static void main(String[] args) {

        int sayi1, sayi2, sonuc;

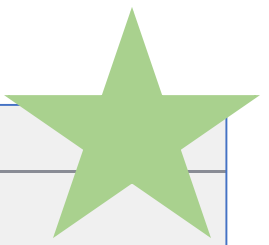
        sayi1 = 20 ;
        sayi2 = 40;

        sonuc = sayi1*sayi2 ;

        System.out.println (sayi1 + " ile " + sayi2 + " çapımının sonucu "+sonuc);
    }
}
```




```
Start Page x Ortalama.java x
Source History
1 ...5 lines
6 package ortalama;
7 /**...4 lines */
11 public class Ortalama {
12     /**...3 lines */
15     public static void main(String[] args) {
16
17         int sayi1 = 10;
18         int sayi2 = 20;
19         int sayi3 = 30;
20         int sonuc;
21
22         sonuc =(sayi1+sayi2+sayi3)/3;
23
24         System.out.println(sayi1 + "," + sayi2 + " ve " + sayi3 + " değerlerinin ortalaması " +sonuc);
25
26     }
27 }
```



```
Start Page X Ortalama.java X
Source History
1 ...5 lines
6 package ortalama;
7 /**...4 lines */
11 public class Ortalama {
12     /**...3 lines */
15     public static void main(String[] args) {
16
17         int sayi1 = 10, sayi2 = 20, sayi3 = 30;
18         int sonuc;
19
20         sonuc =(sayi1 + sayi2 + sayi3 )/3;
21
22         System.out.println(sayi1 + "," + sayi2 + " ve " + sayi3 + " değerlerinin ortalaması " +sonuc);
23
24     }
25 }
26
```





Start Page x Ortalama.java x

Source History

```
1  ...5 lines
6  package ortalama;
7  /**...4 lines */
11 public class Ortalama {
12     /**...3 lines */
15     public static void main(String[] args) {
16
17         int sayi1, sayi2, sayi3, sonuc;
18
19         sayi1=10;
20         sayi2=20;
21         sayi3=30;
22
23         sonuc =(sayi1 + sayi2 + sayi3 )/3;
24
25         System.out.println(sayi1 + "," + sayi2 + " ve " + sayi3 + " değerlerinin ortalaması " +sonuc);
26
27     }
28 }
```



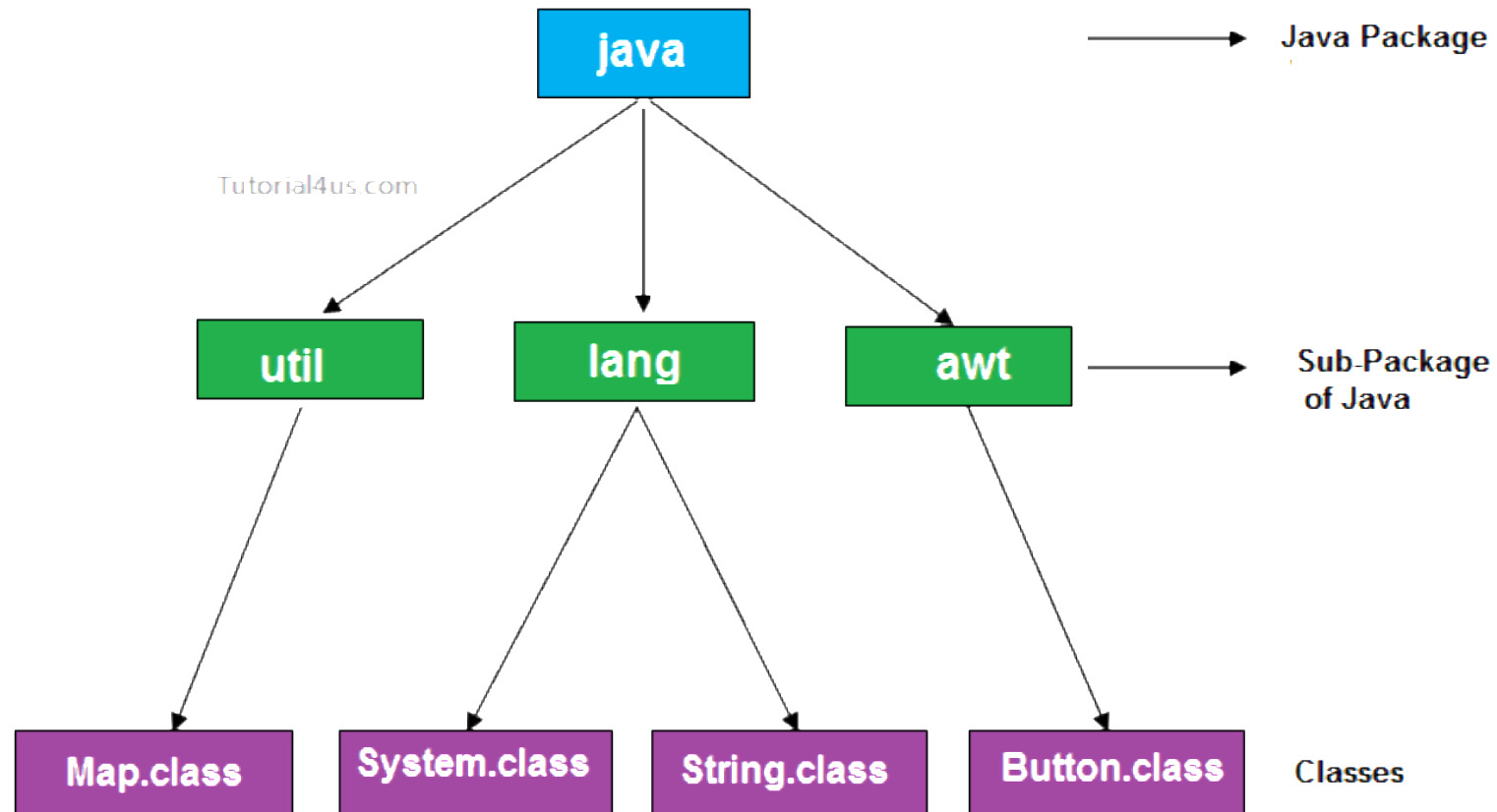
The screenshot displays an IDE with three main panels. The left panel shows a project tree with a package named 'Ortalama' containing a file 'Ortalama.java'. The middle panel shows the output of a successful run: '30,35 ve 25 değerlerinin ortalaması 30' and 'BUILD SUCCESSFUL (total time: 0 seconds)'. The right panel shows the source code of 'Ortalama.java'.

```
1  ...5 lines
6  package ortalama;
7  /**...4 lines */
11 public class Ortalama {
12  /**...3 lines */
15  public static void main(String[] args) {
16
17      int sayi1, sayi2, sayi3, sonuc;
18
19      sayi1=30;
20      sayi2=35;
21      sayi3=25;
22
23      sonuc =(sayi1 + sayi2 + sayi3 )/3;
24
25      System.out.println(sayi1 + "," + sayi2 + " ve " + sayi3 + " değerlerinin ortalaması " +sonuc);
26
27  }
28  }
29
```

# Java Packages & Classes

- Java programları paketlerden oluşur; bir paket içinde tanımlı bulunan sınıflar ve arayüzler, tekrar tekrar kullanılabilirler.
  - **Packages (paketler)** fonksiyonel olarak ilişkili **sınıfları (classes)** ve arayüzleri (interfaces) bir araya toplayan, erişim koruması (access protection) sağlayan vb. yapılardır
- Java'da bulunan önemli paketlerin bir kısmı:
  - java.lang - System, Math, String vb. temel işlemler yapan sınıfları içerir
  - java.util - Scanner, Stack, Map vb. utility sınıfları içerir
  - java.io, java.net, java.awt, java.applet ...
- Bir paketteki sınıfı kullanmak için: **paket\_ismi.sinif\_ismi**

# Java Packages & Classes Examples



# java.util Package ve Scanner Class

- **Scanner** sınıfı, **java.util** paketinin içinde bulunur, kullanıcıdan veri okumamızı veya almamızı sağlayan önemli bir sınıftır
- Bu sınıfın içinde çeşitli metotlar (hazır fonksiyonlar) bulunmaktadır.
- Bu sınıfa ait bir metot kullanmak için şu komutu kullanmamız gerekir:

```
import java.util.Scanner;
```

# Scanner Sınıfına ait Metotlar

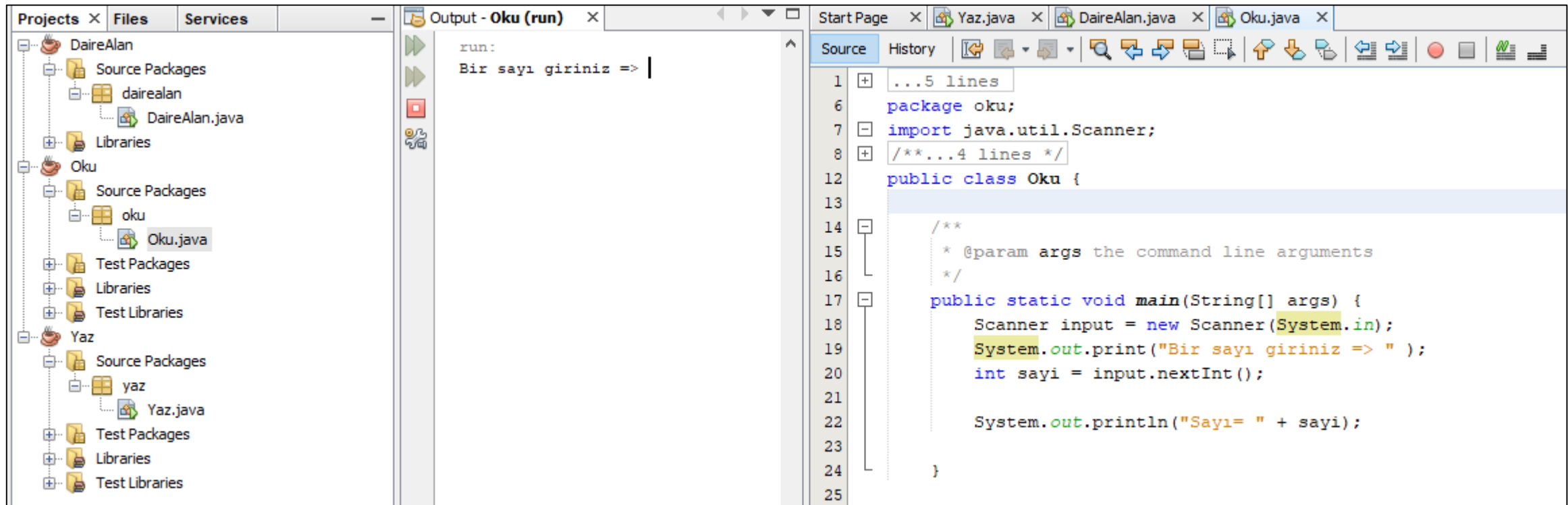
**TABLE 2.1** Methods for **Scanner** Objects

<i>Method</i>	<i>Description</i>
<code>nextByte()</code>	reads an integer of the <code>byte</code> type.
<code>nextShort()</code>	reads an integer of the <code>short</code> type.
<code>nextInt()</code>	reads an integer of the <code>int</code> type.
<code>nextLong()</code>	reads an integer of the <code>long</code> type.
<code>nextFloat()</code>	reads a number of the <code>float</code> type.
<code>nextDouble()</code>	reads a number of the <code>double</code> type.
<code>next()</code>	reads a string that ends before a whitespace character.
<code>nextLine()</code>	reads a line of text (i.e., a string ending with the <i>Enter</i> key pressed).

# Paketleri veya Sınıfları Kullanmak için: Import Komutu

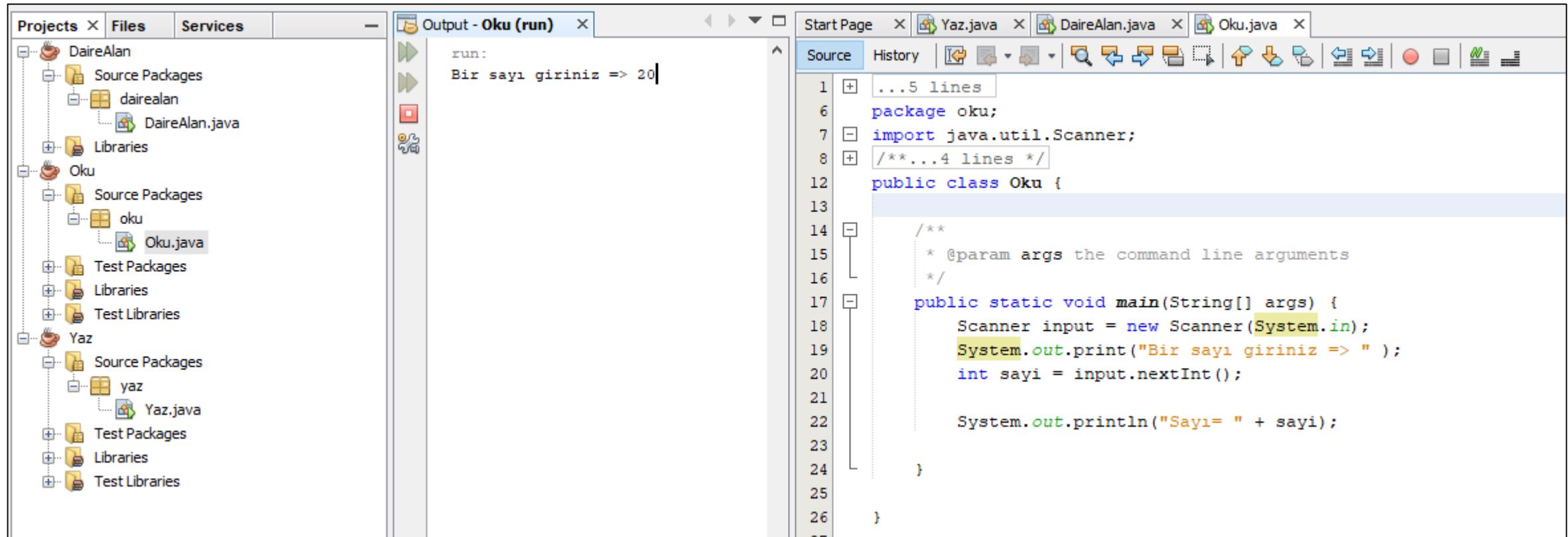
- Programlamaya başlamadan önce, algoritma tasarımı ile hangi paketlere veya sınıflara ihtiyacımız olduğunu tespit etmemiz gerekir.
- İki farklı şekilde **import** komutu kullanılır: *specific import* ve *wildcard import*.
- Belli bir sınıfı sadece import etmek için paket ismi ve sınıf ismi belirtilir:  
**import java.util.Scanner;**
- Bir paketin tüm sınıflarını import etmek için \* sembolünü kullanabiliriz:  
**import java.util.\*;**

# Şimdi, Kullanıcıya soralım...

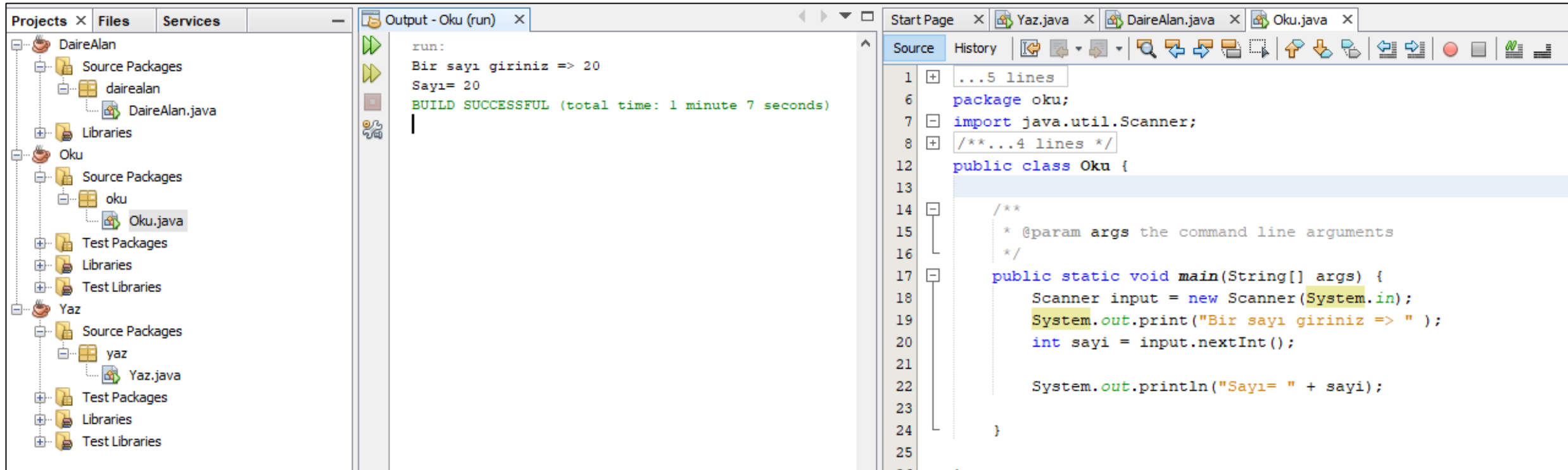




# Kullanıcı bir sayı giriyor...



# Kullanıcının girdiği sayıyı, ekrana yazdıralım !



01	<b>import java.util.Scanner;</b>	<i>Okumak için Scanner sınıfını kullanacağımızı belirtme</i>
02	<b>public class Oku {</b>	
03	<b>public static void main( String[] args ) {</b>	
04	<b>Scanner input = new Scanner(System.in) ;</b>	<i>Okuma yapabilmek için input isimli okuyucu nesnesi tanımlama</i>
05	<b>System.out.print("Bir sayı giriniz =&gt; " ) ;</b>	
06	<b>int sayi = input.nextInt() ;</b>	<i>sayi değişkenini tanımlama ve Kullanıcının girdiği tamsayıyı okuyarak sayi değişkenine atama</i>
07	<b>System.out.println("Sayı= " + sayi) ;</b>	<i>print()metodu ile bir String ile birlikte başka değer de yazdırmak istediğimizde + operatörünü bu şekilde kullanabiliriz.</i>
08	<b>}</b>	
09	<b>}</b>	
01	Bir sayı giriniz => 100	
02	Sayı = 100	



# AlanHesapla.java



- AlanHesapla.java programımızı güncelleyelim.
- Kullanıcıdan yarıçap bilgisini alarak dairenin alanını hesaplayalım ve ekrana yazdıralım.

```
01 Yarıçap => 10
02 Dairenin alanı 314
```

DO IT  
NOW!

```
01 import java.util.Scanner;
02 public class AlanHesapla {
03     public static void main( String[] args ) {
04
05         double yaricap;
06         double alan;
07         Scanner input = new Scanner(System.in);
08
09         System.out.println("Yarıçap => ");
10         yaricap = input.nextDouble();
11
12         alan = yaricap * yaricap * 3.14;
13
14         System.out.println("Dairenin alanı "+ alan);
15     }
16 }
```

Değer saklayacak olan gerekli  
Değişkenleri tanımlama

Kullanıcıdan veri okuma

Sonucu hesapla,  
değeri değişkene ata  
ve sonucu görüntüle



```
01 import java.util.Scanner;
02 public class AlanHesapla {
03     public static void main( String[] args ) {
04         Scanner input = new Scanner(System.in);
05
06         System.out.println("Yarıçap => ");
07
08         double yaricap = input.nextDouble();
09
10         double alan = yaricap * yaricap * 3.14;
11
12         System.out.println("Dairenin alanı "+ alan);
13     }
14 }
```

# Topla.java



- Topla.java – Kullanıcıdan iki sayı (integer) okuyunuz ve toplamını ekrana yazdırınız!

```
01 Bir sayı giriniz => 10
02 Tekrar bir sayı giriniz => 20
03 Toplamı 30
```

Output - Topla (run) X

run:  
Birinci sayınızı giriniz => 24  
İkinci sayınızı giriniz => 12  
24 ile 12 değerlerinin toplamı 36  
BUILD SUCCESSFUL (total time: 4 seconds)

Start Page X Topla.java X

Source History

1 ...5 lines

6 package topla;

7

8 import java.util.Scanner;

9 /\*\*...4 lines \*/

13 public class Topla {

14 /\*\*...3 lines \*/

17 public static void main(String[] args) {

18

19 int a;

20 int b;

21 int toplam;

22

23 Scanner busra = new Scanner(System.in);

24

25 System.out.print ("Birinci sayınızı giriniz => ");

26 a = busra.nextInt();

27

28 System.out.print ("İkinci sayınızı giriniz => ");

29 b = busra.nextInt();

30

31 toplam = a+b;

32

33 System.out.println (a + " ile " + b + " değerlerinin toplamı " +toplam);

34

35 }

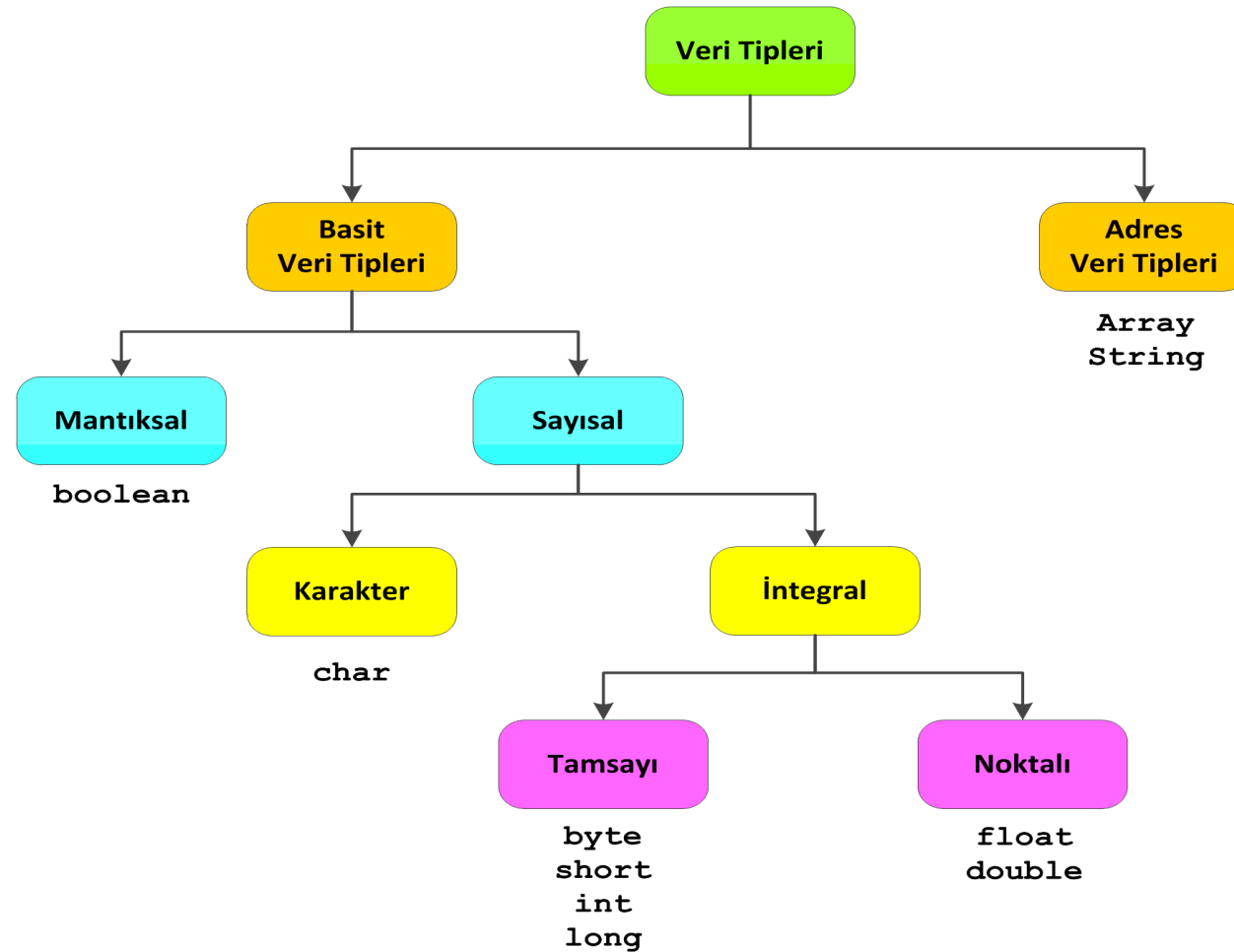
36

37 }



```
1  ...5 lines
6  package top1a;
7
8  import java.util.Scanner;
9  /**...4 lines */
13 public class Topla {
14     /**...3 lines */
17     public static void main(String[] args) {
18
19         Scanner busra = new Scanner(System.in);
20
21         System.out.print ("Birinci sayınızı giriniz => ");
22         int a = busra.nextInt();
23
24         System.out.print ("İkinci sayınızı giriniz => ");
25         int b = busra.nextInt();
26
27         int toplam = a+b;
28
29         System.out.println (a + " ile " + b + " değerlerinin toplamı " +toplam);
30
31     }
32
33 }
```

# Java'da Veri Türleri



	İsmi	Uzunluk	En Küçük Değer	En Büyük Değer
INTEGER (Tamsayı)	byte	1 byte	$-2^7$ (-128)	$2^7 - 1$ (+127)
	short	2 byte	$-2^{15}$ (-32.768)	$2^{15} - 1$ (+32.767)
	int	4 byte	$-2^{31}$ (-2.147.483.648)	$2^{31} - 1$ (+2.147.483.647)
	long	8 byte	$-2^{63}$	$2^{63} - 1$
FLOATING POINT (Gerçel Sayı)	float	4 byte	$-1,7 * 10^{38}$	$1,7 * 10^{38}$
	double	8 byte	$-3,4 * 10^{38}$	$3,4 * 10^{38}$
BOOLEAN (Mantıksal)	boolean	1 bit	true	false
CHARACTER (Karakter)	char	2 byte	0	$2^{16} - 1$ (+65.535)

# Sayısal Veri Türleri: Tamsayılar

- Java'da dört önemli tamsayı türü vardır: **byte**, **short**, **int**, ve **long**.
- Küçük sayıların atanabileceği **byte** veri türündeki bir değişken için bellekte 1 byte uzunlukta yer ayrılırken, diğer veri türlerinde ise o değişkene atanabilecek daha büyük verileri kaydedebilmek üzere daha fazla yere ayrılır.
  - Örneğin, saklayacağınız verinin büyüklüğünün **byte** değişkeni sınırları içerisinde olduğunu biliyorsanız, değişkeni **byte** olarak tanımlayınız
  - Biz genellikle daha kolay kullanım sağladığı için **int** veri türünü tercih edeceğiz

	İsmi	Uzunluk	En Küçük Değer	En Büyük Değer
INTEGER (Tamsayı)	byte	1 byte	$-2^7$ (-128)	$2^7 - 1$ (+127)
	short	2 byte	$-2^{15}$ (-32.768)	$2^{15} - 1$ (+32.767)
	int	4 byte	$-2^{31}$ (-2.147.483.648)	$2^{31} - 1$ (+2.147.483.647)
	long	8 byte	$-2^{63}$	$2^{63} - 1$
FLOATING POINT (Gerçek Sayı)	float	4 byte	$-1,7 * 10^{38}$	$1,7 * 10^{38}$
	double	8 byte	$-3,4 * 10^{38}$	$3,4 * 10^{38}$
BOOLEAN (Mantıksal)	boolean	1 bit	true	false
CHARACTER (Karakter)	char	2 byte	0	$2^{16} - 1$ (+65.535)

# Tamsayı değerler nasıl atanmalı?

Doğru Tanımlama	Yanlış Tanımlama
<b>-100</b>	-100.0
<b>300000</b>	300.000
<b>+100</b>	100,0
<b>3</b>	3.5
<b>100</b>	100.

Nokta (.) ve virgül (,) kullanılmaz !

Veri Türü	YANLIŞ	NEDEN?
byte	128	127'den büyük olamaz!
	127.0	Nokta ve sonraki rakam kullanılamaz!
	-129	-128'den küçük olamaz!
short	-32769	-32768'den küçük olamaz!
	+32768	32767'den büyük olamaz!
	32,767	Virgül kullanılamaz!
	32.767	Nokta kullanılamaz!
int	12521432768	Bu büyüklükte bir değer saklanamaz!
	32,767	Virgül kullanılamaz!
	32.767	Nokta kullanılamaz!

# Tamsayı değerler nasıl okunabilir? Scanner Sınıf Metotları

Method	Data Type	Usage
<code>nextByte()</code>	byte	<code>byte b = input.nextByte();</code>
<code>nextShort()</code>	short	<code>short s = input.nextShort();</code>
<code>nextLong()</code>	long	<code>long l = input.nextLong();</code>
<code>nextInt()</code>	int	<code>int i = input.nextInt();</code>



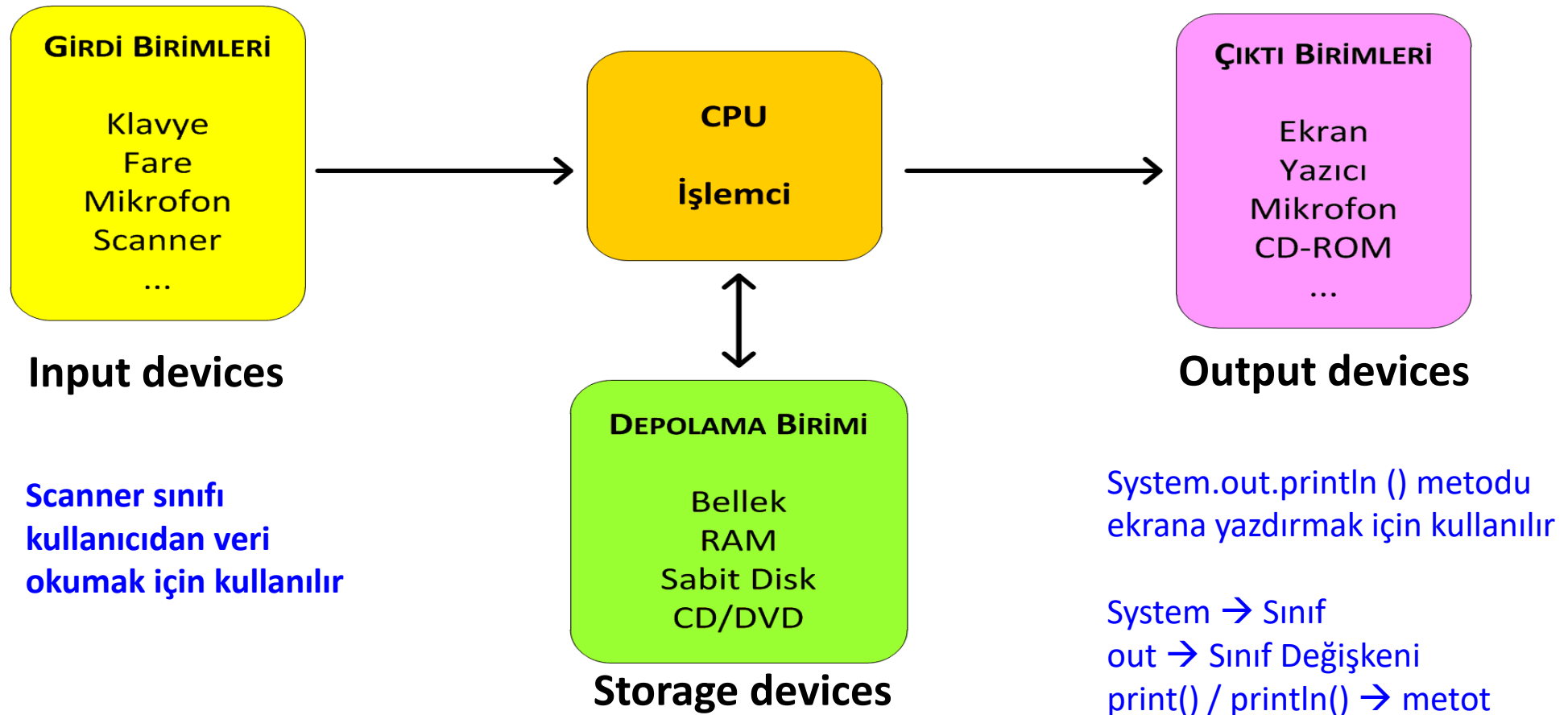
```
01 public class ByteToplam {  
02     public static void main( String[] args ) {  
03         byte sayi1 = 1;  
04         byte sayi2 = 2;  
05         byte toplam = sayi1 + sayi2;  
06     }  
07 }
```

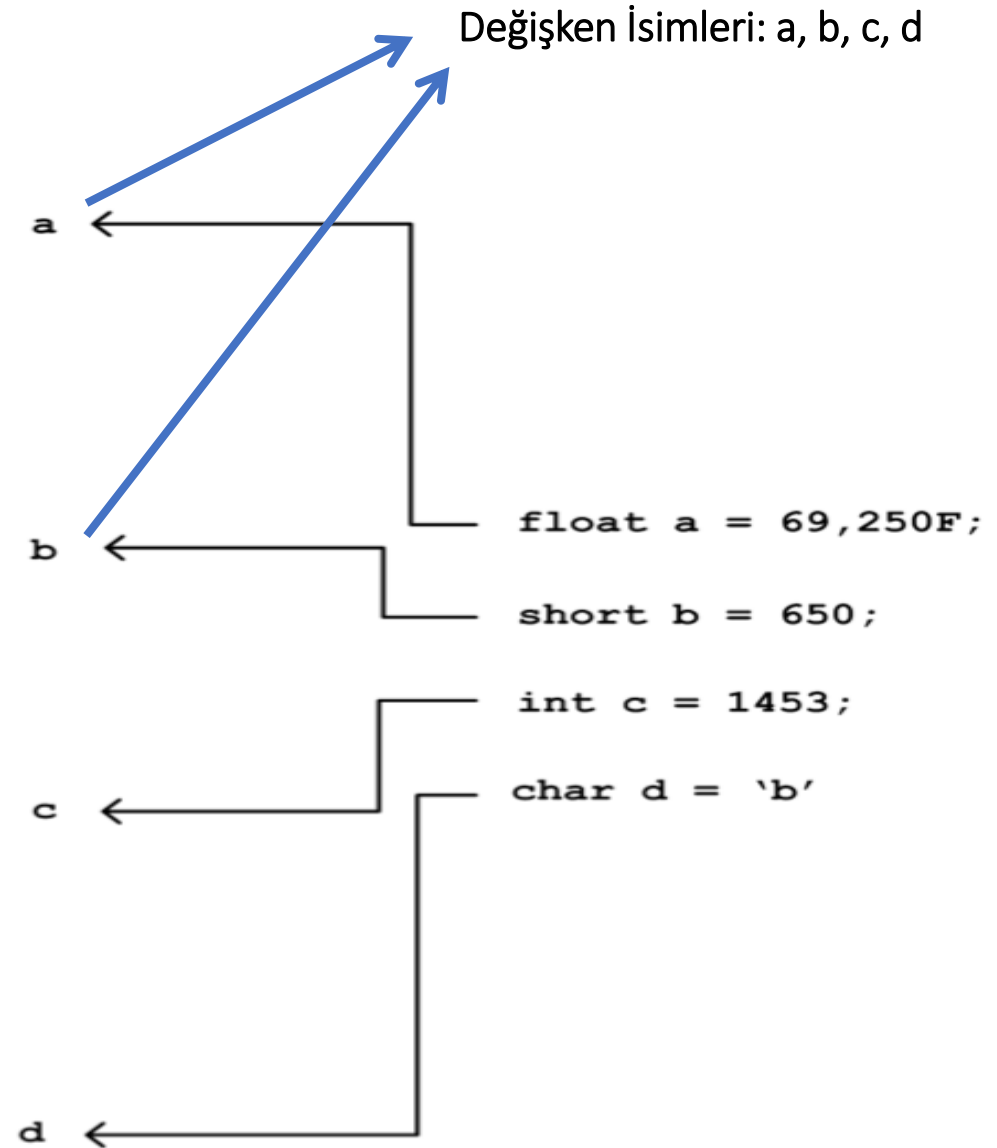
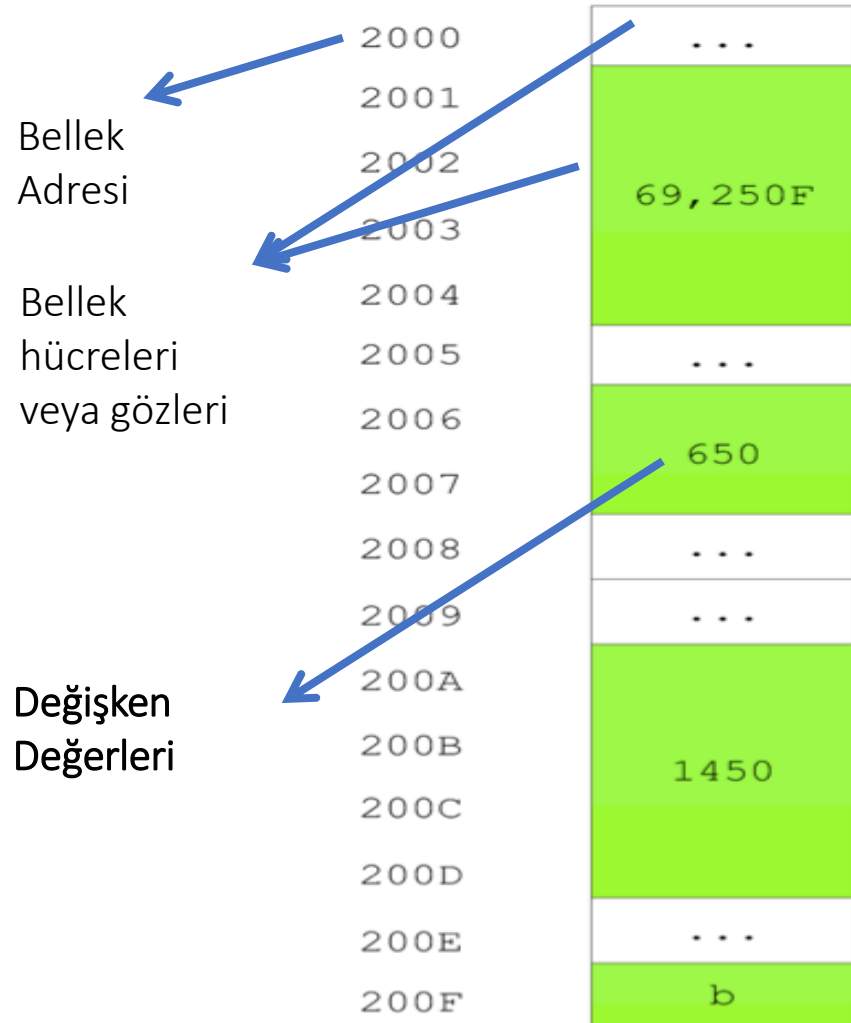
*«ByteToplam programı içindeki tüm değerler [ -128 .. +127 ] aralığında olduğu için sorun çıkmayacağı açıktır.*

*Buna karşın değerler kullanıcıdan alınacak olsaydı, kullanıcının bu menzil dışında bir sayı girmesi durumunda programımızın çalışırken hata nedeni ile işlemi tamamlayamadan sonlanacağına dikkat etmemiz gerekir.*

*Bu durumu önlemek için byte yerine, örneğin short kullandığımızda ise, bellekte daha fazla yer kullanmış oluruz. Eğer gerçekten veriler byte menziline daha büyükse short kullanmak doğru bir işlemdir; fakat küçük sayılar için short kullanırsak da bu kez gereksiz yere fazladan bellek kullanmış oluruz. Dolayısı ile kullanacağımız bir değişkenin sahip olacağı en büyük değeri göz önüne alarak, mümkün olan en dar kapasiteli tamsayı veri türünü kullanmalıyız.»*

# Hatırlayalım: Bellek





Bellek Hücreleri	Değer	Adres	Hücre Adresleri
	10101010	00000000	
	01011111	00000001	
		00000010	
	...	...	
	11110000	00001100	
	00110011	00001101	
	...	...	
	11001100	11111101	
	01010111	11111110	
	10101010	11111111	

0012FF00

...

0012FF65

...

100
...
'A'
...

i

...

ch

...

i ve ch değişkenlerine atanan değerler

Genellikle adresler için on altılı sayı sistemi kullanıyoruz. İkili sayı sistemi veya onlu sayı sistemi de kullanmak mümkün.

# Bellekte ne gerçekleşiyor?

- Kaynak kodunun (.java) yazılması; çalıştırılabilir programın (.class) üretilmesi
- Programın çalıştırılmak üzere ana belleğe yüklenmesi
- Programın çalıştırılması esnasında **değişkenler** için bellekte yer ayrılması
- Programın sonlandırılması ile birlikte bellekteki **program çalıştırılabilir kodu ve değişkenler** için ayrılan yerlerin serbest bırakılması

---

```
int a;
```

Bellekte a değişkeni için göz(ler) ayrılır; 4 byte

```
a = 10;
```

Bu göze 10 değeri kayıt edilir

```
System.out.print (a);
```

a için ayrılan gözdeki 10 değeri ekrana yazılır

---

Komut	İşlem	İsim	Bellek	Adres
<b>03</b> byte adet;	<b>adet</b> değişkenine bellek tahsisi sonrası	adet	00000000	11100101
<b>04</b> adet = 1;	<b>adet</b> değişkenine değer atama sonrası	adet	00000001	11100101
<b>05</b> short yas;	<b>adet ve yas</b> değişkenlerine bellek tahsisi	adet yas	00000001 00000000 00000000	11100101 11100110 11100111
<b>06</b> yas = adet;	<b>yas</b> değişkenine değer atama	adet yas	00000001 00000000 00000001	11100101 11100110 11100111
<b>07</b> int i;	<b>adet, yas, i</b> değişkenlerine bellek tahsisi	adet yas i	00000001 00000000 00000001 00000000 00000000 00000000 00000000 00000000	11100101 11100110 11100111 11101000 11101001 11101010 11101011

# Tek Bir Tamsayı Üzerinde İşlem Yapan (Unary) Operatörler

Operatör	Anlamı	Örnek	Sonuç
+	Pozitif tamsayı	+5	5
-	Negatif tamsayı	-2	-2
++	Bir arttır	2++	3
		++2	3
--	Bir azalt	2--	1
		--2	1

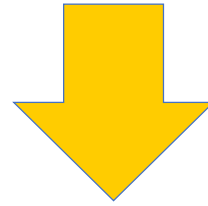
# İki Tamsayı Üzerinde İşlem Yapan (Binary) Operatörler

Operatör	Anlamı	Örnek	Sonuç
+	Toplama	$3 + 5$	8
-	Çıkarma	$1 - 2$	-1
*	Çarpma	$4 * 7$	28
/	Kalansız bölme	$10 / 3$	3
%	Tamsayı bölmeden kalan	$10 \% 3$	1



# Matematiksel bir İfadeyi Java Komutuna Çevirme:

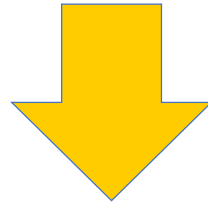
$$\frac{3 + 4x}{5} - \frac{10(y - 5)(a + b + c)}{x} + 9\left(\frac{4}{x} + \frac{9 + x}{y}\right)$$



```
(3 + 4 * x) / 5 - 10 * (y - 5) * (a + b + c) / x +  
9 * (4 / x + (9 + x) / y)
```

# Matematiksel bir İfadeyi Java Komutuna Çevirme:

$$\frac{4}{3(r + 34)} - 9(a + bc) + \frac{3 + d(2 + a)}{a + bd}$$



?

# Operatör Öncelik Sırası

İşlem	Sonuç
$= 2 + 5 - 6 - 2 * 3 + 3$ $= 2 + 5 - 6 - 6 + 3$	-2
$= (2 + 5 - 6) - 2 * (3 + 3)$ $= 1 - 2 * 6$ $= 1 - 12$	-11

Tanımı	Operatör	Öncelik	Yön
Dizi Elemanına Erişim Nesne Üyesine Erişim Metot Çağırma Sonradan Artırma Sonradan Azaltma	[ ] . ( ) ++ --	1	→
Önceden Artırma Önceden Azaltma Artı Sayı Eksi Sayı Olumsuz Mantıksal	++ -- + - !	2	←
Tür Dönüşümü Nesne yaratma	( ) new	3	←
Çarpma Bölme Mod	* / %	4	→
Toplama Çıkarma Metin Birleştirme	+ - 	5	→
6			
Büyüklik – Küçüklük Kontrolü	< <= > >=	7	→
Eşitlik Kontrolü	== !=	8	→
Mantıksal VE	&	9	→
Mantıksal XOR	^	10	→
Mantıksal VEYA		11	→
Koşullu VE	&&	12	→
Koşullu VEYA		13	→
Koşullu Operatör	? :	14	←
Atama	= += -= *= /= %=	15	←

# Kare.java



- Karenin bir kenar (integer) uzunluğunu kullanıcıdan okuyunuz, çevresini ve alanını hesaplayınız, sonra ekrana sonuçları yazdırınız !

```
01 Karenin bir kenar uzunluğunu giriniz => 20
02 Karenin Çevresi => 80
03 Karenin Alanı => 400
```



DO IT  
NOW!

```
01 import java.util.Scanner;
02 public class Kare {
03     public static void main( String[] args ) {
04         Scanner input = new Scanner(System.in);
05
06         System.out.print( "Karenin bir kenar uzunluğunu giriniz=>" );
07         int uzunluk = input.nextInt();
08
09         int a = uzunluk * uzunluk;
10         int c = 4*uzunluk ;
111
12         System.out.println("Karenin Çevresi =>" + c);
13         System.out.println("Karenin Alanı =>" + a);
14     }
15 }
16 }
```

# Java'da Özel Anlamalı Kelimeler

abstract	continue	for	<b>new</b>	switch
assert	default	goto	<b>package</b>	synchronized
boolean	do	if	private	this
break	double	implements	protected	throw
<b>byte</b>	else	<b>import</b>	public	throws
case	enum	instanceof	return	transient
catch	extends	<b>int</b>	<b>short</b>	try
char	final	interface	static	void
class	finally	<b>long</b>	strictfp	volatile
const	float	native	super	while

*Any Questions?*