

İleri Programlama

Tip Dönüşümleri, Aritmetik, İlişkisel ve Mantıksal Operatörler

Hüseyin Ahmetoğlu

Tip Dönüşümleri

- ▶ Java'da, bir int değişkenine double, float veya long değeri atamaya çalışırsanız,
- ▶ "hata: uyumsuz türler: double | float | long'dan int'e olası kayıplı dönüşüm" ile karşılaşacaksınız. Bunun nedeni, kesirli kısmın kesilip kaybolmasıdır.

```
// Assign a "double" value to an "int" variable
double d = 3.5;
int i = d;           //Compilation error: incompatible types: possible lossy conversion from double to int

// Assign a "float" value to an "int" variable
int sum = 55.66f;    //Compilation error: incompatible types: possible lossy conversion from float to int

// Assign a "long" value to an "int" variable
long lg = 123;
int count = lg;      //Compilation error: incompatible types: possible lossy conversion from long to int
```

Tip Dönüşümleri

- ▶ Bir *int* değişkenine bir *double* değer atamak için, *double* işlenen üzerinde çalışmak ve *int*'te kesilmiş bir değer döndürmek için (*int*) *doubleOperand* biçiminde tür atama operatörünü çağırmanız gerekir.
- ▶ Diğer bir deyişle, derleyiciye kesmeyi bilinçli olarak gerçekleştirdiğinizi ve "olası kayıplı dönüşümün" tamamen farkında olduğunuzu söylersiniz. Daha sonra kesilmiş *int* değerini *int* değişkenine atayabilirsiniz.

```
double d = 3.5;
int i;
i = (int)d;    // Cast "double" value of 3.5 to "int" 3. Assign the resultant value 3 to i
               // Casting from "double" to "int" truncates.
```

Type Casting Sözdizimi

```
(type)variable    // e.g., (int)height  
(type)literal    // e.g., (int)55.66
```

- ▶ Açık ve Kapalı olmak üzere iki type casting işlemi vardır.
- ▶ **Explicit type-casting** : Tür atama operatörü ile
- ▶ **Implicit type-casting**: Hassasiyet kaybı yoksa derleyici tarafından yapılır.

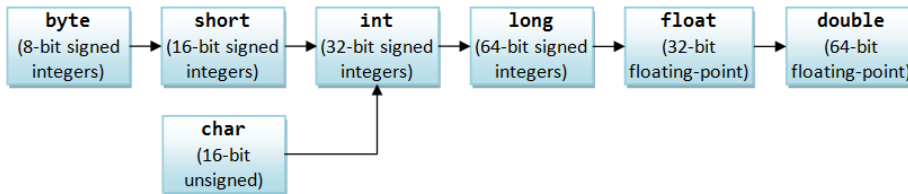
Implicit Type-Casting

- Bir *double* değişkene *int* değeri atarsanız, kesinlik kaybı olmadığından, açık tip dönüşümü gerekli değildir. Derleyici, tür çevirmeyi otomatik olarak gerçekleştirecektir (yani örtük tür çevirme)

```
int i = 3;
double d;
d = i;           // OK, no explicit type casting required
                // d = 3.0
d = (double)i;   // Explicit type casting operator used here
double aDouble = 55; // Compiler auto-casts int 55 to double 55.0
double nought = 0;   // Compiler auto-casts int 0 to double 0.0
                // int 0 and double 0.0 are different.
```

Implicit Type-Casting

- ▶ Aşağıdaki diyagram, derleyici tarafından gerçekleştirilen örtük tür çevirme sırasını gösterir. Kural, genişleyen dönüşüm olarak bilinen hassasiyet kaybını önlemek için küçük türü daha büyük bir türe yükseltmektir.
- ▶ Daralan dönüştürmelerde, derleyiciye olası hassasiyet kaybının farkında olduğunuzu bildirmek için açık tür çevirme gerektirir.
- ▶ Char öğesinin [0, 65535] aralığında bir tamsayı olarak değerlendirildiğini unutmayın.
- ▶ boolean değer herhangi bir tipe dönüştürülemez (yani, boole olmayan değere dönüştürülemez).



Orders of Implicit Type-Casting for Primitives

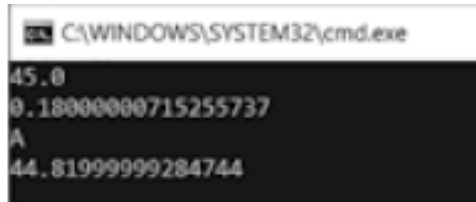
Örnek

```
1  /** Compute the average of running numbers 1 to 100 */
2  public class Average1To100 {
3      public static void main(String[] args) {
4          int sum = 0;
5          double average;
6          for (int number = 1; number <= 100; ++number) {
7              sum += number;      // Final sum is int 5050
8          }
9          average = sum / 100;    // Won't work (average = 50.0 instead of 50.5)
10         System.out.println("Average is " + average); //Average is 50.0
11     }
12 }
```

```
average = (double)sum / 100;    // Cast sum from int to double before division, double / int -> double / double -> double
average = sum / (double)100;    // Cast 100 from int to double before division, int / double -> double / double -> double
average = sum / 100.0;          // int / double -> double / double -> double
average = (double)(sum / 100);  // Won't work. why?
```

Örnek

```
public class Veritipleri {  
    public static void main(String[] args) {  
        float fiyat = 45;  
        // float veri tipindeki değişkene int sayı yerleştirildi  
        System.out.println(fiyat);  
        double kdv = 0.18f;  
        // double veri tipindeki değişkene float sayı yerleştirildi  
        System.out.println(kdv);  
        char text= (char)65;  
        System.out.println(text);  
        double hesap= fiyat - kdv;  
        int say = 0.06f;  
        // Hata!!! int veri tipindeki değişkene float sayı yerleştirilemez  
        System.out.println(hesap);  
    }  
}
```



A screenshot of a Windows command prompt window. The title bar shows the path 'C:\WINDOWS\SYSTEM32\cmd.exe'. The window has a black background with white text. The output of the Java program is displayed as follows:

```
45.0  
0.180000000715255737  
A  
44.81999999284744
```


Örnek

a.

```
(float)34.56767867  ->    34.56768    // float veri tipi
(int)4.3              ->     4          // tam kısım alınır
(int)4.9              ->     4          // tam kısım alınır(yuvarlanmaz!)
```

b.

```
int        sayi = 30;
double     ortalama = sayi/ 4;          // ortalama 7.0 dır,7.5 değildir.
```

Bu örnekte, ortalama değerini kesirli olarak (7.5) elde etmek için **int** veri tipini (*double*) veya **float** veri tipine çevirmeniz gerekir:

```
double ortalama = (double)sayi/ 4; // ortalama=7.5
```

Parse Deyimi

Java dilinde String veri tipini sayısal veri tiplerinden herhangi birine (*Byte*, *Double*, *Float*, *Integer*, *Long*, *Short*) dönüştürmek için **parse** deyimi kullanılır.

Örneğin **String** türdeki bir sayıyı int veri tipine dönüştürmek için:

```
Integer.parseInt(string_degisken)
```

double veri tipine dönüştürmek için:

```
Double.parseDouble(string_degisken)
```

float veri tipine dönüştürmek için ise:

```
Float.parseFloat(string_degisken)
```

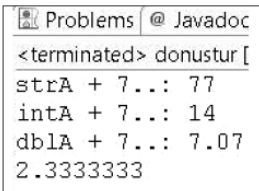
yöntemleri veya metotları kullanılır.

Örneğin:

İşlem	Sonuç
<code>Integer.parseInt("3438")</code>	3438
<code>Double.parseDouble("134.65")</code>	134.65
<code>Float.parseFloat("134.65")</code>	134.65f

Örnek

```
public class donustur {  
    public static void main(String[] args) {  
        int A = 7;  
        // A sayısı string veri tipine dönüştürüldü  
        String strA = A + "";  
        System.out.println("strA + 7..: " + strA + 7);  
        // string veri tipi, int veri tipine dönüştürüldü  
        int intA = Integer.parseInt(strA);  
        System.out.println("intA + 7..: " + (intA + 7));  
        double dblA = Double.parseDouble(strA);  
        // string veri tipi, double veri tipine dönüştürüldü  
        System.out.println("dblA + 7..: " + dblA + 7);  
        float ortalama = (float) intA / 3;  
        System.out.println(ortalama);  
    }  
}
```



```
<terminated> donustur [  
strA + 7..: 77  
intA + 7..: 14  
dblA + 7..: 7.07  
2.3333333
```

Programın ekran çıktısı yandaki gibidir:

Aritmetiksel Operatörler

İşlemler	Aritmetik Operatör	Matematiksel gösterimi	JAVA dili gösterimi
Toplama	+	$X+Y$	$X + Y$
Çıkartma	-	$X-Y$	$X - Y$
Çarpma	*	$(XY),(X.Y),(X*Y)$	$X*Y$
Bölme	/	X/Y ve $\frac{X}{Y}$	X / Y
Üs alma	Yoktur	3^2	$3*3$ <code>Math.pow(3,2);</code>
Karekök alma	Yoktur	$\sqrt{3}$	<code>Math.sqrt(3);</code>
Mod Alma (Kalan)	%	$X \text{ Mod } Y$	$X \% Y$
String Birleştirme	+	BadeSare	<code>"Bade" + "Sare"</code>
Negatif	-	$-Y$	$-Y$
Değer Aktarma	=	$Y \Rightarrow X$	$X=Y$

Java'da üs ve karekök almak için özel bir operatör yoktur. Bunun yerine Math sınıfının `pow()` ve `sqrt()` yöntemleri kullanılır. Üs almak için `Math.pow()`, karekök almak için `Math.sqrt()` kullanılır. Örneğin b^2 işlemi `Math.pow(b,2)` şeklinde ifade edilir.

Örnek

$$z = - \frac{5 \cdot a^{2/3}}{4}$$

1. şeklindeki matematiksel ifadenin Java dilinde kodlaması nasıl yapılır?

Kodlanması: `z = - (5*Math.pow(a, (2/3)) / 4);`

2. a=9 için z değerini bulan programı yazınız.

```
public class AritmetikselOperatorler {  
    public static void main(String[] args) {  
        int a = 9;  
        double z = -(5 * Math.pow(a, (2 / 3)) / 4);  
        System.out.println("z = " + z);  
    }  
}
```

Bileşik Atama Operatörleri

Operation	Mode	Usage	Example
=	Binary	<i>var = expr</i>	<i>x = 5;</i>
+=	Binary	<i>var += expr</i> same as: <i>var = var + expr</i>	<i>x += 5;</i> same as: <i>x = x + 5</i>
-=	Binary	<i>var -= expr</i> same as: <i>var = var - expr</i>	<i>x -= 5;</i> same as: <i>x = x - 5</i>
*=	Binary	<i>var *= expr</i> same as: <i>var = var * expr</i>	<i>x *= 5;</i> same as: <i>x = x * 5</i>
/=	Binary	<i>var /= expr</i> same as: <i>var = var / expr</i>	<i>x /= 5;</i> same as: <i>x = x / 5</i>
%=	Binary	<i>var %= expr</i> same as: <i>var = var % expr</i>	<i>x %= 5;</i> same as: <i>x = x % 5</i>

Bileşik Atama Operatörleri

```
byte b1 = 5, b2 = 8, b3;  
b3 = (byte)(b1 + b2);    // byte + byte -> int + int -> int, need to explicitly cast back to "byte"  
b3 = b1 + b2;           // error: RHS is int, cannot assign to byte  
b1 += b2;               // implicitly casted back to "byte"  
  
char c1 = '0', c2;  
c2 = (char)(c1 + 2);     // char + int -> int + int -> int, need to explicitly cast back to "char"  
c2 = c1 + 2;            // error: RHS is int, cannot assign to char  
c1 += 2;                // implicitly casted back to "char"
```

Arttırma / Azaltma

Operator	Mode	Usage	Description
++ (Increment)	Unary Prefix Unary Postfix	++x x++	Increment the value of the operand by 1. x++ or ++x is the same as x += 1 or x = x + 1
-- (Decrement)	Unary Prefix Unary Postfix	--x x--	Decrement the value of the operand by 1. x-- or --x is the same as x -= 1 or x = x - 1

Example

```
int x = 5;  
x++; // x is 6  
++x; // x is 7
```

```
int y = 6;  
y--; // y is 5  
--y; // y is 4
```

```
int x = 5;  
// 4 ways to increment by 1  
x = x + 1; // x is 6  
x += 1;    // x is 7  
x++;       // x is 8  
++x;       // x is 9  
// 4 ways to decrement by 1  
x = x - 1; // x is 8  
x -= 1;    // x is 7  
x--;       // x is 6  
--x;       // x is 5
```


Arttırma / Azaltma

Operator	Example	Same As
++var (Pre-Increment)	y = ++x;	x = x + 1; y = x;
var++ (Post-Increment)	y = x++;	oldX = x; x = x + 1; y = oldX;
--var (Pre-Decrement)	y = --x;	x = x - 1; y = x;
var-- (Post-Decrement)	y = x--;	oldX = x; x = x - 1; y = oldX;

```
// Two operations in the statement: increment and assignment
x = 5;
y = ++x; // Increment x (=6), then assign x to y (=6). (++x returns x+1)
x = 5;
y = x++; // Assign x to y (=5), then increment x (=6). (x++ returns the oldX)
// After the operations, x gets the SAME value, but the other operation has different outcomes

// Two operations in the statement: increment and println()
x = 5;
System.out.println(++x); // Increment x (=6), then print x (=6). (++x returns x+1)
x = 5;
System.out.println(x++); // Print x (=5), then increment x (=6). (x++ returns the oldX)
```

İlişkisel ve Mantıksal Operatörler

Operator	Mode	Usage	Description	Example (x=5, y=8)
==	Binary	x == y	Equal to	(x == y) ⇒ false
!=	Binary	x != y	Not Equal to	(x != y) ⇒ true
>	Binary	x > y	Greater than	(x > y) ⇒ false
>=	Binary	x >= y	Greater than or equal to	(x >= 5) ⇒ true
<	Binary	x < y	Less than	(y < 8) ⇒ false
<=	Binary	x <= y	Less than or equal to	(y <= 8) ⇒ true

Operator	Mode	Usage	Description
!	Unary	!x	Logical NOT
&&	Binary	x && y	Logical AND
	Binary	x y	Logical OR
^	Binary	x ^ y	Logical Exclusive-OR (XOR)

İlişkisel ve Mantıksal Operatörler

NOT (!)	true	false
Result	false	true

AND (&&)	true	false
true	true	false
false	false	false

OR ()	true	false
true	true	true
false	true	false

XOR (^)	true	false
true	false	true
false	true	false

İlişkisel ve Mantıksal Operatörler

```
// Return true if x is between 0 and 100 (inclusive)
(x >= 0) && (x <= 100)
// wrong to use 0 <= x <= 100

// Return true if x is outside 0 and 100 (inclusive)
(x < 0) || (x > 100)
// or
!((x >= 0) && (x <= 100))

// Return true if year is a leap year
// A year is a leap year if it is divisible by 4 but not by 100, or it is divisible by 400.
((year % 4 == 0) && (year % 100 != 0)) || (year % 400 == 0)
```

İlişkisel ve Mantıksal Operatörler

```
/**
 * Test relational and logical operators
 */
public class RelationalLogicalOpTest {
    public static void main(String[] args) {
        int age = 18;
        double weight = 71.23;
        int height = 191;
        boolean married = false;
        boolean attached = false;
        char gender = 'm';

        System.out.println(!married && !attached && (gender == 'm')); //true
        System.out.println(married && (gender == 'f')); //false
        System.out.println((height >= 180) && (weight >= 65) && (weight <= 80)); //true
        System.out.println((height >= 180) || (weight >= 90)); //true
    }
}
```

Örnek

- Yıl, ay (1-12) ve gün (1-31) verildiğinde, 15 Ekim 1582'den önceki tarihler için true değerini döndüren bir boole ifadesi yazın (Gregoryen takvimi kesme tarihi).

```
(year < 1582) || (year == 1582 && month < 10) || (year == 1582 && month == 10 && day < 15)
```


Mantıksal Operatör Önceliği

- En yüksekten en düşüğe doğru öncelik şudur: '!' (tekli), '^', '&&', '||'. Ancak şüphe duyduğunuzda parantez kullanın!

```
System.out.println(true || true && false);    //true (same as below)
System.out.println(true || (true && false));  //true
System.out.println((true || true) && false);  //false

System.out.println(false && true ^ true);     //false (same as below)
System.out.println(false && (true ^ true));   //false
System.out.println((false && true) ^ true);   //true
```


Kısa Devre İşlemleri

- ▶ İkili AND (&&) ve OR (||) operatörleri, kısa devre operatörleri olarak bilinir; bu, sonuç **sol** operand tarafından belirlenebiliyorsa **sağ** operandın değerlendirilmeyeceği anlamına gelir. Örneğin, `false && rightOperand`, `false` cevabı dönecektir
- ▶ `true || rightOperand`, ise `rightOperand` değerlendirmeden `true` verir.
- ▶ Belirli işlemleri gerçekleştirmek için doğru işlenene güvenirseniz, bunun olumsuz sonuçları olabilir, örn. `false && (++ i < 5)` ancak `++ i` değerlendirilmeyecektir.

String ve '+' Birleştirme Operatörü

- Java'da '+' özel bir operatördür. Aşırı yüklenmiş. Aşırı yükleme, işlenenlerinin türlerine bağlı olarak farklı işlemler gerçekleştirdiği anlamına gelir.

```
1 + 2 ⇒ 3    // int + int ⇒ int
1.2 + 2.2 ⇒ 3.4 // double + double ⇒ double
1 + 2.2 ⇒ 1.0 + 2.2 ⇒ 3.2 // int + double ⇒ double + double ⇒ double
'0' + 2 ⇒ 48 + 2 ⇒ 50    // char + int ⇒ int + int ⇒ int (need to cast back to char '2')
```

```
"Hello" + "world" ⇒ "Helloworld"
"Hi" + ", " + "world" + "!" ⇒ "Hi, world!"
```

String ve '+' Birleştirme Operatörü

```
"The number is " + 5 ⇒ "The number is " + "5" ⇒ "The number is 5"  
"The average is " + average + "!" (suppose average=5.5) ⇒ "The average is " + "5.5" + "!" ⇒ "The average is 5.5!"  
"How about " + a + b (suppose a=1, b=1) ⇒ "How about 1" + b ⇒ "How about 11" (left-associative)  
"How about " + (a + b) (suppose a=1, b=1) ⇒ "How about " + 2 ⇒ "How about 2"
```

```
System.out.println("The sum is: " + sum); // Value of "sum" converted to String and concatenated  
System.out.println("The square of " + input + " is " + squareInput);
```

Escape Characters

String sabitlerin ekranda gösteriminde bazı çıkış karakterleri kullanılır. Bunlar:

Basılacak Karakter veya İşlevi	Çıkış	Örnek kod Karakteri	Ekrana Çıktısı
' (Tek Tırnak)	\'	"Bade\'nin Evi"	Bade'nin Evi
" (Çift Tırnak)	\"	"Bade dedi \"Selam\" "	Bade dedi "Selam"
\ (Backslash)	\\	"C:\\"	C:\
Bir alt satıra geç	\n	"1\n2"	1 2
Bir tab boşluk	\t	"1\t2"	1 2
backspace tuşu	\b	"12\b3"	13
Enter tuşuna basıldı	\r	"Bade\rZehra"	Bade Zehra
[16 bit unicode] Hexadecimal sayının ASCII karşılığı*	\uxxxx	"61 sayısı: \u0061"	61 sayısı: a

* Unicode karakterlerin karşılıklarına www.unicode.org adresinden ulaşabilirsiniz.

Operatörlerin İşlem Sıraları

Öncelik Sırası	İşlemler	İşlem Simgesi
1	Sayıları negatifleştirme	-
2	Parantez içi	(...)
3	Matematiksel fonksiyonlar	Sin, Cos, v.b.
4	Son artırma/azaltma	<ifade>++, <ifade>--
5	Ön Artırma/Azaltma	++<ifade>, --<ifade>, !<ifade>
6	Çarpma, Bölme veya Kalan	*, /, %
7	Toplama veya Çıkarma	+ veya -
8	Kaydırma	<< veya >>, >>>
9	Karşılaştırma	<, <=, >, >=, ==, !=
10	Bit düzeyinde AND, XOR, OR	&, ^,
11	Mantıksal AND, OR	&&,
12	Atama	=, +=, -=, *=, /=, %=
13	Bitsel Atama	>>=, <<=, >>>=
14	Boolean atama	&=, ^=, =

Örnek 2.19: $\frac{a^2 + bc}{2ab}$ şeklindeki matematiksel ifadenin bilgisayar ortamındaki kodlanması aşağıdaki şekilde olur.

Kodlanması: (a*a +b*c) / (2*a*b)

Örnek

Matematiksel ifade:: $y = pr^q + w/x - z$

JAVA dilinde kodlaması: ...: $y = p * \text{Math.pow}(r, q) + w / x - z;$

İşlem önceliğini dikkate alarak, harflerin yerine rakam yazarak matematiksel ifadeyi test edersek:

İşlem önceliği

6 2 1 4 3 5

$$y = 3 * 4^2 + 6 / 2 - 7$$
$$y = 3 * 16 + 6 / 2 - 7$$
$$y = 48 + 6 / 2 - 7$$
$$y = 48 + 3 - 7$$
$$y = 51 - 7$$
$$y = 44$$

sonucunu elde ederiz.

Burada işlem önceliği; ilk önce üs alma, daha sonra çarpma (solda olduğu için), bölme, toplama ve çıkarma şeklindedir.

Örnek 2.22:

$z = -\frac{5a^{2/3}}{\sqrt{x^2 - y^2}}$ şeklindeki matematiksel ifadenin JAVA dilinde kodlanması aşağıdaki şekilde olur.

Kodlanması: $z = - (5 * \text{Math.pow}(a, (2/3))) / (\text{Math.sqrt}(x * x - y * y));$

Örnek

- a. $\left| \frac{x-3}{2\pi y} \right|$ Kodlanması: `Math.abs((x-3)/(2*Math.PI*y))`
- b. $b \leq a \leq 2$ Kodlanması: `(a>=b) && (a <= 2.0)`

Örnek 2.25: $c = \sqrt{a+b^2}$ şeklindeki matematiksel ifadeyi Java dilinde nasıl kodlarsınız?

Cevap: `c=Math.sqrt(a)+Math.pow(b)`

Örnek 2.26:

$\frac{-B + \sqrt{B^2 - 4AC}}{2A}$ şeklindeki matematiksel ifadenin Java dilinde kodlanmasını yazınız?

Cevap: Kodlanması: `(-B + Math.sqrt(B*B - 4*A*C)) / 2*A`

Örnek 2.27: $y=4x^2+2x+5$ şeklindeki matematiksel ifadenin Java dilinde kodlanmasını yazınız?

Kodlanması: `y=4*x*x+2*x+5`

Örnek 2.28:

$$f = \begin{cases} x^2+1, & x \leq 0, \\ x, & x \geq 1, \\ \sqrt{x}, & 0 < x < 1, \end{cases}$$
 şeklinde verilen matematiksel fonksiyonun Java dilinde kodlanmasını yazınız.

Kodlanması:

```
if (x<=0) { f=Math.pow(x,2)+1; }
if (x>=1) { f=x; }
if (x<1 && x>0) {f=Math.sqrt(x); }
```

KAYNAKLAR

- ▶ Programming Notes. (2021, March 11). Retrieved from <https://www3.ntu.edu.sg/home/ehchua/programming/index.html>
- ▶ Çobanoğlu B. (2020) Java ile Programlama ve Veri Yapıları. İstanbul Pusula Yayıncılık. 978-605-2359-84-6