Nesne Yönelimli Programlama

Hazırlayan: M.Ali Akcayol
Gazi Üniversitesi
Bilgisayar Mühendisliği Bölümü

Not: Bu dersin sunumları, “Java Programlama Dili ve Yazılım Tasarımı, Altuğ B. Altıntaş, Papatya Yayıncılık, 2016” kitabı kullanılarak hazırlanmıştır.

Konular

- Atama İşlemleri
- Yordamların Çağrılması
- Java Operatörleri
  - Aritmetik Operatörler
  - İlişkisel Operatörler
  - Mantıksal Operatörler
  - Bit Düzeyinde Operatörler
  - Atama Operatörleri
  - String Operatörü
  - Nesnelerin Karşılaştırılması

Atama İşlemleri

- Java programlama dilinde **veri türleri ve nesneler üzerinde işlem yapmak için operatörler** kullanılır.
- Operatörlerin büyük bölümü hemen hemen tüm programlama dillerinde benzerdir.
- En **yaygın** kullanılan operatörler **toplama (+)** ve **çıkartma (-)** operatörleridir.
- Aynı ifade içinde yer alan **operatörlerin kendi aralarında öncelik sıralaması vardır.**
- Değer atamalarında **sağ taraftaki değer sol taraftaki değişkene atanır.**

```
int a;   // Tanımlama
a = 4;   // Doğru atama
4 = a;   // Yanlış atama
```

3

Atama İşlemleri

Temel türlerde atama

- Atama işlemi, **temel (primitive) türler** için basittir.
- Temel türdeki değişkeni diğerine atadığımızda **sadece içerikler değişir.**

```
package oop;

import java.util.*;

public class JavaProgramlama {

    public static void main(String [] args){

        int a, b;
        a = 4;
        b = 5;
        a = b ;

        System.out.println("a = " + a + ", b = " + b);
    }
}
```

```
a = 5, b = 5
```

4

Atama İşlemleri

Nesneler ve atamalar

- Nesneler için atama işlemleri, temel türlere göre karmaşıktır.
- **Nesneleri yönetmek için referans (adres) kullanılır.**
- **Nesnelerde atama işleminde** referansın gösterdiği hedefte **(adreste) değişiklik olur.**

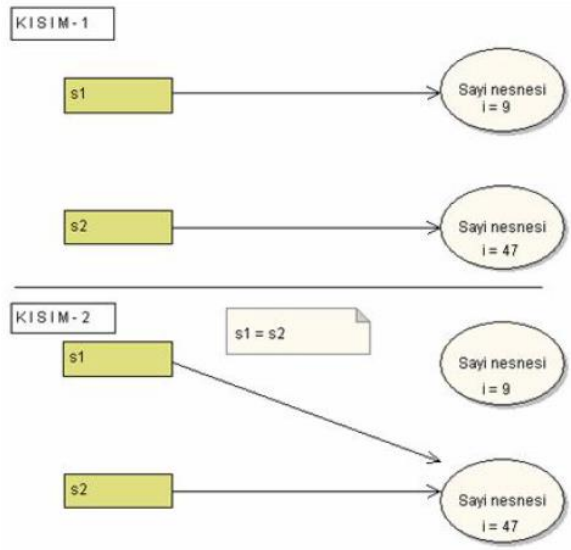
```
class Sayi {
    int i;
}

public class JavaProgramlama {

    public static void main(String [] args){
        Sayi s1 = new Sayi();
        Sayi s2 = new Sayi();
        s1.i = 9;
        s2.i = 47;
        System.out.println("1: s1.i: " + s1.i +", s2.i: " + s2.i);
        s1 = s2; // referanslar kopyalaniyor.. nesneler degil
        System.out.println("2: s1.i: " + s1.i +", s2.i: " + s2.i);
        s1.i = 27;
        System.out.println("3: s1.i: " + s1.i +", s2.i: " + s2.i);
    }
}
```

```
1: s1.i: 9, s2.i: 47
2: s1.i: 47, s2.i: 47
3: s1.i: 27, s2.i: 27
```

5

Atama İşlemleri

Nesneler ve atamalar



6

Atama İşlemleri

Nesneler ve atamalar

- Nesneler arasında atama yerine, **nesnelerin değişkenleri arasında atama yapılırsa referans farklı olur.**

```
public class JavaProgramlama {

    public static void main(String [] args){
        Sayi s1 = new Sayi();
        Sayi s2 = new Sayi();
        s1.i = 9;
        s2.i = 47;
        System.out.println("1: s1.i: " + s1.i +", s2.i: " + s2.i);
        s1.i = s2.i;// değerler kopyalaniyor.. referanslar degil
        System.out.println("2: s1.i: " + s1.i +", s2.i: " + s2.i);
        s1.i = 27;
        System.out.println("3: s1.i: " + s1.i +", s2.i: " + s2.i);
    }
}
```

```
1: s1.i: 9, s2.i: 47
2: s1.i: 47, s2.i: 47
3: s1.i: 27, s2.i: 47
```

7

Konular

- Atama İşlemleri
- Yordamların Çağrılması
- Java Operatörleri
  - Aritmetik Operatörler
  - İlişkisel Operatörler
  - Mantıksal Operatörler
  - Bit Düzeyinde Operatörler
  - Atama Operatörleri
  - String Operatörü
  - Nesnelerin Karşılaştırılması

Yordamların Çağrılması

- **Yordamlar parametre kabul ederler** ve bu parametreleri alarak işlemler gerçekleştirir.

```
1: x.c: a
2: x.c: z
```

```
package oop;

import java.util.*;

class Harf {
    char c;
}

public class JavaProgramlama{
    static void f(Harf h) {
        /* Harf nesnesine yeni bir referans bağlandı (h),
           yoksa oluşturulan Harf nesnesinin veya yeni
           bir Harf nesnesinin bu yordama gönderilmesi gibi
           birşey söz konusu değildir. */
        h.c = 'z';
    }

    public static void main(String[] args) {
        Harf x = new Harf();     // Harf nesnesini oluşturuluyor.
        x.c = 'a';               // Harf nesnesinin c alanına değer atandı
        System.out.println("1: x.c: " + x.c);
        f(x);                    //dikkat
        System.out.println("2: x.c: " + x.c);
    }
}
```

9

Atama İşlemleri

- Yordam çağırmaları temel türler için daha kolaydır.

```
a = 5 olarak gönderildi.
a = 10.0
a = 5.0
```

```
package oop;

import java.util.*;

public class JavaProgramlama{
    static void f(double a) {
        System.out.println("a = 5 olarak gönderildi.");
        a = 10;      // alınan parametrenin değeri 10'a eşitlendi
        System.out.println("a = " + a);
    }

    public static void main(String[] args){
        double a = 5;
        f(a);
        System.out.println("a = " + a ); // buradaki a'nın değeri
    }
}
```

10

Konular

- Atama İşlemleri
- Yordamların Çağrılması
- Java Operatörleri
  - Aritmetik Operatörler
  - İlişkisel Operatörler
  - Mantıksal Operatörler
  - Bit Düzeyinde Operatörler
  - Atama Operatörleri
  - String Operatörü
  - Nesnelerin Karşılaştırılması

Java Operatörleri

- **Operatörler**, programlama dillerindeki **işlem yapma yeteneğine sahip simgelerdir.**
- **Bir işlem bir operatör ile veya bir grup operatörün bir araya getirilmesiyle yapılabilir.**
- **Bir işlem için yordam (method) yazılması da gerekebilir.**
- Java dili oldukça zengin ve esnek operatör kümesine sahiptir:
  - **Aritmetik operatörler**
  - **İlişkisel operatörler**
  - **Mantıksal operatörler**
  - **Bir düzeyinde (bitwise) operatörler**
  - **Atama operatörleri**
  - **String operatörü**

12

Java Operatörleri

- Java dilinde **operatörler**, **ön ek**, **son ek** veya **ara ek** olarak kullanılabilir.

```
operatör değişken   //ön-ek ifadesi

değişken operatör   // son-ek ifadesi

değişken1 operatör değişken2   //ara-ek

değişken1 ? değişken2 : değişken3   //ara ek
```

13

Konular

- Atama İşlemleri
- Yordamların Çağrılması
- Java Operatörleri
  - Aritmetik Operatörler
  - İlişkisel Operatörler
  - Mantıksal Operatörler
  - Bit Düzeyinde Operatörler
  - Atama Operatörleri
  - String Operatörü
  - Nesnelerin Karşılaştırılması

Aritmetik Operatörler

- Java dili kayan-noktalı (floating-point) sayılar ve tamsayılar (integer) için aritmetik işlemleri destekleyen operatörlere sahiptir.
- Bu işlemler, toplama operatörü **(+)**, çıkartma operatörü **(-)**, çarpma operatörü **(*)**, bölme operatörü **(/)** ve mod ile bölme **(%)** operatörüdür.

Operatör	Kullanılış	Açıklama
+	değişken1 + değişken2	değişken1 ile değişken2'yi toplar
-	değişken1 - değişken2	değişken1 ile değişken2'yi çıkarır
*	değişken1 * değişken2	değişken1 ile değişken2'yi çarpar
/	değişken1 / değişken2	değişken1, değişken2 tarafından bölünür
%	değişken1 % değişken2	değişken1'in değişken2 tarafından bölümünden kalan hesaplanır.

Aritmetik Operatörler

Örnek

```
public class JavaProgramlama {
      public static void main(String[] args) {

        // Değişkenler atanan değerler
        int a = 57, b = 42;
        double c = 27.475,  d = 7.22;
        System.out.println("Degisken Degerleri...");
        System.out.println(" a = " + a);
        System.out.println(" b = " + b);
        System.out.println(" c = " + c);
        System.out.println(" d = " + d);

        // Sayıları topluyoruz
        System.out.println("Toplama...");
        System.out.println(" a + b = " + (a + b));
        System.out.println(" c + d = " + (c + d));

        // Sayıları çıkartıyoruz
        System.out.println("Cikartma...");
        System.out.println(" a - b = " + (a - b));
        System.out.println(" c - d = " + (c - d));
```

```
Degisken Degerleri...
 a = 57
 b = 42
 c = 27.475
 d = 7.22
Toplama...
 a + b = 99
 c + d = 34.695
Cikartma...
 a - b = 15
 c - d = 20.255000000000003
Carpma...
 a * b = 2394
 c * d = 198.36950000000002
Bolme...
 a / b = 1
 c / d = 3.805401662049862
Kalan sayiyi hesaplama...
 a % b = 15
 c % d = 5.815000000000002
Karisik tipler...
 b + d = 49.22
 a * c = 1566.075
```

Aritmetik Operatörler

Örnek

```
    // Sayıları çarpıyoruz.
    System.out.println("Carpma...");
    System.out.println(" a * b = " + (a * b));
    System.out.println(" c * d = " + (c * d));

    // Sayıları bölüyoruz
    System.out.println("Bolme...");
    System.out.println(" a / b = " + (a / b));
    System.out.println(" c / d = " + (c / d));

    // Bölme işlemlerinden kalan sayıyı hesaplıyoruz
    System.out.println("Kalan sayiyi hesaplama...");
    System.out.println(" a % b = " + (a % b));
    System.out.println(" c % d = " + (c % d));

    // double ve int tiplerini karışık şekilde kullanıyoruz.
    System.out.println("Karisik tipler...");
    System.out.println(" b + d = " + (b + d));
    System.out.println(" a * c = " + (a * c));
  }
}
```

```
Degisken Degerleri...
 a = 57
 b = 42
 c = 27.475
 d = 7.22
Toplama...
 a + b = 99
 c + d = 34.695
Cikartma...
 a - b = 15
 c - d = 20.255000000000003
Carpma...
 a * b = 2394
 c * d = 198.36950000000002
Bolme...
 a / b = 1
 c / d = 3.805401662049862
Kalan sayiyi hesaplama...
 a % b = 15
 c % d = 5.815000000000002
Karisik tipler...
 b + d = 49.22
 a * c = 1566.075
```


Aritmetik Operatörler

- Aritmetik operatörler **işlem sonucu değiştirecek dönüştürme yaparlar.**

Sonuç Veri Tipi	Değişkenlerin Veri Tipleri
long	Değişkelerin float veya double tipinden farklı olması ve en az bir değişkenin long tipinde olması
int	Değişkelerin float veya double tipinden farklı olması ve değişkenlerin long tipinden farklı olması
double	En az bir değişkenin double tipinde olması
float	Değişkenlerin hiçbirinin double tipinde olmaması ve değişkenlerden en az birinin float tipinde olması

- Toplama ve çıkarma operatörleri de tür dönüştürme yapar.

Operatör	Kullanılış Şekli	Açıklama
+	+ değişken	Eğer değişken char, byte veya short tipinde ise int tipine dönüştürür
-	- değişken	Değişkenin değerini eksi yapar (-1 ile çarpar).

Aritmetik Operatörler

Örnek

```
package oop;

import java.util.*;

public class JavaProgramlama {
	public static void main(String[] args) {
		char kr = 'a' ;
		int b = +kr ;		// otomatik olarak int temel tipine çevrildi
		int c = -b ;		// değeri eksi yaptı
		System.out.println("kr = " + kr );
		System.out.println("b = " + b );
		System.out.println("c = " + c );
	}
}
```

```
kr = a
b = 97
c = -97
```

19

Aritmetik Operatörler

Örnek

- Dönüştürme işlemlerinde değer kaybı olabilir.

```
package oop;

import java.util.*;

public class JavaProgramlama {
    public static void main(String args[]) {
        int a = 5;
        double b = (double) a;
        double x = 4.15 ;
        int y = (int) x ;
        long z = (long) y ;
        System.out.println("b = " + b + " y = " + y + " z = " + z);
    }
}
```

```
b = 5.0 y = 4 z = 4
```

20

Aritmetik Operatörler

Bir artırma ve azaltma

- Bu operatörler değişkenin içeriğini bir arttırmak veya azaltmak için kullanılır.
- **Bir arttırma** için **++** ve **bir azaltma** için **--** operatörleri kullanılır.
- **Ön-ek (prefix):** (--) veya (++) operatörünün kullanılan değişkenin önüne gelmesini ifade eder.
- **Son-ek (postfix):** (--) veya (++) operatörünün değişkenin sonuna gelmesini ifade eder.

Operatör	Kullanılış Şekli	Açıklama
++	değişken++	Önce değişkenin değerini hesaplar sonra değişkenin değerini bir arttırır.
++	++değişken	Önce değişkenin değerini arttırır sonra değişkenin değerini hesaplar.
--	değişken--	Önce değişkenin değerini hesaplar sonra değişkenin değerini bir azaltır.
--	--değişken	Önce değişkenin değerini azaltır sonra değişkenin değerini hesaplar.

21

Aritmetik Operatörler

Örnek

```
package oop;

import java.util.*;

public class JavaProgramlama {
    static void ekranaYaz(String s) {
        System.out.println(s);
    }

    public static void main(String[] args) {
        int i = 1;
        ekranaYaz("i : " + i);
        ekranaYaz("++i : " + ++i);      // önek artırım
        ekranaYaz("i++ : " + i++);      // sonek artırım
        ekranaYaz("i : " + i);
        ekranaYaz("--i : " + --i);      // önek azaltma
        ekranaYaz("i-- : " + i--);      // sonek azaltma
        ekranaYaz("i : " + i);
    }
}
```

```
i : 1
++i : 2
i++ : 2
i : 3
--i : 2
i-- : 2
i : 1
```

22

Konular

- Atama İşlemleri
- Yordamların Çağrılması
- Java Operatörleri
  - Aritmetik Operatörler
  - İlişkisel Operatörler
  - Mantıksal Operatörler
  - Bit Düzeyinde Operatörler
  - Atama Operatörleri
  - String Operatörü
  - Nesnelerin Karşılaştırılması

İlişkisel Operatörler

- İlişkisel operatörler **iki değeri karşılaştırarak** bunların arasındaki **mantıksal ilişkiyi belirlemeye yarar.**
- **İki değer birbirine eşit değilse**, == operatörüyle bu ilişki sonucu **yanlış (false)** olur, **eşitse doğru (true)** olur.

Operatör	Kullanılış Şekli	True değeri döner eğer ki.....
>	değişken1 > değişken2	değişken1, değişken2'den büyükse
>=	değişken1 >= değişken2	değişken1, değişken2'den büyükse veya eşitse
<	değişken1 < değişken2	değişken1, değişken2'den küçükse
<=	değişken1 <= değişken2	değişken1, değişken2'den küçükse veya eşitse
==	değişken1 == değişken2	değişken1, değişken2'ye eşitse
!=	değişken1 != değişken2	değişken1, değişken2'ye eşit değilse

24

İlişkisel Operatörler

Örnek

```
package oop;

import java.util.*;

public class JavaProgramlama {
    public static void main(String[] args) {
        // değişken bildirimleri
        int i = 37, j = 42, k = 42;
        System.out.println("Degisken degerleri...");
        System.out.println(" i = " + i);
        System.out.println(" j = " + j);
        System.out.println(" k = " + k);

        //Büyüktür
        System.out.println("Buyuktur...");
        System.out.println(" i > j = " + (i > j));  //false - i, j  den küçüktür
        System.out.println(" j > i = " + (j > i));  //true - j, i den Büyüktür
        System.out.println(" k > j = " + (k > j));  //false - k, j ye eşit

        //Büyüktür veya eşittir
        System.out.println("Buyuktur veya esittir...");
        System.out.println(" i >= j = " + (i >= j)); //false - i, j  den küçüktür
        System.out.println(" j >= i = " + (j >= i)); //true - j, i den büyüktür
        System.out.println(" k >= j = " + (k >= j)); //true - k, j'ye eşit
```


İlişkisel Operatörler

Örnek

```
        //Küçüktür
        System.out.println("Kucuktur...");
        System.out.println(" i < j = " + (i < j));    //true - i, j'den küçüktür
        System.out.println(" j < i = " + (j < i));    //false - j, i' den büyüktür
        System.out.println(" k < j = " + (k < j));    //false - k, j'ye eşit

        //Küçüktür veya eşittir
        System.out.println("Kucuktur veya esittir...");
        System.out.println(" i <= j = " + (i <= j)); //true - i, j'den küçüktür
        System.out.println(" j <= i = " + (j <= i)); //false - j, i  den büyüktür
        System.out.println(" k <= j = " + (k <= j)); //true - k, j  ye eşit

        //Eşittir
        System.out.println("Esittir...");
        System.out.println("i == j =" + (i == j)); //false - i, j'den küçüktür
        System.out.println(" k == j = " + (k == j)); //true - k, j'ye eşit

        //Eşit değil
        System.out.println("Esit degil...");
        System.out.println(" i != j = " + (i != j)); //true - i, den küçüktür
        System.out.println(" k != j = " + (k != j)); //false - k, ye eşit
    }
}
```

İlişkisel Operatörler

Örnek

```
Degisken degerleri...
 i = 37
 j = 42
 k = 42
Buyuktur...
 i > j = false
 j > i = true
 k > j = false
Buyuktur veya esittir...
 i >= j = false
 j >= i = true
 k >= j = true
Kucuktur...
 i < j = true
 j < i = false
 k < j = false
Kucuktur veya esittir...
 i <= j = true
 j <= i = false
 k <= j = true
Esittir...
i == j =false
 k == j = true
Esit degil...
 i != j = true
 k != j = false
```

27

Konular

- Atama İşlemleri
- Yordamların Çağrılması
- Java Operatörleri
  - Aritmetik Operatörler
  - İlişkisel Operatörler
  - Mantıksal Operatörler
  - Bit Düzeyinde Operatörler
  - Atama Operatörleri
  - String Operatörü
  - Nesnelerin Karşılaştırılması

Mantıksal Operatörler

- Mantıksal operatörler, **birden çok karşılaştırma işlemini birleştirip tek bir koşul ifadesi** haline getirmek için kullanılır.

Operatör	Kullanılış Şekli	İşlevi/Anlamı
&&	değişken1 && değişken2	VE operatörü
||	değişken1 || değişken2	VEYA operatörü
^	değişken1 ^ değişken2	YA DA operatörü
!	! değişken	DEĞİLini alma operatörü

```
boolean a, b;
 a && b     logical AND
 a || b     logical OR
 a &  b     boolean logical AND
 a |  b     boolean logical OR
 a ^  b     boolean logical exclusive OR
!a          logical NOT
```

```
m>10 && m<55

(m>0 && r<55) && z==10

a>10 && b<55 || r<99
```

29

Mantıksal Operatörler

- **Tek '&'** veya **'|'** operatörü **bitwise işlemi yapar.**
- **Çift "&&"** veya **"||"** operatörü **mantıksal sınama yapar.**

```
package oop;

import java.util.*;

public class JavaProgramlama {
    public static void main(String[] args) {
        int a = 6; // 110
        int b = 4; // 100

        // Bitwise AND
        int c = a & b;
        //   110
        // & 100
        // -----
        //   100

        // Bitwise OR
        int d = a | b;
        //   110
        // | 100
        // -----
        //   110

        System.out.println("c = " + c); // 4
        System.out.println("d = " + d); // 6
        }
}
```

```
c = 4
d = 6
```

Aşağıdaki mantıksal sınama yazımı **doğrudur.**

```
if((a != null) && (a.something == 3)){
}
```

Aşağıdaki mantıksal sınama yazımı **yanlıştır.**

```
if((a != null) & (a.something == 3)){
}
```

30

Mantıksal Operatörler

Örnek

```
(a<b)&&(c<d) --> false
(a<b)||(c<d) --> true
! (a<b) --> false
(a<b)&(c<d) --> false
(a<b)|(c<d) --> true
(a<b)^(c<d) --> true
```

```
package oop;

import java.util.*;

public class JavaProgramlama {
    public static void main( String args[] ) {
        int a = 2 ;
        int b = 3 ;
        int c = 6 ;
        int d = 1 ;

        /*  (a < b) = bu ifadenin doğru (true) olduğunu biliyoruz
            (c < d) = bu ifadenin yanlış (false) olduğunu biliyoruz   */

        System.out.println(" (a<b)&&(c<d) --> " + ((a<b)&&(c<d)) );
        System.out.println(" (a<b)||(c<d) --> " + ((a<b)||(c<d)) );
        System.out.println(" ! (a<b) --> " + ( ! (a<b)) );
        System.out.println(" (a<b)&(c<d) --> " + ((a<b)&(c<d)) );
        System.out.println(" (a<b)|(c<d) --> " + ((a<b)|(c<d)) );
        System.out.println(" (a<b)^(c<d) --> " + ((a<b)^(c<d)) );
    }
}
```

31

Konular

- Atama İşlemleri
- Yordamların Çağrılması
- Java Operatörleri
  - Aritmetik Operatörler
  - İlişkisel Operatörler
  - Mantıksal Operatörler
  - Bit Düzeyinde Operatörler
  - Atama Operatörleri
  - String Operatörü
  - Nesnelerin Karşılaştırılması

Bit Düzeyinde Operatörler

- Bit düzeyinde operatörler, **değişkenlerin/sabitlerin ikili değerlerinin bitleri üzerinde işlem yapar.**

Operatör	Kullanılış Şekli	Açıklama
&	`değişken1 & değişken2`	bit düzeyinde VE
|	`değişken1 | değişken2`	bit düzeyinde VEYA
^	`değişken1 ^ değişken2`	bit düzeyinde YA DA
~	`~değişken`	bit düzeyinde tümleme
>>	`değişken1 >> değişken2`	bit düzeyinde sağa öteleme
<<	`değişken1 << değişken2`	bit düzeyinde sağa öteleme
>>>	`değişken1 >>> değişken2`	bit düzeyinde sağa öteleme (unsigned) ?????

33

Bit Düzeyinde Operatörler

VE (AND) Operatörü

- **Her iki değer de true ise sonuç true olur**, diğer durumlarda false olur.

değişken1	değişken2	Sonuç
0	0	0
0	1	0
1	0	0
1	1	1

```
  1010        10
& 1001        9
---------
  1000
```

34

Bit Düzeyinde Operatörler

VEYA (OR) Operatörü

- **Her iki değer de false ise sonuç false olur**, diğer durumlarda true olur.

değiken1	değişken2	Sonuç
0	0	0
0	1	1
1	0	1
1	1	1

```
   1010        10
 | 1001        9
---------
   1011
```

35

Bit Düzeyinde Operatörler

YA DA (Exlusive-OR) Operatörü

- **Her iki değer aynı ise sonuç false olur**, farklı ise true olur.

Değişken1	değişken2	Sonuç
0	0	0
0	1	1
1	0	1
1	1	0

```
   1010        10
 ^ 1001        9
---------
   0011
```

36

Bit Düzeyinde Operatörler

TÜMLEME (NOT) Operatörü

- a bir değişken ise, ~a ifadesi tümleme işlemini ifade eder.
- ~a = (-a) - 1, şeklinde hesaplanır.

Örnek

- ~10= (-10) -1 = -11 sonucunu verir.

Bit Düzeyinde Operatörler

Örnek

```
(a & b) --> 8
(a | b) --> 11
(a ^ b) --> 3
( ~a ) --> -11
( ~b ) --> -10
( ~c ) --> -9
```

```
package oop;

import java.util.*;

public class JavaProgramlama {
    public static void main( String args[] ) {
        int a = 10, b = 9, c = 8 ;
        System.out.println(" (a & b) --> " + (a & b) );
        System.out.println(" (a | b) --> " + (a | b ) );
        System.out.println(" (a ^ b) --> " + (a ^ b) );
        System.out.println(" ( ~a ) --> " + ( ~a ) );
        System.out.println(" ( ~b ) --> " + ( ~b ) );
        System.out.println(" ( ~c ) --> " + ( ~c ) );
    }
}
```

Bit Düzeyinde Operatörler

ÖTELEME (SHIFT) Operatörleri

- **Bit düzeyinde işlem yapan** bir grup operatörün adı **öteleme operatörleri** olarak adlandırılırlar.
- Öteleme operatörleri, **">>"**, **">>"** ve **">>>"** simgeleriyle gösterilmektedir.
- Öteleme operatörleri veri üzerindeki **bitlerin sağa veya sola kaydırılması amacıyla kullanılır.**
- **Öteleme işleminden sonra** değişkenin **değerinde değişiklik olur.**
- **">>"** operatörü **işaretli (aritmetik) sağa kaydırma** yapar.
- **">>>"** operatörü **işaretsiz (mantıksal) sağa kaydırma** yapar.

39

Bit Düzeyinde Operatörler

Örnek

a = 1001

```
(a >> 1) -->4
(a >> 2) -->2
(a << 1) -->18
(a << 2) -->36
(a >>> 2) -->2
```

```
package oop;

import java.util.*;

public class JavaProgramlama {
    public static void main( String args[] ) {
        int a = 9 ;
        System.out.println(" (a >> 1) -->" + (a >> 1) );
        System.out.println(" (a >> 2) -->" + (a >> 2) );
        System.out.println(" (a << 1) -->" + (a << 1) );
        System.out.println(" (a << 2) -->" + (a << 2) );
        System.out.println(" (a >>> 2) -->" + (a >>> 2) );
    }
}
```

40

Konular

- Atama İşlemleri
- Yordamların Çağrılması
- Java Operatörleri
  - Aritmetik Operatörler
  - İlişkisel Operatörler
  - Mantıksal Operatörler
  - Bit Düzeyinde Operatörler
  - Atama Operatörleri
  - String Operatörü
  - Nesnelerin Karşılaştırılması

Atama Operatörleri

- Sabit değeri veya **değişken değerini başka değişkene aktarır.**

```
enbuyukByte-->127
enbuyukShort-->32767
enbuyukInteger-->2147483647
enbuyukLong-->9223372036854775807

enbuyukFloat-->3.4028235E38
enbuyukDouble-->1.79769313486231570E308

birChar-->S
birBoolean-->true
```

```
public class JavaProgramlama {
    public static void ekranaBas(String deger) {
        System.out.println(deger);
    }

    public static void main( String args[] ) {

        // tamsayılar
        byte enbuyukByte = Byte.MAX_VALUE;
        short enbuyukShort = Short.MAX_VALUE;
        int enbuyukInteger = Integer.MAX_VALUE;
        long enbuyukLong = Long.MAX_VALUE;
        ekranaBas("enbuyukByte-->" + enbuyukByte );
        ekranaBas("enbuyukShort-->" + enbuyukShort );
        ekranaBas("enbuyukInteger-->" + enbuyukInteger );
        ekranaBas("enbuyukLong-->" + enbuyukLong );
        ekranaBas("");

        // gerçek sayılar
        float enbuyukFloat = Float.MAX_VALUE;
        double enbuyukDouble = Double.MAX_VALUE;
        ekranaBas("enbuyukFloat-->" + enbuyukFloat );
        ekranaBas("enbuyukDouble-->" + enbuyukDouble );
        ekranaBas("");

        // diğer temel (primitive) tipler
        char birChar = 'S';
        boolean birBoolean = true;
        ekranaBas("birChar-->" + birChar );
        ekranaBas("birBoolean-->" + birBoolean );
    }
}
```

Atama Operatörleri

- Bitişik atama operatörleri ile atama deyimleri daha kısa yazılabilir.

```
toplam = toplam + 1 ;        toplam += 1 ;
```

Operatör	Kullanılış Şekli	Eşittir
+=	değişken1 += değişken2	değişken1 = değişken1 + değişken2
-=	değişken1-= değişken2	değişken1 = değişken1 - değişken2
*=	değişken1*= değişken2	değişken1 = değişken1 * değişken2
/=	değişken1/= değişken2	değişken1 = değişken1 / değişken2
%=	değişken1%= değişken2	değişken1 = değişken1 % değişken2
&=	değişken1&= değişken2	değişken1 = değişken1 & değişken2
|=	değişken1|= değişken2	değişken1 = değişken1 | değişken2
^=	değişken1^= değişken2	değişken1 = değişken1 ^ değişken2
<<=	değişken1<<= değişken2	değişken1 = değişken1 << değişken2
>>=	değişken1>>= değişken2	değişken1 = değişken1 >> değişken2
>>>=	değişken1>>>= değişken2	değişken1 = değişken1 >>> değişken2

43

Konular

- Atama İşlemleri
- Yordamların Çağrılması
- Java Operatörleri
  - Aritmetik Operatörler
  - İlişkisel Operatörler
  - Mantıksal Operatörler
  - Bit Düzeyinde Operatörler
  - Atama Operatörleri
  - String Operatörü
  - Nesnelerin Karşılaştırılması

String Operatörü

- **"+"** operatörü **String verilerde birleştirme** (ard arda ekleme) yapar **(concatenation).**
- Eğer bir **ifade String ile başlarsa**, onu **izleyen veri tipleri de String'e dönüştürülür.**

```
package oop;

import java.util.*;

public class JavaProgramlama {
    public static void main( String args[] ) {
        int x = 0, y = 1, z = 2;
        System.out.println("Sonuc = " + x + y + z);
        System.out.println("Sonuc = " + (x + y + z));
        }
}
```

```
Sonuc = 012
Sonuc = 3
```

45

Konular

- Atama İşlemleri
- Yordamların Çağrılması
- Java Operatörleri
  - Aritmetik Operatörler
  - İlişkisel Operatörler
  - Mantıksal Operatörler
  - Bit Düzeyinde Operatörler
  - Atama Operatörleri
  - String Operatörü
  - Nesnelerin Karşılaştırılması

Nesnelerin Karşılaştırılması

- **Nesnelerin birbirine eşit olup** olmadığı **"=="** veya **"!="** operatörleriyle sınanabilir.
- Örnekte, **ilk kısımda** nesnelerin değişkenlerinin **değerleri aynıdır,** ancak **referansları farklıdır.**

```
package oop;

import java.util.*;

public class JavaProgramlama {
    public static void main( String args[] ) {
        Integer a1 = new Integer(47);
        Integer a2 = new Integer(47);

        System.out.println("Nesne atamasından önce");
        System.out.println("a1 == a2 --->" + (a1 == a2));
        System.out.println("a1 != a2 --->" + (a1 != a2));
        a1 = a2;
        System.out.println("\nNesne atamasından sonra");
        System.out.println("a1 == a2 --->" + (a1 == a2));
        System.out.println("a1 != a2 --->" + (a1 != a2));
    }
}
```

```
Nesne atamasından önce
a1 == a2 --->false
a1 != a2 --->true

Nesne atamasından sonra
a1 == a2 --->true
a1 != a2 --->false
```

47