Java ile Nesne Merkezli Programlamaya Giriş

4. Bölüm
Operatörler

Akın Kaldıroğlu

www.javaturk.org

Aralık 2016

Küçük Ama Önemli Bir Konu

- Bu dosya ve beraberindeki tüm, dosya, kod, vb. eğitim malzemelerinin tüm hakları **Selsoft Yazılım, Danışmanlık, Eğitim ve Tic. Ltd. Şti**.'ne aittir.
- Bu eğitim malzemelerini kişisel bilgilenme ve gelişiminiz amacıyla kullanabilirsiniz ve isteyenleri **http://www.selsoft.academy** adresine yönlendirip, bu malzemelerin en güncel hallerini almalarını sağlayabilirsiniz.
- Yukarıda bahsedilen amaç dışında, bu eğitim malzemelerinin, ticari olsun/olmasın herhangi bir şekilde, toplu bir eğitim faaliyetinde kullanılması, bu amaca yönelik olsun/olmasın basılması, dağıtılması, gerçek ya da sanal/Internet ortamlarında yayınlanması yasaktır. Böyle bir ihtiyaç halinde lütfen benimle, **akin.kaldiroglu@selsoft.academy** adresinden iletişime geçin.
- Bu ve benzeri eğitim malzemelerine katkıda bulunmak ya da düzeltme ve eleştirilerinizi bana iletmek isterseniz çok sevinirim.
- Bol Java'lı günler dilerim.

İçerik

- Bu bölümde şu konular ele alınacaktır:
  - Operatör kavramı,
  - Temel operatörler,
  - Bit operatörleri,
  - Öncelik ve değerlendirme sırası,
  - Operatörlerde çevrim ve yükseltmeler,
  - Ve String nesneleri için ‘+’ işlemcisi.

Operatörler

Operatörler/ İşlemciler (Operators)

- **Operatör** ya da **işlemci**, bir, iki ya da üç tane işlenen/girdi (operand) üzerinde işlem yapıp bir sonuç üreten semboldür.
  - Tek işlenenli operatörlere, tekli (unary) *operatör,*
  - İki işlenenli operatörlere, ikili (binary) *operatör,*
  - Üç işlenenli operatörlere de, üçlü (ternary) *operatör* denir.
- İşle

```
//Tekli operatör
 operator operand ya da  operand operator

// İkili operatör
operand1 operator operand2

//Üçlü operatör
operand1 ? operand2 :  operand3
```

Atama (Assignment) Operatörü

- **Atama** (**assignment**) operatörü, "**=**", en basit operatördür.
- İki işlenen alır ve sağdaki işleneni (tek bir değer üreten daha karmaşık bir ifade de olabilir) soldaki işlenene atar.
  - Bu atama bazen değer atamasıdır bazen de nesne referansı atamasıdır.

```
//Değer ataması
int i = 5;
boolean b = false;

//Nesne referansı ataması
Car myCar = new Car();
```

İkili Aritmetik Operatörler

- Aritmetik operatörler 5 tanedirler ve hepsi iki işlenen alırlar:
  - **+**: Toplama
  - **-**: Çıkarma
  - ***:** Çarpma
  - **/**: Bölme
  - **%**: Kalan ya da mod
- **+** operatörünün, String nesnelerini de arka arkaya (concatenation) ekleyebileceğini ileride göreceğiz.

Birleşik Atama

- Eğer bir değişkenin değerini değiştirip, yeni değerini yine kendisine atayacaksanız, bunu birleşik atama (compound assignment) ile daha az kod yazarak yapabilirsiniz.
- Birleşik atamalar, **+**, **-**, *****, **/**, **%**, **&**, **|**, **^**, **<<**, **>>** ve **>>>** operatörlerinde kullanılabilir.

```
int i = 5;

i = i + 3;          // i şimdi 8
i += 3;          // i şimdi 11

i -= 4;          // i şimdi 7;
i *= 2;          // i şimdi 14
i /= 7;          // i şimdi 2
```

Aritmetik Yükseltmeler - I

- Aritmetik operatörler, yaptıkları işlem sonucunda en az int tipinde bir değer üretirler. Yani
  - Aritmetik operatörün en az bir işleneni double ise sonuç double,
  - Değilse, aritmetik operatörün en az bir işleneni float ise sonuç float,
  - Değilse, aritmetik operatörün en az bir işleneni long ise sonuç long,
  - Aksi taktirde sonuç int tipinde olur.
- Yani, int ve daha küçük tipteki işlenenlerle yapılan aritmetik işlemler muhakkak bir int sonucu üretir.
- Daha büyük tipler varsa sonuç en büyük tipten olur.

Aritmetik Yükseltmeler - II

- Birleşik atama kullanılması durumunda bir daraltan çevrime ihtiyaç olursa bu otomatik olarak yapılır.

```
int i = 5;
double d = 2.3d;

i = i + d                  // Hata!
i = (int) (i + d);         // Cast gerekli

i = 5;
i += d;          // i şimdi 7
i -= d;          // i şimdi 4;
i *= d;          // i şimdi 9
i /= d;          // i şimdi 3
```

ArithmeticConversion.java

Tekli Aritmetik Operatörler

- Tekli aritmetik operatörler 5 tanedir:
  - **+**: İşleneni pozitif yapar veya pozitif sayısal değeri gösterir. Sayılar bu operatör olmasa da zaten pozitiftirler.
  - **-**: İşleneni negatif yapar veya negatif sayısal değeri gösterir.
  - **++**: Artım operatörüdür, sayısal değeri 1 arttırır.
  - **--**: Eksiltme operatörüdür, sayısal değeri 1 eksiltir.
  - **!**: Mantıksal tümleme operatörüdür, boolean bir işlenenin tümleyenini alır.
- **+** ve **-**, bir atama vb. operatörlerle birlikte kullanılabilir, aksi taktirde "Not a statement" hatası oluşur.
- **+** ve **-**, ikili aritmetik operatörler gibi en az int sonuç üretir.

Arttırma ve Eksiltme Operatörleri

- **++** ve **--**, işlenenin öncesinde ya da sonrasında kullanılabilir:
  - Öncesinde kullanıldığında, önce arttırma ya da eksiltme yapılır, sonra işlenenin değeri hesaplanır,
  - Sonrasında kullanıldığında, önce işlenenin değeri hesaplanır sonı arttırma ya da eksiltme yapılır.

```
int i = 5;

int j = ++i;    // i ve j şimdi 6
j = i++;        // j halen 6 ama i 7 oldu

j = --i;        // i ve j şimdi 6;
j = i--;        // j şimdi de 6 ama i 5 oldu
```

UnaryOperators.java

Kıyaslama (Relational) Operatörleri

- *Kıyaslama* (*relational*) operatörleri 6 tanedir:
- 4 tane büyüklük kıyaslama operatörü, iki basit sayısal (tam ya da kesirli sayı) işlenen alır ve boolean bir sonuç üretirler:
  - **>**: Büyük müdür?
  - **>=**: Büyük eşit midir?
  - **<**: Küçük müdür?
  - **<=**: Küçük eşit midir?
- Eşitlik kıyaslayan operatörler iki tane sayısal, boolean ya da referans işlenen alırlar:
  - **==**: Eşit midir?
  - **!=**: Eşit değil midir?

== Operatörü

- Eşitlik kıyaslamak için "**==**" operatörünü kullanın.
  - **if** ya da **while** gibi yapılarda yanlışlıkla eşitlik kıyaslaması için "**==**" yerine atama operatörü "**=**" yazmak, yaygın bir hatadır.
- "==" operatörünü, işlemcilerinin değerlerinin eşit olup olmadıklarını kıyaslar.
- Bu yüzden "==" operatörü ile
  - Basit tiplerin değerleri
  - Referansların ise adresleri

  kıyaslanır.

Referanslarda == Operatörü

- Referansların adreslerinin kıyaslandığından, referanslar için "==" operatörü ancak ve ancak iki referans da aynı adresi yani nesneyi gösteriyorsa true döndürür.
- Çünkü "==" operatörü, referansların gösterdiği nesnelerin içeriklerinin aynı olup olmadığı kontrol etmez.

```
String s1 = new String("String");
String s2 = new String("String");
if (s1 == s2)
    System.out.println("The same");
else
    System.out.println("Different"); // Different

s1 = s2;   // Equalizing the references.

if (s1 == s2)
    System.out.println("The same"); // The same
else
    System.out.println("Different");
```

RelationalOperators.java

Şartlı (Conditional) Operatörler

- Şartlı ya da koşullu (conditional) operatörler, iki mantıksal ifade arasında şartlı olarak boolean sonuç üreten operatörlerdir.

- İşlenenler de daima boolean ifadeler olmalıdır.
  - **&**: VE (AND), ikili operatördür,
  - **|**: VEYA (OR) , ikili operatördür,
  - **^**: DIŞLAYICI (XOR) , ikili operatördür,
  - **!**: DEĞİL (NOT), tekli operatördür,

Kısayol Şart Operatörleri

- Şartlı ya da koşullu (conditional) operatörlerin kısa yol (short circuit) şekilleri de vardır.
- VE "**&**" için "**&&**", VEYA "**|**" için ise "**||**" kullanılır.
- Kısa yol operatörleri şöyle çalışır:
  - **&&**: Sağ taraf, ancak sol taraf doğruysa (true) değerlendirilir.
  - **||**: Sağ taraf, ancak sol taraf yanlışsa (false) değerlendirilir.
- Kısa yol operatörlerinin sağ taraflarındaki ifadelerde metot çağrısı varsa, bu çağrının yapılmayabileceğine dikkat edin.

ConditionalOperators.java

Aralık Kıyaslaması

- Java'da aralık kıyaslaması (range comparision) ancak şart operatörleri ile yapılabilir.
- Yani şu kod hatalıdır:

```
boolean b = 3 < i < 7;
```

- Bunun yerine şart operatörleri kullanılarak çoklu kıyaslama yapılmalıdır.

```
boolean b = (3 < i) & (i < 7);
```

RangeComparision.java

Üçlü Şart Operatörü

- Üçlü şart (ternary conditional) operatörü, "**?:**" 3 tane işlenen alan tek operatördür.
- İleride ele alınacak olan if-else cümlesinin kısaltılmış halidir.
- İlk ifade boolean olmalıdır ve doğru ise ikinci ifade, değil ise üçüncü ifade, atamanın solundaki değere atanır.

```
int i = 8;
int j = 11;
int min = (i <= j) ? i : j;
```

TernaryOperator.java

Bit Operatörleri

Bit Seviyesinde Çalışan Operatörler

- Java'da, bit seviyesinde işlem yapan operatörler de vardır.
- Bu operatörler, işlenenlerin bit yapıları üzerinden çalışır.
- Bit seviyesinde işlem yapan iki operatör grubu vardır:
  - Mantıksal
  - Kaydırma
- Bu operatörler daha çok, bit seviyesinde işlemlerin yapıldığı, gömülü (embedded) vb. uygulamalarda kullanılır.

Mantıksal Bit Operatörleri - I

- Mantıksal bit (bitwise logical) operatörleri 4 tanedir:
  - **&**: İki işlenenin bitlerini VE (AND) ile işler,
  - **|**: İki işlenenin bitlerini VEYA (OR) ile işler,
  - **^**: İki işlenenin bitlerini DIŞLAYICI (EX-OR) ile işler,
  - **~**: Bir işlenenin bitlerinin TÜMLEYENini (DEĞİLi) alır.
- Bu operatörler tam sayı tipleri üzerinde işlem yaparlar ve en az bir int üretirler.

Mantıksal Bit Operatörleri - II

- Mantıksal bit operatörleri ile şartlı operatörlerin iki tanesi (**&** ve **|**) sembol olarak aynıdır.
  - İşlenenler boolean ise sonuç ta boolean, işlenenler tam sayı ise sonuç da tam sayı olur.
- Dolayısıyla **&** ve **|** overloaded operatörlerdendir.
- Bu operatörlerde kısa yol tanımlı değildir.

BitwiseLogicalOperators.java

Kaydırma (Shift) Operatörleri

- Kaydırma (shift) operatörleri, işlenenin bit dizisini, verilen mesafe kadar sağa ya da sola kaydırırlar.
- Kaydırma operatörleri iki işlenen alır ve işlenenleri tam sayı olmalıdır:
  - **>>**: Girilen sayı kadar sağa kaydırma yapar,
  - **<<**: Girilen sayı kadar sola kaydırma yapar,
  - **>>>**: Girilen sayı kadar sağa, işaretsiz kaydırma yapar.
- Bu operatörler tam sayı tipleri üzerinde işlem yaparlar ve en az bir int üretirler.
- Bir sayıyı sağa kaydırmak onu 2'ye bölmek, sola kaydırmak ise onu 2 ile çarpmak demektir.

ShiftOperators.java

Öncelik ve Değerlendirme Sırası

Öncelik ve Değerlendirme Sırası

- Birden fazla operatör aynı cümlede kullanılırsa hangisinin önce çalışacağı, öncelik sırası (precedence) ile, operatörün birden fazla işleneni varsa, hangisini önce işleyeceği ise değerlendirme sırası (associativity) ile belirlenir.

```
2 + 8 * 5  => 2 + (8 * 5)              // Predecence

a = b = c = 4 => a = (b = (c = 4)) // Associativity
```

Öncelik (Precedence) I

- Bir operatör değerlendirilmeden önce işlenenleri değerlendirilir:
  - Bunun istisnası **&&**, **||** ve **?:** operatörleridir.
- Öncelik sıralamasında daha yukarıda olan operatörler daha aşağıda olanlardan önce değerlendirilirler.
- Aynı önceliğe sahip operatörler aynı ifadede bulunuyorlarsa, değerlendirme sırası soldan sağa olur.
- İfadede bulunan parantezler, değerlendirmede önceliğe sahiptirler.
- Bir metot çağrısında parametreler, soldan sağa değerlendirilir.

Öncelik II

- Öncelik sırasını ezbere bilmek gerekli değildir, bu konudaki ilkelerin bilinmesi önemlidir.
- Ayrıca öncelik sırasını bilmeyi gerektiren kod yazmak ise iyi bir uygulama değildir.
  - Aslolan, olabildiğince fazla satır ve parantez kullanarak okunabilir kod yazmaktır.

```
int a = 6;
int b = 5;
int c = 10;
float rs = a + ++b * c / a * b;

boolean b1 = false, b2 = true, b3 = true;
boolean bool = b1 & b2 | b3;
```

Değerlendirme Sırası (Associativity)

- Sonra gelen arttırma ve eksiltme operatörleri dışında bütün tekli operatörler, sağdan sola doğru işlerler.
- Bütün ikili operatörler soldan sağa doğru işlerler,
  - Bunun tek istisnası ise atama (=) operatörüdür.
  - Birleşik atama operatörleri de bu şekilde davranır.

```
int i = 8;     // From right to left
i = i++;       // From right to left
i = ++i;       // From left to right
a = b = c = 4  =>  a = (b = (c = 4))
// From left to right: The same meaning
a + b + c  and (a + b) + c
a – b – c  and (a - b) - c
// From left to right
new Student().register(); // (new Student()).register()
```

OperatorPrecedence.java

Sayısal Yükseltmeler

Sayısal Yükseltmeler

- Sayısal yükseltmelerin (numeric promotion), belli operatörlerde, işlenenlerin tiplerini, en az bir int sonuç üretecek şekilde yükselttiğini daha önce belirtmiştik.

- Aritmetik yükseltmelerin iki türü vardır:
  - Tekli operatörlerde yükseltme
  - İkili operatörlerde yükseltme

Tekli Yükseltme

- Eğer tekli operatörün işleneni int'den daha küçük bir tip ise, işlenenin tipi int'e yükseltilir, aksi taktirde yükseltme yapılmaz.
- Tekli yükseltme şu operatörlerde yapılır:
  - **+**, **-**, **~**,
  - **>>**, **<<**, **>>>**: Bu operatörlerde yükseltme ayrı ayrı yapılır, yani kayacak sayı long ise mesafe de long olmaz, sadece int'e yükseltilir.
  - Dizi indekslerinde (array index) içinde ve dizi büyüklüklerinde (array size), "**[]**" içinde yükseltme yapılır.
- Diğer tür operatörlerde, örneğin **++** ve **--**'de, yükseltme yapılmaz.

UnaryConversions.java

İkili Yükseltme

- İki işlenen alan operatörlerde, işlenenlerden birisinin tipi double ise diğeri de double, değilse, birisinin tipi float ise diğeri de float, değilse, birisinin tipi long ise diğeri de long, değilse ikisi de gerekiyorsa int tipine yükseltilir.

- Şu ikili operatörlerde yükseltme yapılır:
  - “+”, “-”, “*”, “/”, “%”,
  - “<", "<=", ">“, “>=”, “==”, “!=”,
  - bit operatörleri “&”, “^”, “|”
- Bazı durumlarda “?”de de yükseltme yapılır.

Operatör Overloading
(İşlemci Tekrar Tanımlama)

İşlemciyi Tekrar Tanımlama

- Aynı operatörün farklı bağlamlarda farklı anlamlarda çalışmasına **operatör overloading** (**işlemci tekrar tanımlama**) denir.
- Farklı bağlamdan kasıt genelde farklı operanddır.
- İşlemcilerin programcılar tarafından tekrar tanımlanabilmeleri, C++ gibi bazı dillerde mümkündür ve bu şekilde aynı işlemci farklı bağlamlarda farklı işlevler görür.
- Java işlemcilerin tekrar tanımlanarak kullanımını sıklıkla uygulayan bir dil değildir.
  - Hatta işlemcilerin programcılar tarafından tekrar tanımlanmalarına izin vermez.

Overloaded Operatörler

- Java'da birden fazla bağlamda kullanılan dolayısıyla tekrar tanımlanmış çok az operatör vardır:
  - En bilinen overlaoded operatör "**+**"dır.
    - "+" sayıları toplarken ve String ile kullanımda, String nesnelerini arka ark eklemektedir.
  - **&** ve **|** de overloaded operatörlerdendir.
    - Operandları boolean olduğunda şart (conditional) operatörü olur ve boo sonuç üretirken, operandları tam sayı (byte, chart, short, int ve long) olduğunda ise bit seviyesinde işlem yaparlar ve yine bir tamsayı üretirler.

String Ekleme İşlemcisi "+" - I

- "+", String nesnelerinin arka arkaya birbirlerine eklenmeleri için de kullanılır.
- Eğer "+" işlemcisinin işlenenlerinden bir tanesi String ise, sonuç String nesnesi ile diğer işlenenin String formunun arka arkaya eklenmesidir.

```
"The square root of 2 is " + Math.sqrt(2)

"The square root of 2 is 1.4142135623730952"
```

String Ekleme İşlemcisi “+” - II

- Unutmayın, “+” işlemcisi işleme soldan başlar (left-associative).
- Bu yüzden “+” işlemcisinin toplama mı yapacağı yoksa arda arda ekleme mi yapacağı soldaki işlenenin tipiyle belirlenir.

```
"Java" + 1 + 2 => Java12
1 + 2 + "Java"  => 3Java
```

StringConcat.java

char ile “+” Kullanımı

- “+”, **char** değerlerin arka arkaya birbirlerine eklenmeleri için kullanıldığında, karakterlerin tam sayı değerlerini toplar ve int bir sonuç üretir.
  - Çünkü iki tane karakterin yanyana gelip yeni bir char oluşturması mümkün değildir.
- Bu durum “+”ın String nesnesi ile kulanımından farklıdır.
  - Dolayısıyla “+”, String nesnelerini ard arda eklerken (concat), **char** ın tam sayı değerleri toplar

```
char a = 'a', b = 'b';
int c = a + b;                    // c => 195
System.out.println(a + b);          // 195
System.out.println("a + b: " + 'a' + 'b');// a+b: ab
System.out.println('a' + 'b');      // 195
```

CharConcat.java

instanceof Operatörü

instanceof Operatörü

- Referansların gösterdiği nesnelerin gerçek tipini kontrol etmek amacıyla kullanılan **instanceof** operatörü daha ileride ele alınacaktır.

Özet

- Bu bölümde Java'nın operatörlerini işledik.
- Tekli ve ikili operatörlerle bir tane olan üçlü operatörü ele aldık.
- Operatörlerin çalışmasında sayısal tiplerle ilgili en az int olacak şekilde bazı tip yükseltmeleri yaptığını gördük.
- Benzer şekilde operatörlerin çalışmasında, işlenenlerini ele alırken sıraya önem verdiğini de gördük.
- Operatörlerin, aynı ifadede kullanıldığında öncelik sıralarının önemli olduğunu gördük.

Ödevler

Ödevler

- Operators isimli bir sınıf oluşturup, içine koyacağınız main metoda aşağıdaki operatörleri kullanacak şekilde, yerel değişkenlerle işlemler yapın. İşlemleri yapmak için gerekli yardımcı metotları da oluşturun.
  - Birleşik atamalar,
  - Değişkenin önünde ve arkasında kullanarak arttırma ve eksiltme operatörleri,
  - Normal ve kısa devre halleriyle şartlı operatörler.

Ödevler

- Bottles.java isimli örneğin nasıl çalıştığını inceleyin ve anlayın.