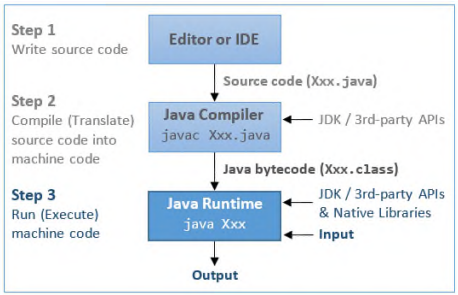
İleri Programlama

Temel Sözdizimi, Değişkenler ve Türler

Hüseyin Ahmetoğlu

Java Kod Şablonu ve Kodlama Adımları

```
1  /**
2   * Comment to state the purpose of the program
3   */
4  public class Classname {   // Choose a meaningful Classname. Save as "Classname.java"
5     public static void main(String[] args) {  // Entry point of the program
6        // Your programming statements here!!!
7     }
8  }
```



Yorum Satırı

- Yorumlar, kodunuzu ve program mantığınızı belgelemek ve açıklamak için kullanılır. Yorumlar, programlama beyanları değildir. Derleyici tarafından yok sayılırlar ve programın yürütülmesinde hiçbir sonuçları yoktur. Yine de, başkalarının programlarınızı anlaması için (ve ayrıca üç gün sonra kendiniz için) belgeler ve açıklamalar sağlamak için yorumlar ÇOK ÖNEMLİDİR.
- Çok Satırlı Yorum: / * ile başlar ve * / ile biter ve birden çok satıra yayılabilir. / ** .... * / özel bir belge açıklamasıdır. Bu yorumlar, dokümantasyon üretmek için çıkarılabilir.
- Satır Sonu (Tek Satır) Yorum: // ile başlar ve mevcut satırın sonuna kadar sürer.
- Program geliştirme sırasında, bir yığın ifadeyi geri alınamaz bir şekilde silmek yerine, bu ifadeleri daha sonra gerekirse geri alabilmek için yorum yapabilirsiniz.

Deyimler

- **Deyim**: Bir programlama ifadesi, bir programdaki en küçük bağımsız birimdir, tıpkı Türkçe'deki bir cümle gibi. Bir parça programlama eylemi gerçekleştirir.
- Tıpkı Türkçe bir cümlenin nokta ile bitmesi gibi, bir programlama deyimi noktalı virgülle (;) sonlandırılmalıdır. (Neden bir cümle gibi bir nokta ile bitmiyor? Bunun nedeni, dönemin ondalık sayılarının noktayla kullanmasıdır.)

```
// Each of the following lines is a programming statement, which ends with a semi-colon (;).
// A programming statement performs a piece of programming action.
int number1 = 10;
int number2, number3 = 99;
int product;
number2 = 8;
product = number1 * number2 * number3;
System.out.println("Hello");
```

Bloklar

- **Blok**: Blok, bir çift küme parantezi {} ile çevrili bir programlama deyim grubudur.
- Blok içindeki tüm ifadeler tek bir birim olarak ele alınır. Bloklar, sınıf, yöntem, if-else ve döngü gibi yapılarda gövde olarak kullanılır; bunlar birden çok ifade içerebilir, ancak bir birim (tek gövde) olarak değerlendirilir.
- Bir bileşik ifadeyi sonlandırmak için kapanış parantezinden sonra noktalı virgül koymaya gerek yoktur. Boş bloğa (yani, küme parantezlerinin içinde hiçbir ifade olmayışına) izin verilmez.

Bloklar

```
// Each of the followings is a "compound" statement comprising one or more blocks of statements.
// No terminating semi-colon needed after the closing brace to end the "compound" statement.
// Take note that a "compound" statement is usually written over a few lines for readability.
if (mark >= 50) {     // A if statement
   System.out.println("PASS");
   System.out.println("Well Done!");
   System.out.println("Keep it Up!");
}

if (input != -1) {   // A if-else statement
   System.out.println("Continue");
} else {
   System.out.println("Exit");
}

i = 1;
while (i < 8) {      // A while-loop statement
   System.out.print(i + " ");
   ++i;
}

public static void main(String[] args) {   // A method definition statement
   ...statements...
}

public class Hello {   // A class definition statement
   ...statements...
}
```

Beyaz boşluklar

- Beyaz Boşluklar: Boşluk, sekme ve satırsonu toplu olarak beyaz boşluklar olarak adlandırılır.

- Java, çoğu programlama dili gibi, fazladan beyaz boşlukları yok sayar. Yani, birden çok bitişik beyaz boşluk, tek bir beyaz boşluk olarak değerlendirilir. Ek beyaz boşluklar ve ekstra satırlar göz ardı edilir.

```
int sum = 0;     // Cannot write "intsum". Need at least one white space between "int" and "sum"
double average;  // Again, need at least a white space between "double" and "average"
```

```
// Same as above with many redundant white spaces. Hard to read.
int  sum
=0     ;

   double
average
;

// Also same as above with minimal white space. Also hard to read
int sum=0;double average;
```

Kaynak Kodu biçimlendirme

- Kaynak Kodunu Biçimlendirme: Belirtildiği gibi, fazladan beyaz boşluklar yok sayılır ve hesaplama açısından önemi yoktur. Bununla birlikte, uygun girinti (sekmeler ve boşluklarla) ve ekstra boş satırlar, programın okunabilirliğini büyük ölçüde artırır. Bu, programlarınızı anlamak için başkaları için (ve üç gün sonra sizin için) son derece önemlidir.

```
public class Hello{public static void main(String[] args){System.out.println("Hello, world!");}}
```

Kaynak Kodu biçimlendirme

- Küme parantezleri: Java'nın kuralı, başlangıç parantezini satırın sonuna yerleştirmek ve bitiş parantezini ifadenin başlangıcıyla hizalamaktır. {} 'İ doğru şekilde eşleştirin. Dengesiz {}, yeni başlayanlar için en yaygın sözdizimi hatalarından biridir.
- Girinti: Bir bloğun gövdesinin her seviyesini, bloğun hiyerarşisine göre fazladan 3 veya 4 boşluk girintili yapın. Sekme boşlukları düzenleyiciye bağlı olduğundan sekme kullanmayın.
- "Kod yazıldığından çok daha sık okunur." Bu nedenle, kurallara ve önerilen kodlama stiline uyarak kodunuzun okunabilir olduğundan (başkaları ve 3 gün sonra kendiniz tarafından) emin olmalısınız.

Kaynak Kodu biçimlendirme

```
/**
 * Recommended Java programming style (Documentation comments about the class)
 */
public class ClassName {      // Place the beginning brace at the end of the current line
   public static void main(String[] args) {  // Indent the body by an extra 3 or 4 spaces for each level

      // Use empty line liberally to improve readability
      // Sequential statements
      statement-1;
      statement-2;

      // A if-else statement
      if (test) {
         true-statements;
      } else {
         false-statements;
      }

      // A loop statement
      init;
      while (test) {
         body-statements;
         update;
      }
   }
}   // Ending brace aligned with the start of the statement
```

Değişkenler ve Türler

İçerisinde veri sakladığımız, ismini ve tipini bizim belirlediğimiz bellek alanlarına **değişken** (*variable*) adı verilmektedir. Değişkenler, bellekte herhangi bir adresi gösteren sembolik birer isimden başka bir şey değildir. Kullandığımız programlama dili ne olursa olsun kendisine verdiğimiz değerleri belleğinde hangi ad ile saklayacağını biz söylüyoruz. Belleğin neresinde saklayacağına (adres bilgisine) ise bilgisayar kendisi karar vermektedir. Bilgilerin geçici olarak tutulduğu yere **bellek** denir. Değişkenlerde belleklerde tutulur.

Değişkenler ve Türler

Java dilinde değişkenleri üçe ayırabiliriz:

1. **Sayısal tipteki değişkenler:** İçerisinde sayıları sakladığımız değişkenlere **sayısal** veya **nümerik değişken** adını veriyoruz.

```
int a;
// seklindeki tanımlamada a degişkeni sadece tamsayı alabilir.
```

2. **Alfa sayısal tipteki değişkenler:** İçerisinde sayı haricindeki bilgileri (isim, soy isim, adres, vb.) sakladığımız değişkenlere de **alfa sayısal, alfa nümerik, metin** veya **string değişken** adını veriyoruz.

```
String a;
// a degiskeni sadece string türünde veri alabilir.
```

3. **Referans tipindeki değişkenler (yapılandırıcı-constructer):** Nesnelerin hafızadaki adresini tutarlar. Nesne değişkenlerini tanımlamak için `new` komutu kullanılır. Nesne `new` komutu ile oluşturulduğunda hafızada bu nesnenin tüm bileşenlerine yetecek kadar yer ayrılır.

```
Araba taksi // nesnenin referansını bildirir.
taksi= new Araba()  // Araba sınıfından taksi nesnesi olusturulur.
```

Değişkenler ve Türler

- Bir değişkenin bir veri türü vardır. Sık kullanılan Java veri türleri şunlardır:
  - int: 123 ve -456 gibi tam sayılar (tam sayılar) içindir.
  - double: e veya E'nin 10 tabanının üssünü gösterdiği 3.1416, -55.66, 1.2e3 veya -4.5E-6 gibi isteğe bağlı bir ondalık basamağa ve kesirli bölüme sahip kayan noktalı sayı (gerçek sayılar) anlamına gelir.
  - String: "Merhaba" ve "Günaydın!" Gibi metinler içindir. Dizeler bir çift, çift tırnak işareti içine alınır.
  - char: "a", "8" gibi tek bir karakter içindir. Bir karakter, bir çift tek tırnak içine alınır.

Değişkenler ve Türler

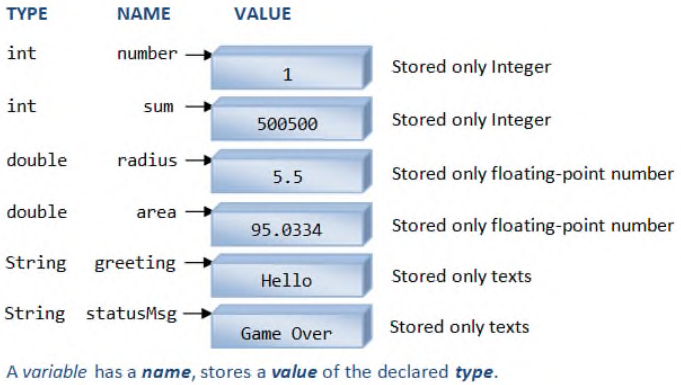
- Java'da, bir değişkeni kullanmadan önce bir değişkenin adını ve türünü belirtmeniz gerekir. Örneğin,

```
int sum;          // Declare an "int" variable named "sum"
double average;   // Declare a "double" variable named "average"
String message;   // Declare a "String" variable named "message"
char grade;       // Declare a "char" variable named "grade"
```

- Bir değişken, beyan edilen veri türünün bir değerini saklayabilir. Çoğu programlama dilinde bir değişkenin bir türle ilişkilendirildiğini ve yalnızca bu belirli türdeki değeri depolayabildiğini not etmek önemlidir. Örneğin, bir int değişkeni 123 gibi bir tamsayı değeri depolayabilir, ancak 12.34 gibi kayan noktalı sayı veya "Merhaba" gibi bir dize bu değişken içerisinde saklanamaz.
- Tür kavramı, ikili dizilerden (0'lar ve 1'ler) oluşan verilerin yorumlanmasını basitleştirmek için erken programlama dillerinde tanıtıldı.
- Tür, verilerin boyutunu ve düzenini, değerlerinin aralığını ve uygulanabilecek işlem kümesini belirler.

Değişkenler ve Türler

- Yandaki şemada üç tür değişken gösterilmektedir: int, double ve String.
- Bir int değişkeni bir tamsayı (veya tam sayı veya sabit noktalı sayı) depolar
- Double değişken bir kayan noktalı sayı (veya gerçek sayı) depolar
- String değişkeni metinleri depolar.



Değişken adlandırma kuralları

- Bir değişkeni (veya bir yöntem veya sınıf gibi başka bir varlığı) adlandırmak için bir tanımlayıcı gereklidir. Java, tanımlayıcılara aşağıdaki kuralları uygular:
- Tanımlayıcı, büyük ve küçük harfler (a-z, A-Z), rakamlar (0-9), alt çizgi (_) ve dolar işaretinden ($) oluşan, herhangi bir uzunluktaki karakter dizisidir.
- Beyaz boşluk (boşluk, sekme, satırsonu) ve diğer özel karakterlere (+, -, *, /, @, &, virgül, vb.) İzin verilmez. Boşluk ve tireye (-) izin verilmediğini, yani "maksimum deger" ve "maksimum-deger" in geçerli adlar olmadığını unutmayın.
- Bir tanımlayıcı bir harf (a-z, A-Z) veya alt çizgi (_) ile başlamalıdır. Bir rakamla (0-9) başlayamaz (çünkü bu bir sayı ile karıştırılabilir).
- Bir tanımlayıcı, ayrılmış bir anahtar kelime veya ayrılmış bir değişmez değer (ör., Class, int, double, if, else, for, true, false, null) olamaz.
- Tanımlayıcılar büyük / küçük harfe duyarlıdır. book, Book ve BOOK aynı değişken DEĞİLDİR.
- Örnekler: abc, _xyz, $123, _1_2_3 geçerli tanımlayıcılardır. Ancak 1abc, min-value, yüzey alanı, ab@c geçerli tanımlayıcılar DEĞİLDİR.

Değişken adlandırma kuralları

- Değişken adı, bir isim veya kelimeler arasında boşluk bırakmadan birkaç kelimeden oluşan bir isim tümcesidir. İlk kelime küçük harfle yazılırken, geri kalan kelimeler ilk harfleri büyük. Örnekler için radius, area, fontSize, numStudents, xMax, yMin, xTopLeft, isValidInput ve thisIsAVeryLongVariableName.

Değişken Tanımlama

- Programınızda bir değişken kullanmak için, önce adını ve türünü aşağıdaki sözdizimlerinden birinde bildirerek tanıtmanız gerekir.
- Bir değişkeni bildirme eylemi, tipin bir değerini tutabilen büyüklükte bir depo tahsis eder.

Syntax	Example

`// Declare a variable of a specified type`
`type identifier;`
	`int sum;`
	`double average;`
	`String statusMsg;`

`// Declare multiple variables of the SAME type,`
`//  separated by commas`
`type identifier1, identifier2, ..., identifierN;`
	`int number, count;`
	`double sum, difference, product, quotient;`
	`String helloMsg, gameOverMsg;`

`// Declare a variable and assign an initial value`
`type identifier = initialValue;`
	`int magicNumber = 99;`
	`double pi = 3.14169265;`
	`String helloMsg = "hello,";`

`// Declare multiple variables of the SAME type,`
`//  with initial values`
`type identifier1 = initValue1, ..., identifierN = initValueN;`
	`int sum = 0, product = 1;`
	`double height = 1.2; length = 3.45;`
	`String greetingMsg = "hi!", quitMsg = "bye!";`

Değişken Tanımlama

- Her değişken yalnızca bir kez bildirilebilir çünkü tanımlayıcı benzersiz olmaldır..
- Kullanılmadan önce bildirildiği sürece, programın herhangi bir yerinde bir değişken tanımlayabilirsiniz.
- Bir değişkenin türü, bildirildikten sonra program içinde değiştirilemez.
- Bir değişken bildirim ifadesi bir türle başlar ve yalnızca bu tür için çalışır. Başka bir deyişle, tek bir bildirim ifadesinde iki farklı türdeki değişkenleri bildiremezsiniz.
- Java, statik olarak yazılmış bir dildir. Bu, türün derleme zamanında çözüldüğü ve asla değişmediği anlamına gelir.

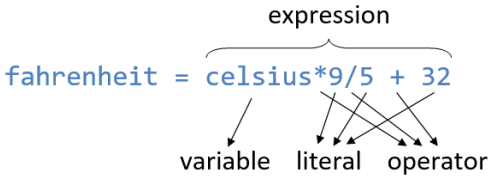
Sabitler (Final değişkenler)

- Sabitler değiştirilemez (*immutable*) değişkenlerdir ve final anahtar sözcüğü ile bildirilir. final değişkenlere yalnızca BİR KEZ değer atayabilirsiniz. Program yürütülürken değerleri değiştirilemez. Örneğin:

```
final double PI = 3.14159265;  // Declare and initialize the constant

final int SCREEN_X_MAX = 1280;
SCREEN_X_MAX = 800;  //compilation error: cannot assign a value to final variable

// You can only assign value to final variables ONCE
final int SCREEN_Y_MIN;
SCREEN_Y_MIN = 0;   // First assignment
SCREEN_Y_MIN = 10;  //compilation error: variable might already have been assigned
```

Atamalar

- Belirli bir türden tek bir değer elde etmek için değerlendirilebilen, işleçlerin ('+' ve '-' gibi) ve işlenenlerin (değişkenler veya sabit değerler) birleşimi ile değişkenlere değer atanabilir.




```
// "int" literals
((1 + 2) * 3 / 4) % 6   // This expression is evaluated to an "int" value

// "double" literals
3.45 + 6.7              // This expression is evaluated to a "double" value

// Assume that variables sum and number are "int"
sum + number * number   // evaluates to an "int" value

// Assume that variables principal and interestRate are "double"
principal * (1.0 + interestRate)  // evaluates to a "double" value
```

Atamalar

- Bir atama ifadesi, RHS'yi (Sağ Taraf) değerlendirir ve ortaya çıkan değeri LHS'nin (Sol Taraf) değişkenine atar.

Syntax	Example
`// Assign the RHS literal value to the LHS variable` `variable = literalValue;`	`int number;` `number = 9;`
`// Evaluate the RHS expression and assign the result to the LHS variable` `variable = expression;`	`int sum = 0, number = 8;` `sum = sum + number;`

```
int number;
number = 8;           // Assign RHS literal value of 8 to the LHS variable number
number = number + 1;  // Evaluate the RHS expression (number + 1),
                      //  and assign the resultant value back to the LHS variable number

8 = number;           // Invalid in Programming, LHS shall be a variable
number + 1 = sum;     // Invalid in Programming, LHS shall be a variable
```

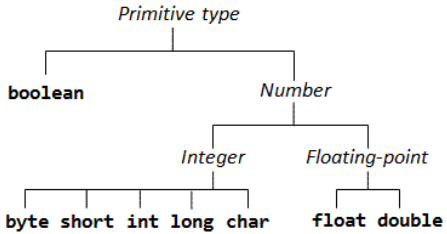
```
int x = 79;
x = x + 1;

END
```

x 80

İlkel Türler

- Java'da iki geniş veri türü kategorisi vardır:

  İlkel türler (ör. int, double),
  Referans türleri (ör. Nesneler ve diziler).



TYPE		DESCRIPTION
`byte`	**Integer**	8-bit signed integer The range is $[-2^7, 2^7-1] = [-128, 127]$
`short`		16-bit signed integer The range is $[-2^{15}, 2^{15}-1] = [-32768, 32767]$
`int`		32-bit signed integer The range is $[-2^{31}, 2^{31}-1] = [-2147483648, 2147483647]$ (≈9 digits, ±2G)
`long`		64-bit signed integer The range is $[-2^{63}, 2^{63}-1] = [-9223372036854775808, 9223372036854775807]$ (≈19 digits)
`float`	**Floating-Point Number** $F \times 2^E$	32-bit single precision floating-point number (≈6-7 significant decimal digits, in the range of $\pm[1.4 \times 10^{-45}, 3.4028235 \times 10^{38}]$)
`double`		64-bit double precision floating-point number (≈14-15 significant decimal digits, in the range of $\pm[4.9 \times 10^{-324}, 1.7976931348623157 \times 10^{308}]$)
`char`	**Character** Represented in 16-bit Unicode `'\u0000'` to `'\uFFFF'`. Can be treated as integer in the range of `[0, 65535]` in arithmetic operations. (Unlike C/C++, which uses 8-bit ASCII code.)	
`boolean`	**Binary** Takes a literal value of either `true` or `false`. The size of `boolean` is not defined in the Java specification, but requires at least one bit. `boolean`s are used in *test* in decision and loop, not applicable for arithmetic operations. (Unlike C/C++, which uses integer `0` for false, and non-zero for true.)	

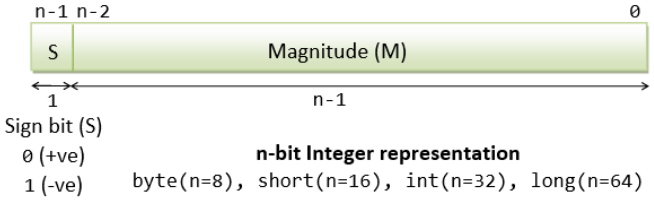
Tamsayılar ve Kayan Noktalı Sayılar

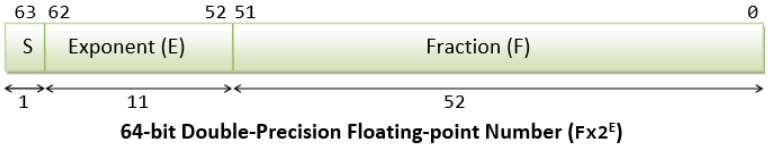
- Bilgisayar programlamasında tam sayılar (123, -456 gibi) ve kayan nokta sayıları (1.23, -4.56, 1.2e3, -4.5e-6 gibi) TAMAMEN farklıdır.

  Tamsayılar ve kayan nokta sayıları farklı şekilde temsil edilir ve saklanır.
  Tamsayılar ve kayan noktalı sayılar farklı şekilde çalıştırılır.

Tamsayılar ve Kayan Noktalı Sayılar Nasıl Gösterilir

ve Nasıl Bilgisayar Belleğinde Saklanır?





Veri Gösterimi

- `byte 1` is `"00000001"` (8-bit).
- `short 1` is `"00000000 00000001"` (16-bit).
- `int 1` is `"00000000 00000000 00000000 00000001"` (32-bit).
- `long 1` is `"00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000001"` (64-bit).
- `float 1.0` is `"0 01111111 0000000 00000000 00000000"` (32-bit).
- `double 1.0` is `"0 01111111111 0000 00000000 00000000 00000000 00000000 00000000 00000000"` (64-bit).
- `char '1'` is `"00000000 00110001"` (16-bit) (Unicode number 49).
- `String "1"` is a complex object (many many bits).

- Bir değeri yorumlamadan önce bir değerin türünü bilmeniz GEREKİR. Örneğin, "00000000 00000000 00000000 00000001" bit modeli, türünü (veya temsilini) bilmediğiniz sürece yorumlanamaz.

İlkel Sayı Türlerinin Maksimum / Minimum Değerleri

```
public class PrimitiveTypesMinMaxBitLen {
   public static void main(String[] args) {
      /* int (32-bit signed integer) */
      System.out.println("int(min) = " + Integer.MIN_VALUE);
      //int(min) = -2147483648
      System.out.println("int(max) = " + Integer.MAX_VALUE);
      //int(max) = 2147483647
      System.out.println("int(bit-length) = " + Integer.SIZE);
      //int(bit-length) = 32

      /* byte (8-bit signed integer) */
      System.out.println("byte(min) = " + Byte.MIN_VALUE);
      //byte(min) = -128
      System.out.println("byte(max) = " + Byte.MAX_VALUE);
      //byte(max) = 127
      System.out.println("byte(bit-length) = " + Byte.SIZE);
      //byte(bit-length) = 8

      /* short (16-bit signed integer) */
      System.out.println("short(min) = " + Short.MIN_VALUE);
      //short(min) = -32768
      System.out.println("short(max) = " + Short.MAX_VALUE);
      //short(max) = 32767
      System.out.println("short(bit-length) = " + Short.SIZE);
      //short(bit-length) = 16

      /* long (64-bit signed integer) */
      System.out.println("long(min) = " + Long.MIN_VALUE);
      //long(min) = -9223372036854775808
      System.out.println("long(max) = " + Long.MAX_VALUE);
      //long(max) = 9223372036854775807
      System.out.println("long(bit-length) = " + Long.SIZE);
      //long(bit-length) = 64

      /* char (16-bit character or 16-bit unsigned integer) */
      System.out.println("char(min) = " + (int)Character.MIN_VALUE);
      //char(min) = 0
      System.out.println("char(max) = " + (int)Character.MAX_VALUE);
      //char(max) = 65535
      System.out.println("char(bit-length) = " + Character.SIZE);
      //char(bit-length) = 16

      /* float (32-bit floating-point) */
      System.out.println("float(min) = " + Float.MIN_VALUE);
      //float(min) = 1.4E-45
      System.out.println("float(max) = " + Float.MAX_VALUE);
      //float(max) = 3.4028235E38
      System.out.println("float(bit-length) = " + Float.SIZE);
      //float(bit-length) = 32

      /* double (64-bit floating-point) */
      System.out.println("double(min) = " + Double.MIN_VALUE);
      //double(min) = 4.9E-324
      System.out.println("double(max) = " + Double.MAX_VALUE);
      //double(max) = 1.7976931348623157E308
      System.out.println("double(bit-length) = " + Double.SIZE);
      //double(bit-length) = 64

      /* No equivalent constants for boolean type */
   }
}
```

String

- 8 ilkel türün yanında, bir diğer önemli ve sık kullanılan tür String'dir. String, "Merhaba dünya" gibi bir karakter dizisidir (metinler). String, ilkel bir tür değildir

```
String greetingMsg = "hello, world";  // String is enclosed in double-quotes
char gender = 'm';       // char is enclosed in single-quotes
String statusMsg = "";   // an empty String
```

Değişkenler için Veri Türlerinin Seçimi

- Bir programcı olarak, programlarınızda kullanılacak değişkenlerin türüne SİZ karar vermelisiniz. Çoğu zaman karar sezgiseldir. Örneğin, sayma ve tam sayı için bir integer türü kullanın; kesirli bölümlü sayı için bir kayan nokta türü, metin mesajı için String, tek bir karakter için karakter ve ikili sonuçlar için boole.

```
String name = "Paul";
String brand = "idol";
double processorSpeedInGHz = 2.66; // or float
double ramSizeInGB = 8;            // or float
int harddiskSizeInGB = 500;        // or short
int monitorInInch = 15;            // or byte
double price = 1760.55;
char servicePlan = 'C';
boolean onSiteService = true;
boolean extendedWarranty = false;
```

Aritmetik operatörler

Operator	Mode	Usage	Description	Examples
+	Binary Unary	x + y +x	Addition Unary positive	1 + 2 ⇒ 3 1.1 + 2.2 ⇒ 3.3
-	Binary Unary	x - y -x	Subtraction Unary negate	1 - 2 ⇒ -1 1.1 - 2.2 ⇒ -1.1
*	Binary	x * y	Multiplication	2 * 3 ⇒ 6 3.3 * 1.0 ⇒ 3.3
/	Binary	x / y	Division	1 / 2 ⇒ 0 1.0 / 2.0 ⇒ 0.5
%	Binary	x % y	Modulus (Remainder)	5 % 2 ⇒ 1 -5 % 2 ⇒ -1 5.5 % 2.2 ⇒ 1.1

Aritmetiksel İfadeler

- (1+2*a)/3 + (4*(b+c)*(5-d-e))/f - 6*(7/g+h)

$$\frac{1+2a}{3}+\frac{4(b+c)(5-d-e)}{f}-6\left(\frac{7}{g}+h\right)$$

Öncelik Kuralları

- Parantezler () en yüksek önceliğe sahiptir ve değerlendirme sırasını değiştirmek için kullanılabilir.
- Çarpma (*), bölme (/) ve modül (%) aynı önceliğe sahiptir. Toplama (+) ve çıkarmaya (-) göre önceliklidirler.
  - Örneğin, 1 + 2 * 3-4 / 5 +% 6 7, 1+ (2 * 3) - (4/5) + (6% 7) olarak yorumlanır.
- Aynı öncelik seviyesinde (yani toplama / çıkarma ve çarpma / bölme / modül), ifade soldan sağa doğru değerlendirilir (sola ilişkilendirilebilir olarak adlandırılır).
  - Örnekler için 1 + 2-3 + 4, ((1 + 2) -3) +4 olarak değerlendirilir
  - ve 1 * 2% 3/4, ((1 * 2)% 3) / 4 olarak değerlendirilir.

Aritmetik İşlemlerde Tip Dönüşümü

- Aritmetik operatörler (+, -, *, /,%) yalnızca ilkel sayı türleri için geçerlidir: byte, short, int, long, float, double ve char.
- Boole için geçerli değildirler.

İnt, long, float, double'ın Aynı Tip Operandları

- `int` ⊕ `int` ⇒ `int`, where ⊕ denotes a binary arithmetic operators such as +, -, *, /, %.
- `long` ⊕ `long` ⇒ `long`
- `float` ⊕ `float` ⇒ `float`
- `double` ⊕ `double` ⇒ `double`

İnt bölme sonucunun bir int yani int / int ⇒ int ürettiğine dikkat etmek önemlidir. Örneğin, 1/2 ⇒ 0 (int), ancak 1.0 / 2.0 ⇒ 0.5 (double / double ⇒ double).

byte, short, char: int

- byte ⊕ byte ⇒ int ⊕ int ⇒ int, where ⊕ denotes a binary arithmetic operators such as +, -, *, /, %.
- short ⊕ short ⇒ int ⊕ int ⇒ int
- char ⊕ char ⇒ int ⊕ int ⇒ int

```
byte b1 = 5, b2 = 9, b3;
// byte + byte -> int + int -> int
b3 = b1 + b2;           // error: RHS is "int", cannot assign to LHS of "byte"
b3 = (byte)(b1 + b2);   // Need explicit type casting (to be discussed later)
```

Bununla birlikte, bileşik aritmetik operatörler (+ =, - =, * =, / =,% =) kullanılırsa (daha sonra işlenecek), sonuç otomatik olarak LHS'ye dönüştürülür. Örneğin,

```
byte b1 = 5, b2 = 9;
b2 += b1;     // Result in "int", but automatically converted back to "byte"
```

Karışık Tip Aritmetik İşlemler

- İki işlenen farklı türlere aitse, daha küçük türün değeri otomatik olarak daha büyük türe yükseltilir (örtük tür çevirimi olarak bilinir). İşlem daha sonra daha büyük tipte gerçekleştirilir ve daha büyük tipte bir değer olarak değerlendirilir.
- byte, short veya char, diğer işlenenin türüyle karşılaştırılmadan önce int'e yükseltilir.
- Yükseltme sırası şöyledir: int ⇒ long ⇒ float ⇒ double.

1. int / double ⇒ double / double ⇒ double. Hence, 1/2 ⇒ 0, 1.0/2.0 ⇒ 0.5, 1.0/2 ⇒ 0.5, 1/2.0 ⇒ 0.5
2. 9 / 5 * 20.1 ⇒ (9 / 5) * 20.1 ⇒ 1 * 20.1 ⇒ 1.0 * 20.1 ⇒ 20.1
3. char '0' + int 2 ⇒ int 48 + int 2 ⇒ int 50
4. char ⊕ float ⇒ int ⊕ float ⇒ float ⊕ float ⇒ float
5. byte ⊕ double ⇒ int ⊕ double ⇒ double ⊕ double ⇒ double

Overflow

```
/**
 * Illustrate "int" overflow
 */
public class OverflowTest {
   public static void main(String[] args) {
      // Range of int is [-2147483648, 2147483647]
      int i1 = 2147483647;        // maximum int
      System.out.println(i + 1);  //-2147483648 (overflow)
      System.out.println(i + 2);  //-2147483647 (overflow)
      System.out.println(i + 3);  //-2147483646 (overflow)
      System.out.println(i * 2);  //-2 (overflow)
      System.out.println(i * i);  //1 (overflow)

      int i2 = -2147483648;         // minimum int
      System.out.println(i2 - 1);   //2147483647 (overflow)
      System.out.println(i2 - 2);   //2147483646 (overflow)
      System.out.println(i2 * i2);  //0 (overflow)
   }
}
```

Tam sayılar ve Kayan noktalı sayılar

```java
/**
 * Test preciseness for int/float/double
 */
public class TestPreciseness {
   public static void main(String[] args) {
      // doubles are NOT precise
      System.out.println(2.2 + 4.4);       //6.6000000000000005
      System.out.println(6.6 - 2.2 - 4.4); //-8.881784197001252E-16 (NOT Zero!)
      // Compare two doubles
      System.out.println((6.6) == (2.2 + 4.4));  //false

      // int is precise, float/double are NOT!
      int i1 = 123456789;
      System.out.println(i1*10); //1234567890 (exact within the range)

      float f1 = 123456789.0f;   // float keeps 6-7 significant digits
      System.out.println(f1);    //1.23456792E8 (=123456792 close but not exact)
      System.out.println(f1*10); //1.23456794E9 (=1234567940)
   }
}
```

Örnek

Bir manavdaki toplam meyve miktarını hesaplayan basit bir program yazalım.

Çözüm:

```
public class Ornek1 {
       public static void main(String[] args) {
              // Değişkenleri tanımlıyoruz
              byte nar=10; // nar miktarı
              int elma= 500; // elma miktarı
              int armut = 120;  // armut miktarı
              int toplam = 0;
              toplam= nar+elma+armut;  // toplam miktar
              System.out.println("Manavdaki toplam meyve..:" );
              System.out.println("Toplam="+ toplam);
              }
}
```

1. Eğer aynı örnekte toplam değişkeninin veri tipini byte olarak değiştirsek derleyicimiz int tipinin byte veri tipine dönüştürülemeyeceğini belirten aşağıdaki gibi bir hata mesajı verecektir.

```
Exception in thread "main" java.lang.Error: Unresolved compilation problem:
        Type mismatch: cannot convert from int to byte

        at test2.main(test2.java:8)
```

2. Eğer aynı örnekte byte türündeki nar değişkenine 127'den daha büyük bir değer aktarsaydık `byte nar=210;` benzer bir tip uyuşmazlığı hatası ile karşılaşacaktık.

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