Algoritma ve Programlama II

Fatih Özkaynak

Ders İçeriği

- Java Veri Tipleri
  - Temel veri tipleri
  - Referans veri tipleri
- Değişken tipleri
- Operatörler
- Sayılar ile Çalışmak

Temel Veri Tipleri

- Java dilinde sekiz temel veri tipi bulunmaktadır.
- Bu veri tipleri anahtar sözcükler aracılığı ile önceden tanımlanmıştır.
  - byte
  - short
  - int
  - long
  - float
  - double
  - boolean
  - char

Byte veri tipi

- Byte veri tipi ikinin tümleyeni olan işaretli 8-bit bir tamsayıdır.
- Minimum değeri -128 (-2^7) ‘dir.
- Maksimum değeri 127 (2^7 -1) ‘dir.
- Varsayılan değeri 0’dır.
- Byte veri tipi büyük dizileri saklamak için kullanılabilir. Genellikle tamsayıların yerine kullanılmaktadır. Çünkü int veri tipinden dört kat daha küçüktür.
- Örnek: byte a = 100 , byte b = -50

Short Veri Tipi

- Short veri tipi ikinin tümleyeni olan işaretli 16-bit tamsayıdır.
- Minimum değeri -32,768 (-2^15) ‘dir.
- Maksimum değeri 32,767 (2^15 -1) ‘dir
- Short veri tipi de byte veri tipi gibi kullanılabilir. Bu veri tipi int veri tipine göre iki kat daha küçüktür.
- Varsayılan değeri 0 ’dır.
- Örnek: short s = 10000, short r = -20000

Int Veri Tipi

- Int veri tipi ikinin tümleyeni olan işaretli 32-bit uzunluğunda bir tamsayıdır.
- Minimum değeri - 2,147,483,648 (-2^31)
- Maksimum değeri 2,147,483,647 (2^31 -1)
- Int genellikle tamsayıları saklamak için varsayılan veri tipi olarak kullanılmaktadır. Hafıza ile alakalı endişelerin olmadığı durumlarda tercih edilebilir.
- Varsayılan değeri 0 ‘dır.
- Örnek: int a = 100000, int b = -200000

Long Veri Tipi

- Long veri tipi ikinin tümleyeni olan işaretli 64-bit uzunluğunda tamsayıdır.
- Minimum değeri -9,223,372,036,854,775,808 (-2^63)
- Maksimum değeri 9,223,372,036,854,775,807 (2^63 -1)
- Bu veri tipi genellikle daha büyük tamsayıları saklamak için kullanılmaktadır.
- Varsayılan değeri 0L.
- Örnek: long a = 100000L, long b = -200000L

Float Veri Tipi

- Float veri tipi tek duyarlıklı 32-bit IEEE 754 kayan noktalı sayı biçimidir.
- Float is genellikle kayan noktalı sayıları göstermek için kullanılır.
- Varsayılan değeri 0.0f.
- Float veri tipi asla tam değerlerle kullanılamazlar.
- Örnek: float f1 = 234.5f

Double Veri Tipi

- Double veri tipi çift duyarlıklı 64-bit IEEE 754 kayan noktalı sayı biçimidir.
- Bu veri itipi genellikle ondalıklı değerler için varsayılan veri tipidir.
- Varsayılan değeri 0.0d ‘dir.
- Örnek: double d1 = 123.4

Boolean Veri Tipi

- Boolean veri tipi bir bitlik bilgi temsil etmektir.
- Sadece olası iki değer alabilir. Bu değerler doğru (true) ve yanlış (false) biçimindedir.
- Bu veri tipi doğru/yanlış koşullarını izleyen basit bayrak (flags) değerleri için kullanılabilir.
- Varsayılan değeri yanlış (false) ‘dir.
- Örnek: boolean one = true

Char Veri Tipi

- char veri tipi işaretli 16-bit Unicode bir karakterdir.
- Minimum değeri '\u0000' (veya 0).
- Maksimum değeri '\uffff' (veya 65,535 dahil).
- Char veri tipi herhangi bir karakteri depolamak için kullanılır.
- Örnek: char harfA ='A'

Referans Veri Tipleri

- Referans veri tipleri sınıfların oluşturulması ile tanımlanabilir.
- Referans veri tipleri nesnelere erişilerek kullanılır.
- Bu veri tipleri spesifik bir veri tipi olarak tanımlanır. Örnek: Öğrenci, Araba, Aşk
- Sınıfların nesnelerini saklamak için bu veri tipleri kullanılabilir.
- Varsayılan değeri null (boş/değersiz) ‘dur.
- Örnek: Aşk aşk = new Aşk("karşılıksız");

String a = "merhaba";

\n	Newline (0x0a)
\r	Carriage return (0x0d)
\f	Formfeed (0x0c)
\b	Backspace (0x08)
\s	Space (0x20)
\t	tab
\"	Double quote
\'	Single quote
\\	backslash
\ddd	Octal character (ddd)

Veri Tipi	Alan Büyüklüğü	Kategori
byte	8 bit	Tamsayı Tipleri
short	16 bit	
int	32 bit	
long	64 bit	
float	32 bit	Kesirli Sayı Tipleri
double	64 bit	
char	16 bit	Karakter Tipi
boolean	-	Mantıksal Tip (ture/false)

Değişken Tanımlama

- Programlarımız içinde kullanacağımız değişkenler bir isim ile depolanmaktadır.
- Her bir değişkenin tipi ve uzunluğuna özel veri tipleri ile karar verilir.
- Değişkenler kullanılmadan önce mutlaka tanımlanmalıdır
- tür – ad
- tür – ad = değer

```
int a, b, c; // a, b ve c isimli üç tamsayı değişkeni tanımlar.
int a = 10, b = 10; // başlangıç değeri atamaya örnek
byte B = 22; // B isminde bir byte tipinde bir değişken tanımlar ve ilk değer atar.
double pi = 3.14159; // PI değişkeni tanımlar.
char a = 'a'; // a isminde bir char değişkeni tanımlar ve 'a‘ değerini atar.
```

Değişken Türleri

- Java dilinde üç farklı tipte değişken tanımlanabilir.
  - Yerel (Local) değişkenler
  - Örnek/Somut (Instance) değişkenler
  - Sınıf (Class/static) değişkenler

Yerel (Lokal) Değişkenler

- Lokal değişkenler; metotlar, yapıcılar ve bloklar içerisinde kullanılmaktadır.
- Lokal değişkenler; metotlar, yapıcılar ve bloklar içerisine girildiği zaman oluşturulurlar ve çıkıldığında sonlandırılırlar.
- Erişim belirleyicileri lokal değişkenlerde kullanılmaz.
- Lokal değişkenler sadece metotlar, yapıcılar ve bloklar tanımlandığında görülür (visible) olur.
- Lokal değişkenler yığıt (stack) seviyesinde uygulanır.
- Lokal değişkenler için varsayılan bir değer yoktur. Bu yüzden lokal değişkenlere ilk defa kullanılmadan önce mutlaka bir ilk değer atanmak zorundadır.

Lokal Değişkenler

```
public class Test{
   public void Yas(){
      int yas = 0;
      yas = yas + 7;
      System.out.println("fatih yaşı : " + yas);
   }

   public static void main(String args[]){
      Test test = new Test();
      test.Yas();
}
```

Örnek (Instance) Değişkenler

- Örnek değişkenler bir sınıf içerisinde tanımlanır.
- Örnek değişkenler metot, yapıcı ve bloklar içerisinde tanımlanmazlar.
- Bir nesne için kuyruk (heap) içerisinde yer ayrıldığından her bir örnek değişken oluşturulur.
- Örnek değişkenler bir nesne 'new' anahtar sözcüğü ile tanımlandığıı zaman oluşturulur ve nesne sonlandırıldığı zaman sonlanır.
- Örnek değişkenler bir ilk değer tutmaktadır.
- Örnek değişkenler için erişim belirleyicileri verilebilir.
- Örnek değişkenler sınıf içerisindeki tüm metot, yapıcı ve bloklar için görünürdür. Normal olarak bu değişkenlerin erişim düzeyinin gizli (private) olması önerilmektedir. Fakat alt sınıflar için görünümü sağlayabilmek için erişim belirleyicileri kullanılabilir.
- Örnek değişkenler bir varsayılan değere sahiptir. Yapıcı metotlar çalıştığı zaman bu değerler atanır.
- Örnek değişkenlere sınıf içerisinden doğrudan erişilebilir. Fakat statik metotlar ve farklı sınıflardan erişim için ObjectReference.VariableName biçimi kullanılmalıdır.

```
import java.io.*;
public class Personel{
   // bu örnek değişkene herhangi bir çocuk sınıf erişebilir.
   public String isim;

   // ucret değişkenine sadece Personel sınıfından
erişilebilir.
   private double ucret;

   // yapıcı metot tarafından isim değişkenine ilk değer
atanıyor.
   public Personel (String personelAd){
      isim = personelAd;
   }
   // ucret değişkenine değer atanıyor.
   public void ataUcret(double personelMaas){
      ucret = personelMaas;
   }

// Bu metot bilgileri ekrana yazıyor
public void goster(){
   System.out.println("isim  : " + isim );
   System.out.println("ücret :" + ucret);
}

public static void main(String args[]){
   Personel per1 = new Personel("Fatih");
   per1.ataUcret(1000);
   per1.goster();
}
```

Class/Static (Sınıf/Statik) Değişkenler

- Sınıf değişkenler de statik değişkenler olarak bilinir ve static anahtar sözcüğü ile tanımlanır.
- Her sınıfın o sınıftan ne kadar nesne oluşturulduğuna bakılmaksızın bir sınıf değişkeni koyası tutulur.
- Statik değişkenler nadiren sabit (constants) olarak kullanılırlar.
- Sabitler (constants) public/private, final ve static olarak tanımlanabilir.
- Sabit değişkenlerin ilk değerleri asla değişmez.
- Statik değişkenler statik bellek içerisinde depolanır.
- Statik değişkenler program başladığı zaman oluşturulur. Program durduğu zaman sonlandırılır.
- Görünürlüğü örnek değişkenler gibidir. Fakat çoğu statik değişken sınıf kullanıcıları için erişilebilir olsun diye public tanımlanmaktadır.
- Varsayılan değerleri örnek değişkenlerde olduğu gibidir.
- ClassName.VariableName biçiminde statik değişkenlere erişilebilir.

```
import java.io.*;

public class Personel{
   // ucret değişkeni private static bir değişkendir
   private static double ucret;

   // ETIKET bir constant değişkendir.
   public static final String ETIKET = "Yazılım Mühendislerinin";

   public static void main(String args[]){
      ucret = 1000;
      System.out.println(ETIKET + "ortalama ücretini " + ucret + « TL artır");
   }
}
```

Modifier (Değiştiriciler / Belirleyiciler)

- Java dilinde iki tip belirleyici bulunmaktadır.
  - Erişim belirleyiciler (Access Control Modifiers) – Inheritance (Kalıtım)
    - Erişim belirleyicisinin kullanılmadığı durumda public erişim belirleyicisi kullanılmaktadır.
    - Sadece sınıf içerisinden erişebilmek için (private).
    - Her yerde görünür (visible) olması için (public).
    - Paket ve alt sınıflar tarafından görünür olabilmesi için (protected).
  - Diğer belirleyiciler (Non-Access Control Modifiers)
    - static modifier: sınıf metotlar ve değişkenler için oluşturulur.
    - final modifier: sınıf, metot ve değişkenlerin uygulanmasını sonlandırır.
    - abstract modifier: soyut (abstract) sınıflar ve metotlar oluşturmak için kullanılır.
    - synchronized ve volatile modifiers: thread (iş parçacıkları) için kullanılır.

```
public class sinifAdi { // ... }

private boolean kontrolDegeri;

static final double weeks = 9.5;

protected static final int KUTUGENISLIGI = 42;

public static void main(String[] arguments) { // metot gövdesi }
```

Operatörler

- Java değişkenleri manipüle etmek için zengim bir operatör kümesi sunmaktadır. Bu operatörler aşağıdaki gibi gruplara ayrılabilir.:
  - Aritmetik operatörler
  - ilişkisel operatörler
  - Bit tabanlı operatörler
  - Mantıksal operatörler
  - Atama operatörleri
  - Diğer operatörler (? :) (instance of)

SR.NO	Operatörler ve Örnekleri A=10 ve B=20
1	**+ (Toplama)** Operatörün her iki tarafındaki değişkenler toplanır. **Örnek:** A + B işlemi sonucunda 30 değeri elde edilir.
2	**- (Çıkarma)** Operatörün solundaki değişkenden sağındaki çıkarılır. **Örnek:** A - B işlemi sonucunda -10 değeri elde edilir.
3	*** (Çarpma)** Operatörün her iki tarafındaki değerler çarpılır **Örnek:** A * B işlemi sonucunda 200 değeri elde edilir.
4	**/ (Bölme)** Operatörün solundaki değişken sağındakine bölünür **Örnek:** B / A işlemi sonucunda 2 değeri elde edilir.
5	**% (Mod)** Operatörün solundakinin sağındakine bölümünden kalanını verir. **Örnek:** B % A işlemi sonucunda 0 değeri elde edilir.
6	**++ (Artırma)** Operatör değişkenin değerini 1 artırır. **Örnek:** B++ işlemi sonucunda 21 değeri elde edilir.
7	**-- (Azaltma)** Operatör değişkenin değerini 1 azaltır. **Örnek:** B-- işlemi sonucunda 19 değeri elde edilir.

No	Operatörler ve Örnekleri A=10 ve B=20
1	**== (eşit mi)** Değişkenlerin değerlerinin eşit olup olmadığını kontrol eder. Eşit ise koşul true olur. **Örnek:** (A == B) için sonuç: true.
2	**!= (eşit değil mi)** Değişkenlerin değerlerini kontrol eder. Eşit değil ise koşul true olur. **Örnek:** (A != B) için sonuç: true.
3	**> (büyük mü)** Değişkenlerin değerlerini kontrol eder. Soldaki değişken büyük ise koşul true olur. **Örnek:** (A > B) için sonuç: false.
4	**< (küçük mü)** Değişkenlerin değerlerini kontrol eder. Soldaki değişken küçük ise koşul true olur. **Örnek:** (A < B) için sonuç: true.
5	**>= (büyük eşit mi)** Değişkenlerin değerlerini kontrol eder. Soldaki değişken büyük veya eşit ise koşul true olur. **Örnek** (A >= B) için sonuç: false.
6	**<= (küçük eşit mi)** Değişkenlerin değerlerini kontrol eder. Soldaki değişken küçük veya eşit ise koşul true olur. **Örnek** <= B) için sonuç: true.

Bit Operatörleri

a = 60 ve b = 13 değişkenlerini önce binary olarak yazalım

a = 0011 1100

b = 0000 1101

a&b = 0000 1100

a|b = 0011 1101

a^b = 0011 0001

~a = 1100 0011

SNo	Operatör ve Tanımlar A=60 ve B=13
1	**& (bitwise ve)** Mantıksal AND işlemi uygulanır. **Örnek:** (A & B) B işlemi sonucunda 12 değeri elde edilir. Yani 0000 1100
2	**| (bitwise veya)** Mantıksal OR işlemi uygulanır. **Örnek:** (A | B) B işlemi sonucunda 61 değeri elde edilir. Yani 0011 1101
3	**^ (bitwise XOR)** Mantıksal XOR işlemi uygulanır. **Örnek:** (A ^ B) B işlemi sonucunda 49 değeri elde edilir. Yani 0011 0001
4	**~ (bitwise tümleyen)** Değişkenin bire tümleyeni alınır. **Örnek:** (~A) B işlemi sonucunda -61 değeri elde edilir. Yani 100 0011
5	**<< (sola kaydır)** Operatörün solundaki değişken sağındaki değer kadar sola kaydırılır. **Örnek:** A << 2 B işlemi sonucunda 240 değeri elde edilir. Yani 1111 0000
6	**>> (sağa kaydır)** Operatörün solundaki değişken sağındaki değer kadar sağa kaydırılır. **Örnek:** A >> 2 B işlemi sonucunda 15 değeri elde edilir. Yani 1111
7	**>>> (sıfıra tamamla sağa kaydır)** Operatörün solundaki değişken sağındaki değer kadar sağa kaydırılır. Kaydırılan değerlerin yerine sıfır yazılır. **Örnek:** A >>>2 B işlemi sonucunda 15 değeri elde edilir. Yani 0000 1111

Mantıksal Operatörler

No	Tanımlar A=true B=false
1	**&& (mantıksal ve)** Mantıksal veya işlemi uygulanır. **Örnek** (A && B) işlemi sonucunda elde edilen değer: false.
2	**|| (mantıksal veya)** Mantıksal veya işlemi uygulanır. **Örnek** (A || B) işlemi sonucunda elde edilen değer: true.
3	**! (mantıksal değil)** Değişkenin tersi alınır. **Örnek** !(A && B) işlemi sonucunda elde edilen değer: true.

	Atama Operatörleri ve Tanımları
1	C = A + B ataması ile C = A + B
2	C += A ataması ile C = C + A
3	C -= A ataması ile C = C – A
4	C *= A ataması ile C = C * A
5	C/= A ataması ile C = C / A
6	C %= A ataması ile C = C % A
7	C <<= 2 ataması ile C = C << 2
8	C >>= 2 ataması ile C = C >> 2
9	C &= 2 ataması ile C = C & 2
10	C ^= 2 ataması ile C = C ^ 2
11	C |= 2 ataması ile C = C | 2

variable = (expression) ? value true : value false

```
public class Test {

   public static void main(String args[]){
      int a, b;
      a = 10;
      b = (a == 1) ? 20: 30;
      System.out.println( " b’nin değeri : " +  b );

      b = (a == 10) ? 20: 30;
      System.out.println( " b’nin değeri : " + b );
   }
}
```

instance of Operatorü

- Nesne referans değişkenleri için kullanılabilecek bir operatördür.
- Operatörün sol tarafındaki değişkenin sınıf/arayüz tipinde olup olmadığını kontrol etmektedir.

```
public class Test {

   public static void main(String args[]){
      String isim = "Fatih";
      boolean sonuc = isim instanceof String;
      System.out.println( sonuc );
   }
}
```

Örnek

```
class Arac {}

public class Araba extends Arac {
   public static void main(String args[]){
      Arac a = new Araba();
      boolean sonuc =  a instanceof Araba;
      System.out.println( sonuc );
   }
}
```

Operatör Önceliği

- Postfix () [] .
- Unary ++ - - ! ~
- Multiplicative * / %
- Additive + -
- Shift >> >>> <<
- Relational > >= < <=
- Equality == !=
- Bitwise AND &
- Bitwise XOR ^
- Bitwise OR |
- Logical AND &&
- Logical OR ||
- Conditional ?:
- Assignment = += -= *= /= %= >>= <<= &= ^= |=

Sayılar ile Alakalı Metotlar

- xxxValue():
- compareTo():
- equals():
- valueOf():
- toString():
- parseInt():
- abs():
- ceil():
- floor():
- rint():
- round():
- min():
- max():
- exp():
- log():
- pow():
- sqrt():
- sin():
- cos():
- tan():
- asin():
- acos():
- atan():
- atan2():
- toDegrees():
- toRadians():
- random():

Karakterler

```
char ch = 'a';

// Unicode yunanca omega karakteri
char uniChar = '\u039A';

// bir karakter dizisi
char[] charArray ={ 'a', 'b', 'c', 'd', 'e' };

Character ch = new Character('a');
```

Karakterler ile Alakalı Metotlar

- isLetter()
- isDigit()
- isWhitespace()
- isUpperCase()
- isLowerCase()
- toUpperCase()
- toLowerCase()
- toString()