◀ ተመለሥ ▲ማውጫ ቀጥል ▶

Print Page

ምዕራፍ ፯

መደባት (Classes)

7.1 ምዕላደ-ቃላት

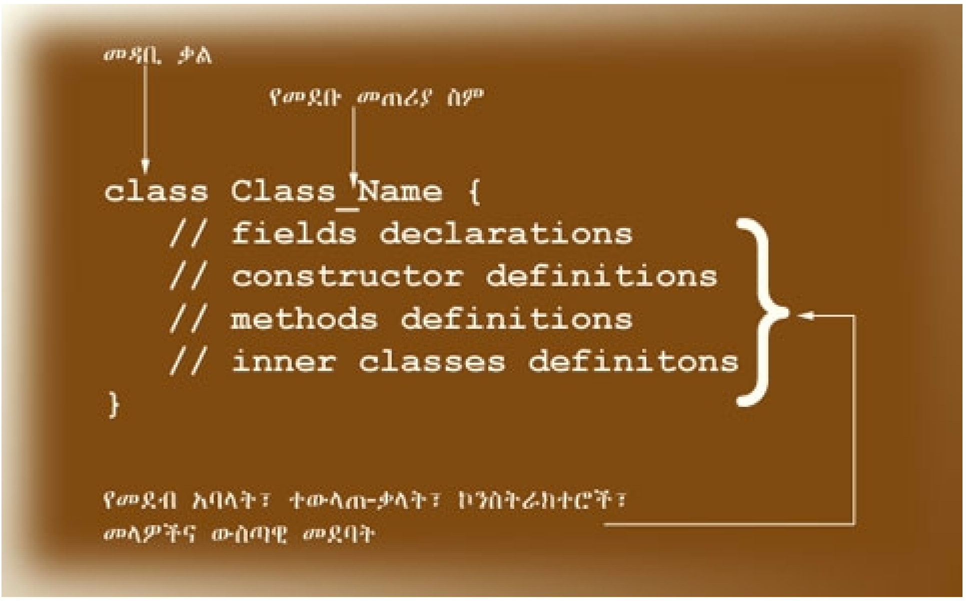
አማርኛ ቃል	እንግሊዘኛ ቃል
መደብ	class
ተውላጠ-ቃል	variable
መላ/ፋንክሽን	method/function
ኮንስትራክተር	constructor
የመደብ ርቢ	object
ረቂቅ	abstract
እስታቲክ	static
ስመአልባ	anonymous

7.2 መግቢያ

ከጃቫ የዴታ ዓይነታት መካከል አንዱ መደብ መሆኑን በምዕራፍ ፫ ተጠቊሟል። በዚህ ክፍል ስለ መደብ በሰፊው እንወያያለን። መደብ/ባት ስንል የጃቫ ፕሮግራም ፥ የጃቫ ፕሮግራም ስንል መደብ/ባት ማለታችን ነው። መደባት የጃቫን መዋቅርና ቅርጽ እንዲሁም ገጸ-ባሕሪይና ተግባራት ይደነግጋሉ።

በዚህ ክፍል የምናተኩረው በመደብ አባላቶች ላይ ነው። ስለመደብ ዓይነቶችና ግንኙነታቸው በሚቀጥለው ምዕራፍ እንደርስበታለን።

የመደብ ቃል አገባብ፦



```
class Class_Name {
    // fields declarations
    // constructor definitions
    // methods definitions
    // inner classes definitons
}
```

7.3 የመደብ ዓይነታት

የጃቫ የመደብ ዓይነቶች በአራት መልክ ሊታዩ ይችላሉ። በተጨባጥ የመደብ ዓይነት ላይ አተኩረን ሌሎቹን በመጥቀስ እናልፋለን።

1. **ተጨባጭ መደባት፦** እንደዚህ ዓይነት መደባት ለሥራ ዝግጁ ናቸው። አራብቶ በሥራ ላይ ማስማራት ይፈቀዳል። እስካሁን ድረስ በዚህ ጥናት ለምሳሌ የቀረቡት መደቦች በሙሉ ተጨባጭ ናቸው።
2. ረቂቅ መደባት
3. ውስጣዊ መደባት
4. ስምአልባ መደባት

7.4 የመደብ አባላት

የመደብ አባላት ፥ ባሕሪይን ፥ ተግባራትንና ፥ ግንኙነትን ይገልጻሉ። የመደቡን ምን ማንነት ይመሰርታሉ። የአባላቱ ዓይነትና መጠን በአንድ በኩል እንደ ችግሩ ፥ በሌላ እንደ ፕሮግራም ጸሐፊው ነው። የአባላት ዓይነታት እነሆ።

1. ተውላጠ-ቃላት (መስክ) (fields)
2. ኮንስትራክተሮች (Constructors)
3. መላ (ተግባራት) (functions)
4. ውስጣዊ መደባት
5. እስታቲክ ክፍል

7.5 አባል ተውላጠ-ቃላት (Fields)

አባል ተውላጠ-ቃላት የአንድን መደብ ባሕሪይ ይገልጻሉ። የመደቡን ዴታዎችንና የመደብ ርባታዎችን ይጠብቃሉ። ዓይነታቸው መሠረታዊ ዴታና የመደብ ርባታዎች ይጨምራሉ። በመደባቸው ውስጥ አባል የሆኑት መላዎች (methods) በቀጥታ ማየትና መጠቀም ይችላሉ።

አባል ተውላጠ–ቃላት

```
public class SimpleTokenizer {

   /* Fields: class variables declaration */
   String text        = "" ;
   String delimiter   = "" ;
   int current        = -1 ;
   int prev           = 0 ;

   // ...
}
```

ይህ መደብ አራት ተውላጠ-ቃል አባላት አሉት። እንዚህን ተውላጠ-ቃላት ልዩ ልዩ ዴታዎች ለመጠበቅ ፥ ለማቆየት ፥ ለማሻሻል ፥ ለመጻፍና ለመሠረዝ የሚገለገላቸው ናቸው። ሁለቱ የString መደብ ርበታ ሲሆኑ የቀሩት ግን የኢንትጀር መሠረታዊ ዴታ

ዓይነት ናቸው። በተጨማሪ ሁሉም መነሻ ዋጋ ተሰይመዋል ፤ ነገር ግን ይህ የግድ መደረግ የለበትም ፤ አማራጭ ነው። አንድ ተውላጠ-ቃል ሲፈጠር ፥ መነሻ ዋጋ ቢሰይምም ባይሰየምም ፥ ጃቫ በራሱ መነሻ ዋጋ ሁሉጊዜ አስቀድሞ ይሰይማል። ተውላጠ-ቃላቱን በማጽዳት ለሥራ እንደማዘጋጀት።

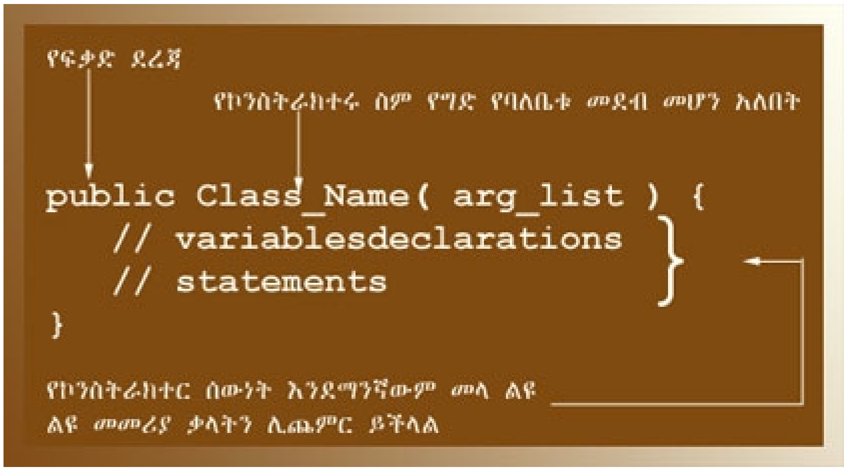
7.6 ኮንስትራክትሮች (Constructors)

እስካሁን ድረስ ባሉት የጥናት ክፍሎች ኮንስትራክተሮችን በቀጥተኛ መንገድ መደቦች ውስጥ አስገብተን አልተጠቀምንም። በሌላ በኩል ግን ፥ ማንኛውንም መደብ ያለኮንስትራክተር መገንባትና መጠቀም በፍጹም አይቻልም። ይሁን እንጂ ፥ እስካሁን ድረስ የተሰጡት ምሳሌዎች በሚገባ ሠርተዋል ፤ ይሠራሉ። ይህ እንዴት ሊሆን ይችላል? እውነቱ ምንድን ነው? ይኸን ጥያቄ ለመመለስ መጀመሪያ ኮንስትራክተር ምንድን ነው የሚለውን ጥያቄ እንመርምር።

በጃቫ ሕግ መሠረት የማንኛውም መደብ ርቢዎች (objects) ሲፈጠሩ ፥ አንድ ጊዜ ብቻ ራሳቸውን የሚጠሩ «ልዩ መላዎች» ናቸው። ኮንስትራክተሮችና መላዎች በመሠረቱ ተመሳሳይ ናቸው ፤ ነገር ግን ጥቂትና ጥብቅ ልዩነት አላቸው።

ኮንስትራክተሮች የመደብ ርቢዎች ቅድሚያዊ ዝግጅት እንዲያደርጉ ዕድል ይሠጣሉ። በነሱ ውስጥ ፥ የመደብ ርቢዎች የግል ተውላጠ-ቃላታቸውን መነሻ ዋጋ ይሰይማሉ ፤ ራሳቸውን ለሥራ ያዘጋጃሉ ወይም ቀዳሚ ተግባራትን ይፈጽማሉ።

የኮንስትራክተሮች ቃል አገባብ፦



```
public Class_Name( arg_list ) {
    // variablesdeclarations
    // statements
}
```

የፍቃድ ደረጃ
የኮንስትራክተሩ ስም የግድ የባለቤቱ መደብ መሆን አለበት
የኮንስትራክተር ሰውነት እንደማንኛውም መላ ልዩ ልዩ መመሪያ ቃላትን ሊጨምር ይችላል

እንደመደቦች የኮንስትራክተሮች ሚና ይለያያል። አንድ መደብ በኮንስትራክተሩ ውስጥ ምንም ሥራ ማካሄድ ካልፈለገ ፥ የኮንስትራክተሩን ሰውነት ባዶ መተው አንዱ አማራጭ ነው።

ኮንስትራክተር

```
public class SimpleTokenizer {

   /* Fields: class fields declaration */
   String text        = "" ;
   String delimiter   = "" ;
   int current        = -1 ;
   int prev           = 0 ;

   /* Constructor: intializes fields */
   SimpleTokenizer(String text, String delimiter) {
      this.text = text ;
```

```
        this.delimiter = delimiter  ;
    }
}
```

ይህ ምሳሌ እንደሚያሳየው ፣ ኮንስትራክተሮች ስማቸውን ከባለቤት መደባቸው ይወርሳሉ። ይህ ኮንስትራክተር ሁለት ዕሴቶችን ተቀብሎ ሁለቱን አባል ተውላጠ-ቃላት ያድሳል። ወደ ሥራ ስሚሪቱ በጥሪ ላይ የተመሰረተ ሆኖ ጥሪው የሚመጣው መደቡ ሲራባ ብቻ ነው። ተከታዩ ቃል አንድ የኮንስትራክተር ጥሪ ያመነጫል።

```
SimpleTokenizer st = new SimpleTokenizer("a+b", "+") ;
```

ከላይ ላነሳነው ጥያቄ መልሱ ይህ ነው። አንድ መደብ የራሱን ኮንስትራክተር ካልደነገገ ፣ የጃቫ ኮምፓይለር የራሱን ኮንስትራክተር ይከታል። ይህ ኮንስትራክተር በእንግሊዘኛ **default constructor** ብለው ይጠሩታል።

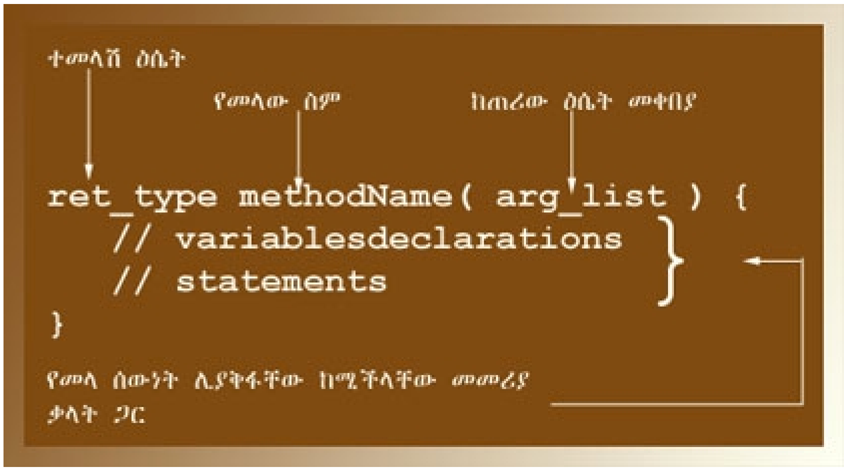
ስለኮንስትራክተሮች ያጻጻፍና ያሠራር ሕግ ጥቂቶቹን ባቻ መንካታችን ግልጽ ቢሆንም ፣ መዘለል ከሌላባቸው መካከል አንዱ ኮንስትራክተሮችን በቀጥታ መጥራት በፍጹም አይፈቀድም። በተደጋጋሚ እንደተጠቀሰው ፣ በአንድ የመደብ ርቢ ሕይወት ውስጥ አንድ ጊዜ ብቻ ራሳቸውን ይጠራሉ። እሱም በመደቡ የርባታ ሂደት ውስጥ ነው።

7.7 መላዎች (Methods/Functions)

መደባዊ የሆኑ የፕሮግራም መጻፊያ ቋንቋዎች ፣ ተከታታይ መመሪያ በሥራ ላይ አዋይ አካሎቻቸውን መላዎች (methods/functions/operations) ብለው ይጠሯቸዋል።

መደቦች ያለመላዎች በዙ ርቀው መሄድ አይችሉም። በአባል ተውላጠ-ቃላት ባሕሪያቸውን ይገልጻሉ ፤ ነገር ግን ተግባሮቻቸውን በመላዎቻቸው ያከናውናሉ። ከሞላ ጐደል የጃቫ ፕሮግራም ሥራዎች የሚከናወኑት በመላዎች ነው።

የመላዎች ቃል አገባብ፦


```
ret_type methodName( arg_list ) {
   // variablesdeclarations
   // statements
}
```

መላዎች የራሳቸው የሆነ ስም ፣ ሰውነት ፣ መገናኛ ዘይቤ አላቸው። ለሥራ ሁልጊዜ በስም መጠራት አለባቸው። ሥራቸውን ካጠናቀቁ በኋላ ፣ ሂደቱን ለጠሪው ይመልሳሉ። ከሌላ መደብ ሆኖ ለመጥራት ፣ ባለቤት መደባቸውን ማራባት ወይም በሱ ሥር መሄድ የግድ ነው። መላዎች ዴታ መረከብና መመለስ ይችላሉ።

መላዎች ሁልጊዜ በመደብ ሥር መኖር አለባቸው። ራሳቸውን ችለው መቆም አይፈቀድላቸውም። ያሉበት መደብ ባለቤታቸው ነው። አንድ መደብ ዜሮ ወይም ብዙ መላዎች የመደንገግ ችሎታው የተጠበቀ ነው።

መላዎች

```
public class SimpleTokenizer {
   /* Fields: class fields declaration */
   String text       = "" ;
   String delimiter  = "" ;
   int current       = -1 ;
   int prev          = 0 ;

   /* Constructor: intializes fields */
   SimpleTokenizer(String text, String delimiter) {
      this.text = text ;
      this.delimiter = delimiter  ;
   }

   /* Method: Returns the next token */
   String nextToken() {
      current = getIndex(current + 1) ;   // get the del's index
      if (current > -1) {                 // if exist, get token
         String newToken = text.substring(prev, current) ;
         prev = current + 1 ;             // advance prev
         return newToken ;                // return new token
      }
      return null ;                       // no more token
   }

   /** Method: Returns the index of a delimiter if found */
   int getIndex(int from) {
      for (int i=0; i < delimiter.length(); i++) {
         int pos = text.indexOf(delimiter.charAt(i), from) ;
         if (pos > -1)
            return pos ;
      }
      return -1 ;
   }

   /** Method: An entry point for program execution */
   public static void main(String[] args) throws Exception {
      String expression = "a+b-c*d/e" ;
      String operators  =  "+-*/" ;
      SimpleTokenizer st = new SimpleTokenizer(expression,operators) ;

      String token = st.nextToken() ;
      int i = 2;
      while (token != null) {
         System.out.println("Token: " + token) ;
         token = st.nextToken() ;
         i-- ;
      }
   }
}
```

Download: SimpleTokenizer.java

ምንም እንኳን ይህ ኮድ ረዥም ይሁን እንጂ ፣ የመላዎችን አጻጻፍና አቋቋም ፣ እንዲሁም አጠራር ያሳያል። ሦስት መላዎች፦ **`nextToken(..)`** ፣ **`getIndex(..)`** ፣ እና **`main(..)`** አሉ። የ`main(..)` መላ የፕሮግራሙ መነሻ ወይም መንደርደሪያ ቃል ነው።

በ`static` ቃል የተደነገጉ መላዎች ባለቤት መደባቸው ርቢ ውስጥ ሳይገባ ለጥሪ ዝግጁ ናቸው። ለምሳሌ ፣ የ`Math` መደብ አያሌ የእስታቲክ መላዎች አሉት ፤ ስለዚህ የሱን ርቢ ሳናወጣ በቀጥታ ቀጥሎ እንደሚታየው የእስታቲክ መለዎችን መጥራት እንችላለን።

```
double result = Math.exp(10) ;
```

የሜይን **(`main`)** መላ እስታቲክ በመሆኑ ፣ እስታቲክ ያልሆኑትን መላዎች በቀጥታ መጥራት አይፈቀድለትም። «ያለእስታቲክ» መላዎች ግልጋሎታቸውን የሚለግሱት ለመደብ ርቢዎች ብቻ ነው። ስለሆነም የሜይን መላ ጥሪ ከማቅረቡ በፊት የመደብ ርቢ ማውጣት ነበረበት።

ከዛ በኋላ ነበር የሜይን መላ ሌሎቹን መላዎች መጥራት የቻለው። ያጠራሩ ቃል ከላይ እንደሚታየው እንዲህ ነበር ።

```
token = st.nextToken() ;
```

«ያለእስታቲክ» መላዎች እርስበራሳቸው ይጠራራሉ። «ባለእስታቲክ» መላዎችም እንደዚሁ። ባለእስታቲክ መላዎች ያለእስታቲክ መላዎችን በቀጥታ መጥራት አይፈቀድላቸው እንጂ ፣ ያለአስታቲኮቹን ግን መጥራት ይፈቀድላቸዋል። ለምን የሚለውን ጥያቄ ለአንባቢው።

7.8 ፍቃዶችና ዓይነታቸው

ማንኛውም የመደብ አባል የፍቃድ ደረጃ አለው። ፍቃዱ የአባሉን እይታ እስከየት ድረስ እንደሆነ ይወስናል። አራቱ የፍቃድ ዓይነቶችና እይታቸው በሚከተለው ሠንጠረዥ አለ።

የፍቃድ ዓይነታት

ፍቃዶች	**መደብ**	**ፓኬጅ**	**ንኡስ መደብ**	**ዓለም**
`private`	ይታያል	አይታይም	አይታይም	አይታይም
`no specifier`	ይታያል	ይታያል	አይታይም	አይታይም
`protected`	ይታያል	ይታያል	ይታያል	አይታይም
`public`	ይታያል	ይታያል	ይታያል	ይታያል

Source for this table

ዓላማው ሌሎች መደቦች የአንድን መደብ አባላት እስከየት ደረጃ ድረስ ማየትና መጠቀም እንደሚችሉ ለመቆጣጠር ነው። ይህ ራሱን ለውጭ መደቦች አሳልፎ የማይሰጥ ፕሮግራም ለመጻፍ ይረዳል። የቅንነት ስህተትን ይቀንሳል።

የፍቃድ ዓይነታት

```
/**
 * A class that denies object instantiation
 */
class Neighbor {
   /* Fields: class fields declaration */
   private static Neighbor instance = null ;
   /* Private cconstructor, thus no obj. inst. allowed */
   private Neighbor() {
      super() ;
   }

   /* Returns instance of itself */
   public static Neighbor getInstance() {
      if (instance == null)
         instance = new Neighbor() ;
      return instance ;
   }

   /** Prints a friendly message */
   public void print() {
      System.out.println("Hola Neighbor!") ;
   }
}

/**
 * A main class that tries to use Neighbor
 */
public class AccessControl {
   /** Method: An entry point for program execution */
   public static void main(String[] args) throws Exception {
      Neighbor n = new Neighbor() ; // No! But why?
      // Neighbor n = Neighbor.getInstance() ;   // Ok!
      // n.print() ;                             // Ok!
   }
}
```

Download: AccessControl.java

ይህ ምሳሌ ሁለት መደቦች አሉት። የNeighbor መደብ ጥብቅ የፍቃድ ዓይነት ይጠቀማል። የውጭ መደቦች ይኸንን መደብ ማራባት በፍጹም አይችሉም ፤ ነገር ግን ፥ ራሱ የሚፈጥረውን ርቢ መጠየቅና ያንን ብቻ መጠቀም ይችላሉ። አንባቢው በቅርብ ኮዱን ይመረምረው ዘንድ ዝርዝሩ ተዘሏል።

◀ ተመለሥ ▲ ማውጫ ቀጥል ▶

To contact: info@senamirmir.com

Copyright © 2002-2005 Senamirmir Project