

Java ile Nesne Merkezli Programlamaya Giriş

6. Bölüm Diziler

Akın Kaldırođlu
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 bilgilendirme 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/İnternet 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.

➤ İyi Java'lı günler dilerim. www.selsoft.academy

Gündem

- Bu bölümde en temel torba (collection) yapısı olan diziler (arrays) ele alınacaktır.
- Ayrıca
 - Tek boyutlu ve çok boyutlu diziler ile
 - Arrays sınıfı
 - main metoda parametre geçme
- incelenecektir.



Anahtar Kelimeler

Torbalar (Collections)

- Bütün dillerde olduğu gibi Java'da da, birden fazla basit değişkeni ya da nesneyi yönetmeyi sağlayan **torba (collection)** yapıları vardır.
- Java'nın **java.util** paketinde çok yetenekli bir **torba çerçevesi (collection framework)** vardır,
 - Bu torbaları ileride ayrı bir bölümde ele alacağız.
- Bu bölümde, bu yetenekli torbalardan önce, en temel torba yapısı olan, **diziyi (array)** ele alacağız

Dizi (Array)

- Java'da diziler, **belli sayıda** ve **aynı tipten** yani **homojen** olan elemanları (basit ya da referans) sıralı bir şekilde tutan veri yapılarıdır.
- Java'da diziler, nesnedirler.
- Dizilerin iki kısıtı vardır:
 - Uzunlukları sabittir ve bu bilgi oluşturulurken bilinmelidir.
 - Tek bir tipten olan elemanlar saklayabilirler, yani diziler, homojen veri yapılarıdır.
- Bu iki kısıttan dolayı diziler hızlıdır ama kullanımlarında sıkıntıya sebep olabilirler.
- Diziler, basit ya da referans tipten verileri tutabilirler.

Dizi Tanıtımı

- Dizilerin tipi vardır ve bu tip aslında, dizi içinde saklanacak elemanların tipidir.
- Dizi tanıtımı, referans değişkeni tanıtımı gibidir, sadece diziyi göstermek üzere "**[]**" kullanılır.
 - "**[]**" işaretini nerede olduğunun önemi yoktur:

```
ElementType[] arrayName;  
ya da  
ElementType [] arrayName;  
ya da  
ElementType []arrayName;  
ya da  
ElementType arrayName[];
```

Dizi Tanımlama

- Diziyi tanımlamak için dizinin boyutuna ihtiyaç vardır.
- Dizinin boyutu "int" tipinde (long olamaz) bir sabite ya da değişkendir.
 - Bir dizi en az "0" boyutunda olabilir.
- Bu şekilde belirtilen sayıda odaya/hücreye (cell) sahip olan bir dizi nesnesi oluşturulur.
- Kurucu çağrısı tanımda yapılmaz, bu çağrıyı JVM halleder.

```
ElementType[] arrayName = new ElementType[noOfElements];  
ElementType arrayName[] = new ElementType[noOfElements];
```

```
int[] intArray = new int[20];  
Pizza[] pizzas = new Pizza[5];  
Student students[] = new Student[5000];
```


Dizi Elemanlarının İlk Değeri

- Bir dizi oluşturulduğunda, odacıklardaki elemanlar, dizinin tipinin varsayılan değerine sahip olur.
 - Bu değer **boolean** için "**false**", diğer yedi sayısal basit tip için ise "**0**" in bir versiyonudur.
 - Referans tipler için ise, varsayılan değer "**null**" dır.
- Dolayısıyla bir dizi nesnesi oluşturmak ile o dizinin odacıklarının içine değer atamak farklı şeylerdir.
- Sağlıklı bir dizi yapısı için dizinin odacıkları varsayılan değerde bırakılmamalı ve ilk değer atanmalıdır.

Dizi Nesneleri

- Diziler birer nesnedirler.
- Tanımlanan dizi değişkeni de aslında, dizi nesnesini gösteren bir referanstır.
- Diziler üzerinde uzunluğunu yani dizinin oda sayısını veren **int** tipinde bir alan bulunur:
 - *length*

Dizi Elemanlarına Erişim

- Dizi elemanlarına, dizinin oda numarası ya da indisi (index) ile erişilir.
- Oda numarası, "0"dan başlar ve dizinin uzunluğunun bir eksiğine kadar (**length-1**) devam eder.

```
int i = intArray[intArray.length - 1]; // i => 0  
Pizza pizza = pizzas[2]; // null  
pizzas[0] = new Pizza();
```

Diziye İlk Değer Atama I

- Dizideki odacıklara ilk değer atamanın ilk yolu, odacıklara tek tek indisleriyle ulaşarak atama yapmaktır.
 - Bu durumda **for** ya da **while** döngüsü kullanılır.
- Oda numarasıyla odalara tek tek ulaşılırken sınır değer olan **length-1**'e dikkat edilmesi gereklidir.

```
Random r = new Random();  
for (int i = 0; i < intArray.length; i++) {  
    int randomInt = r.nextInt();  
    int sayi = randomInt % 100;  
    intArray[i] = sayi;  
}
```

Diziye İlk Değer Atama II

- Dizi oluşturulurken içinde saklanacak veriler biliniyorsa, "{}" içinde virgül ile ayrılarak sıralı olarak verilebilir.
- Bu tip ilk değer vermede "new" anahtar kelimesi kullanılmaz.
- Java'da sadece 3 çeşit nesne "new" anahtar kelimesi kullanılmadan oluşturulabilir: dizi ve java.lang.String

```
ElementType[] arrayName = { initial values };  
  
int[] array = {1,2,3,4,5,6,7,8,9,0};  
  
Pizza[] pizzalar = {new Pizza(), new Pizza(), null};  
  
boolean[] b = {true, false}
```

Gelişmiş for

- **for**'un "her biri" (**for each**) anlamında bir kullanımı daha vardır.
- Dizi gibi farklı torba yapıları üzerinde çalışıp, bir indis kullanmadan, torbadaki elemanlara **tek tek ulaşmak** (**iteration**) için özel bir **for** yapısıdır.

```
for (type element:collection)
```

```
int[] intArray = new int[10];  
...  
for (int i:intArray)  
    System.out.println(i);
```

ArrayDemo.java

www.selsoft.academy

Oda Numaraları

- Dizilerin odalarına erişim "**int**" cinsinden değişken ya da bir sabite ile olur.
 - Oda erişiminde "**long**" kullanılamaz ama "**int**" e otomatik olarak çevrilen "**byte**", "**short**" ve "**char**" kullanılabilir.
- Dizi oluşturulurken, boyut olarak negatif rakam verilemez.
 - Bu durumda `java.lang.NegativeArraySizeException` fırlatılır.
- Dolayısıyla ancak 2^{31} odacıklı bir dizi oluşturulabilir.
- Odacıklara erişimde de "**0**"dan küçük veya **length-1**'den büyük bir indis kullanılması halinde `java.lang.ArrayIndexOutOfBoundsException` fırlatılır.

Parametre Olarak Dizi Geçme

- Dizileri metotlara parametre olarak geçmek için bir kaç farklı şekil söz konusudur.

```
int[] array1 = new int[3];  
array1[0] = 17;  
array1[1] = 22;  
array1[2] = -8;  
calculateAverage(array1);  
  
int[] array2 = {81, 19, -14};  
calculateAverage(array2);  
  
calculateAverage(new int[]{43, 25, 99});  
  
calculateAverage({43, 25, 99}); // Error!  
  
calculateAverage(new int[3]); // Default values
```

ArrayParameters.java

www.selsoft.academy

Çok Boyutlu Diziler

www.selsoft.academy

Çok Boyutlu Diziler - I

- Çok boyutlu diziler Java'da dizi içinde dizi olarak ifade edilirler.
 - Çok boyutlu diziler, ilk dizinin her bir odasına bir başka dizi koyarak elde edilir.
 - Çok boyutlu diziler, matris gibi çok boyutlu yapıları ifadede kullanılır.
- Tanımlanan esas ya da ilk dizinin boyutu verilmelidir.
 - Tanımlama sırasında içteki dizilerin boyutlarının belirlenmesine ihtiyaç yoktur.

```
ElementType[][] name = new ElementType[noOfElements][];  
  
int[][]coordinates = new int[4][];
```

Çok Boyutlu Diziler - II

- Çok boyutlu dizilerin kısa yolla da, elemanları vererek tanımlayabilirsiniz.

```
ElementType[][] name = {...}, {...}, ..., {...}
```

- Çok boyutlu dizilerin işlenmesi için boyut sayısı kadar iç içe döngüye ihtiyaç vardır.

MultiDimArray.java

www.selsoft.academy

Uygulama

- “*” kullanarak verilen bilgilerle aşağıdaki şekilleri çizen programları, dizileri kullanarak yazın. Yani “*” ve “ ”ları bir dizide tutun ve sonra diziyi yazdırın.
 - Yükseklik ve genişlik ile dikdörtgen,
 - Yükseklik ile dik üçgen,
 - Yükseklik ile eşkenar üçgen

```
* * * * *
*
*
*
*
*
*
*
*
*
*
*
* * * * *
```

```
*
***
*****
*****
*****
*****
*****
*****
*****
*****
*****
*****
*****
```

```

*
***
*****
*****
*****
*****
*****
*****
*****
*****
*****
*****
```

Arrays Sınıfı

www.selsoft.academy

Arrays Sınıfı

- `java.util` paketindeki `Arrays` sınıfı, dizilerle ilgili bir kaç tane kolaylık sağlayıcı (utility) metota sahiptir:
 - `binarySearch`
 - `copyOf`
 - `copyOfRange`
 - `fill`
 - `sort`

ArraysDemo.java

www.selsoft.academy

main Metoda Parametre Geme

www.selsoft.academy

Main Metoda Parametre Geçme

- main metoda parametre geçilebilir.
- Geçilen parametreler, main metodun argümanı olan **String** dizisinde saklanır.
- `public static void main(String[] args)` ında parametreler verilmelidir.

MainExample.java

www.selsoft.academy

Özet

- Bu bölümde veri yapılarının en basiti olan diziler ele alındı.
- Tek boyutlu ve çok boyutlu dizilerin nasıl işleneceği incelendi.
- `for`'un "`for each`" yapısı ele alındı.
- `java.util.Arrays` sınıfında bulunan kolaylık metotları çalışıldı.

Ödevler

Ödevler

- Girilen bir sayıya kadar olan asal sayıları Sieve of Eratosthenes'in algoritmasını kullanarak bulunuz.
 - Algoritma hakkında bilgi için http://en.wikipedia.org/wiki/Sieve_of_Eratosthenes adresini kullanabilirsiniz.
- Girilen bir dizideki elemanları, sıralarını bozmadan, tekil olarak konsola yazan bir program yazın.
 - Örneğin girdi {3, 3, 87, 56, 1, 87, 3, 2 } ise çıktı {3, 87, 56, 1, 2 } olmalıdır.

Ödevler - I

- Aşağıdaki arayüzü yerine getiren bir yığın (stack) sınıfı yazın:

```
// Default maximum stack size
public static final int MAX_STACK_SIZE;

// Put element on the top
public void push(String newElement) {}

// Pop element from the top
public String pop() {...}

// Remove all elements from stack
public void clear() {...}

// Stack status operations

// Is stack empty?
public boolean isEmpty() {...}

// Is stack full?
public boolean isFull() {...}

// How many elements in stack?
public int size() {...}

// Capacity of stack?
public int getCapacity() {...}

// Outputs the elements in the stack. For testing/debugging purposes only
public void showElements() {}
```

Ödevler - II

- Aşağıdaki arayüzü yerine getiren bir kuyruk (queue) sınıfı yazın:

```
// Default maximum queue size
public static final int MAX_QUEUE_SIZE;

// Insert element at the bottom
public void queue(String newElement) {}

// Pop element from the top
public String dequeue() {...}

// Remove all elements from queue
public void clear() {...}

// Queue status operations

// Is queue empty?
public boolean isEmpty() {...}

// Is queue full?
public boolean isFull() {...}

// How many elements in queue?
public int size() {...}

// Capacity of queue?
public int getCapacity() {...}

// Outputs the elements in the queue. For testing/debugging purposes only
public void showElements() {}
```