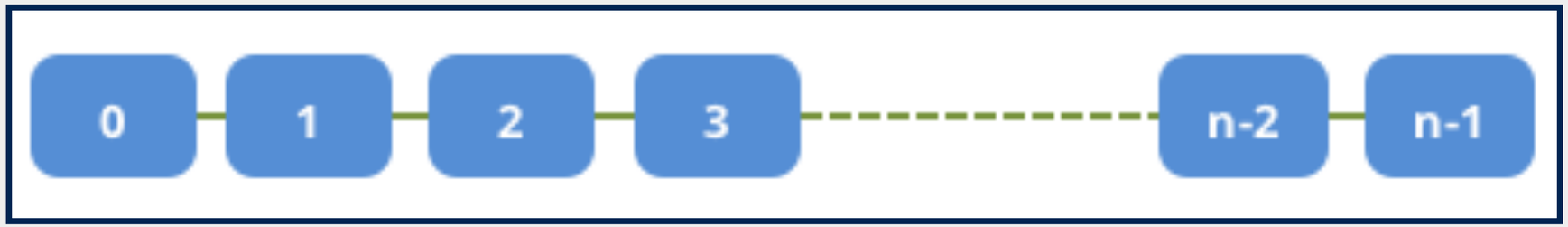
Diziler

Torbalar (Collections)

- Bütün dillerde olduğu gibi Java'da da, birden fazla ilkel değişkeni ya da nesneyi yönetmeyi sağlayan **torba** (**collection**) yapıları vardır.
- Java'nın `java.util` paketinde 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, **dizileri** (**arrays**) ele alacağız

Diziler (Arrays) - I

- Java’da diziler, belli sayıda ve aynı tipten yani homojen olan elemanları, dizili (ordered) bir şekilde tutan veri yapılarıdır (data structure).
  - Dizilerin elemanları, ilkel ya da referans tipten verileri olabilirler.
  - Diziler elemanları, **oda/hücre** (**cell**) denen yapılarda tutulur.



Diziler (Arrays) - II

- Dizilerin iki kısıtı vardır:
  - Uzunlukları sabittir ve bu bilgi dizi oluşturulurken verilmelidir,
  - Diziler, homojen veri yapılarıdır, elemanları aynı tipten olmalıdır.
- Bu iki kısıttan dolayı diziler hızlıdırlar ama zaman zaman sıkıntıya sebep olabilirler.
  - Bu kısıtları olmayan torbalar Java'nın torba çerçevesinde vardır.

Dizi Tanımlamak

Dizi Tanıtımı - I

- Dizilerin tipi vardır ve bu tip, dizinin içinde tutulacak elemanların tipidir.
- Dizi tanıtımı, referans değişkeni tanıtımı gibidir, isim ve referans değişkeni gereklidir.
  - Farklılık, diziyi göstermek üzere `[]` kullanımıdır.

```
ElementType[]  arrayName;

int[] intArray;
Pizza[] pizzas;
Student[] students;
```

Dizi Tanıtımı - II

- `[]` işaretini nerede olduğunun önemi yoktur, tipten ya da referanstan sonra olabilir.
  - Yaygın kulanım tipten sonra olmasıdır.

```
ElementType[]  arrayName;
int[] intArray;

ElementType [] arrayName;
int [] intArray;

ElementType []arrayName;
int []intArray;

ElementType arrayName[];
int intArray[];
```

Dizi Tanımlamak - I

- Diziyi tanımlamak için dizinin boyutuna ihtiyaç vardır.
  - Dizinin boyutu `int` tipinde bir sabite ya da değişkendir.
- Bu şekilde, belirtilen sayıda odaya sahip olan bir dizi oluşturulur.
  - Kurucu çağrısı tanımda yapılmaz, bu çağrıyı JVM halleder.

Dizi Tanımlamak - II

- Dizinin uzunluğu (length, size) **0** ya da **pozitif tam** sayı olmalıdır.

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

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

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

Dizi Elemanlarının İlk Değeri

- Bir dizi oluşturulduğunda, odalarındaki elemanlar, dizinin tipinin varsayılan değerine sahip olur.
  - Bu değer `boolean` için `false`, diğer yedi sayısal ilkel tip için ise **0**'ın bir türüdür.
  - Referans tipler için ise varsayılan değer `null`'dır.
- Dolayısıyla bir dizi nesnesi oluşturmak ile o dizinin odalarına değer atamak farklı şeylerdir.
  - Sağlıklı bir dizi yapısı için dizinin odaları varsayılan değerde bırakılmamalı ve ilk değer atanmalıdır.

Diziler Nesnedir

- Java’da diziler, nesnedirler.
- Diziyi gösteren değişken de aslında, dizi nesnesini gösteren bir referanstır.
  - Dizinin referansına `null` atanabilir.
- Diziler üzerinde uzunluğunu yani dizinin oda sayısını veren `int` tipinde bir alan bulunur:
  - `length`

Soru ve Cevap Zamanı!

selsoft
build better, deliver faster

info@selsoft.com.tr
selsoft.com.tr

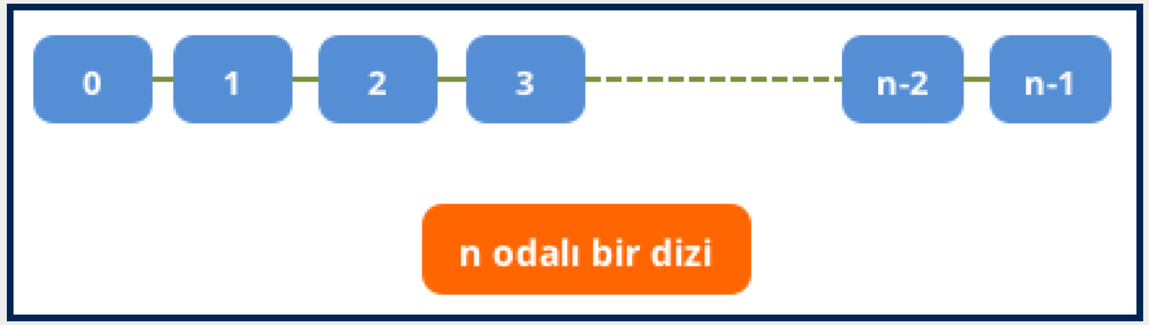
selsoft

build better, deliver faster

Elemanlara Erişim

Dizi Elemanlarına Erişim

- Dizi elemanlarına `[]` içinde dizinin oda numarası, 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 is 0
Pizza pizza = pizzas[2]; // pizza is null
pizzas[0] = new Pizza(); // Assigning a new pizza
```

Oda Numaraları - I

- Dizinin elemanlarına erişirken oda numarası olarak `int` tipinde değişken ya da bir sabite kullanılmalıdır.
  - Oda erişiminde `long` kullanılamaz ama `int`'e otomatik olarak çevrilen `byte`, `short` ve `char` kullanılabilir.
- Dizi oluşturulurken boyut olarak negatif sayı verilemez.
  - Bu durumda `java.lang.NegativeArraySizeException` fırlatılır.
- Dolayısıyla en çok `2^31-1` odalı bir dizi oluşturulabilir.

Oda Numaraları - II

- Odalara erişimde `0`'dan küçük veya **`length-1`**'den büyük bir indis kullanılması halinde **`java.lang.ArrayIndexOutOfBoundsException`** fırlatılır.
- Dolayısıyla dizinin elemanlarına erişimde muhakkak bu sınırların arasında olunduğunun kontrolü yapılmalıdır.

Dizi Elemanlarına İlk Değer Atama - I

- Dizideki odalara ilk değer atamanın ilk yolu, odalara tek tek ulaşarak atama yapmaktır.
  - Bu durumda **`for`** ya da **`while`** döngüsü kullanılır.
- Oda numarasıyla odalara ulaşılırken sınır değerler olan `0` ve **`length-1`**'e dikkat edilmesi gereklidir.

```
Random r = new Random();
for (int i = 0; i < intArray.length; i++) {
  int randomInt = r.nextInt();  //One of 2^32 ints
  int sayi = randomInt % 100;
  intArray[i] = sayi;
}
```

Dizi Elemanlarına İlk Değer Atama - II

- Dizi oluşturulurken içinde saklanacak veriler biliniyorsa, "{ }" içinde virgül ile ayrılarak verilebilir.
  - Böyle ilk değer vermede **new** kullanılmaz.

```
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}
```

Dizi Elemanlarına İlk Değer Atama - III

- Bu şekilde ilk değer vermede tanıtma ve ilk değer atama bir arada olmalıdır aksi taktirde derleme hatası olur.

```
boolean[] b;
b = {true, false} // Hata!
```

ArrayDemo

- `org.javaturk.oopj.ch06.ArrayDemo`

Soru ve Cevap Zamanı!

selsoft
build better, deliver faster

info@selsoft.com.tr
selsoft.com.tr

selsoft

build better, deliver faster

for each

For Each - I

- `for`'un **her biri** ya da **for each** anlamında bir kullanımı vardır.
- Dizi gibi torbalar üzerinde çalışıp, bir indis kullanmadan, elemanlara **tek tek ulaşmak** (**iteration**) için özel bir `for` yapısıdır.

For Each - II

- `for`'un **her biri** (**for each**) anlamında bir kullanımı vardır.
- Dizi gibi torbalar üzerinde çalışıp, bir indis kullanmadan, elemanlara **tek tek ulaşmak** (**iteration**) için özel bir `for` yapısıdır.

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

```
int[] array;
… // Initialization, etc.
for(int i : array)
   System.out.println(i);
```

```
int[] array;
… // Initialization, etc.
for(int i = 0; i < array.length; i++)
   System.out.println(array[i]);
```

For Each - III

- Bu **for each** anlamındaki `for` yapısı sadece dizideki elemanlara erişim için kullanılır, onlara atama yapmaz.

- Dolayısıyla **for each** sadece **okuma** amaçlıdır (**read only**), derleme ya da çalışma zamanında hata vermese bile aşağıdaki kodun bir etkisi yoktur:

```
int[] intArray = new int[10];

for(int i:intArray)
    i = 5;    // No effect!
```

ForEach

- `org.javaturk.oopj.ch06.ForEach`

Soru ve Cevap Zamanı!

selsoft
build better, deliver faster

info@selsoft.com.tr
selsoft.com.tr

selsoft

build better, deliver faster

Parametre Olarak Dizi

Metotlara Parametre Olarak Geçme

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

```
void calculateSum(int[] array){…}

int[] array1 = new int[3];
…
calculateSum(array1);

int[] array2 = {81, 19, -14};
calculateSum(array2);

calculateSum(new int[]{43, 25, 99}); // OK
calculateSum({43, 25, 99});    //Error! Type info missing!
calculateSum(new int[3]);      //Passing with default values!
```