JAVA ARRAY

JAVA ARRAY

Array Nedir?

Array, aynı tipten çok sayıda değişken tanımlamak için kullanılır. Soyut bir veri yapısıdır. Matematikteki sonlu dizi kavramına benzer. Java dilinde array bir sınıftır. Her sınıfın soyut bir veri tipi olduğunu biliyoruz. Array sınıfı array yaratma, arraylerle işlem yapma, array içinde bileşen arama ve array'in bileşenlerini sıralama gibi array ile ilgili işlemleri yapmaya yarayan öğeleri içeren bir sınıftır.

JAVA ARRAY

Array Yaratma

Array bir sınıf olduğuna göre, bir array'i yaratma bir sınıftan nesne yaratma gibidir. Üç aşaması vardır:

Birinci Aşama : Array sınıfının bildirimi

İkinci Aşama : Array sınıfından array nesnesini yaratma

Üçüncü Aşama : Array'in bileşenlerine değer atama

Şimdi bu üç aşamanın nasıl yapıldığını bir örnek üzerinde görelim.

JAVA ARRAY

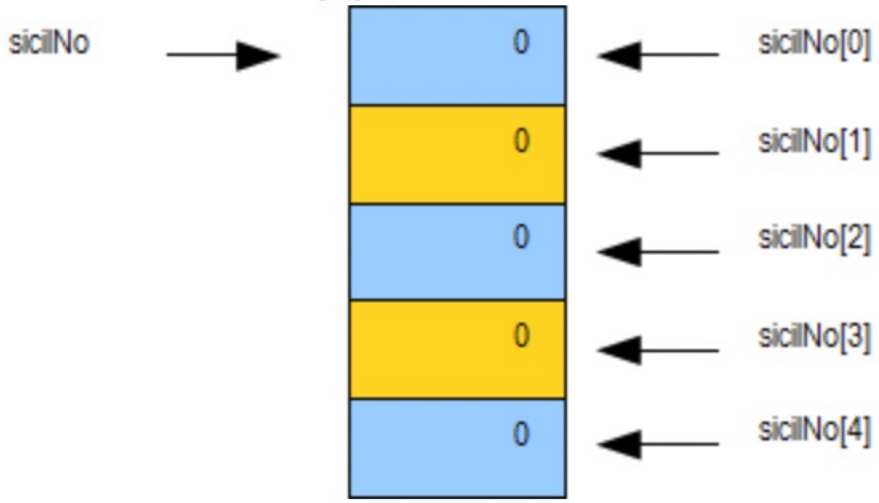
Birinci aşama:

```
int [] sicilNo ;
```

deyimi int tipinden `sicilNo` adlı bir array bildirimidir. Bu aşamada `sicilNo` null işaret eden bir referanstır (işaretçi, pointer).

İkinci aşama:

```
sicilNo = new int[5] ;
```



JAVA ARRAY

Constructor `sicilNo` tarafından işaret edilen bir nesne yaratır. Başka bir deyişle, ana bellekte (heap içinde) 5 tane `int` değer tutacak bir yer ayırır. `sicilNo` bellekte ayrılan bu yeri işaret eder. O nedenle, `sicilNo` 'ya referans (işaretçi, pointer) denmektedir. O halde, array adları referans tipidir. Başka bir deyişle, array'in adı bir değer değil, kendisine ait nesnenin bellekte bulunduğu adresi işaret eden referanstır.

Yukarıdaki örnekte, arrayin işaret ettiği nesnenin içinde
`int` tipi veri tutacak 5 tane bellek adresi vardır. Bu adresler

```
SicilNo[0]
sicilNo[1]
sicilNo[2]
sicilNo[3]
sicilNo[4]
```

adları tarafından işaret (referans) edilirler.

JAVA ARRAY

[ ] Operatörü

Array adının sonuna gelen `[ ]` parantezleri, arrayin bileşenlerini, yukarıda gösterildiği gibi, damgalarıyla (indis, index) belirler.

Array'in Bileşenleri

`sicilNo[i] (i=0,1,2,3,4)` ler söz konusu nesne içinde birer değişkendir. Bu değişkenler sayesinde, array beş tane `int` tipi veriyi bir arada tutabilme yeteneğine sahip olur. Bu değişkenlere array'in bileşenleri ya da öğeleri denir. `0,1,2,3,4` sayıları bileşenlerin sıra numaralarıdır; damga (index) adını alırlar. Sıra numaraları (index) daima 0 dan başlar, birer artarak gider. n tane bileşeni olan bir array'in ilk bileşeninin damgası 0, son bileşeninin damgası (n-1) olur. Bir array'in uzunluğu onun bileşenlerinin sayısıdır.

Eğer `new int[5]` yerine new `int[500]` yazsaydık, 5 bileşen yerine 500 bileşen elde ederdik.

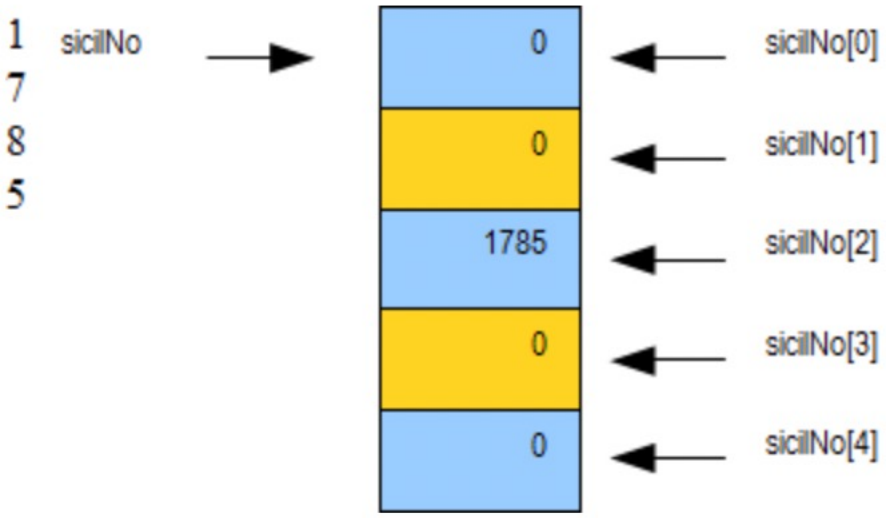
Arrayin işaret ettiği nesne yaratılınca, onun bileşenleri kendiliğinden başlangıç değeri alırlar. Bunlara `öndeğer` (default value) denir. Öndeğerler bileşenlerin veri tipine bağlı olarak değişir. Java dilinde bütün sayısal değişkenlerin öndeğerleri daima `0` dır. Boolean tipin öndeğeri `false` olur. Referans tiplerde ise `null` olur. O nedenle, yukarıda `sicilNo` referansının işaret ettiği nesne içindeki `SicilNo[0], sicilNo[1], sicilNo[2], sicilNo[3] , sicilNo[4]` bileşenlerinin (değişken) öndeğerleri kendiliğinden `0` olur.

JAVA ARRAY

Üçüncü aşama:

```
sicilNo[2] = 1785;
```

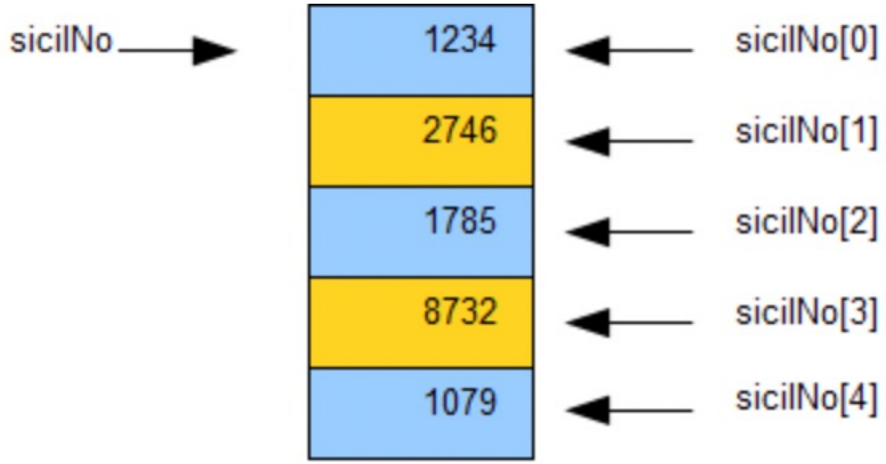
ataması, array'in üçüncü bileşenine 1785 değerini atar.



İstenirse,

```
int [] sicilNo = new int[5] ;
```

deyimi ile ilk iki aşama tamamlanır, bileşenlere değer atama işi sonraya bırakılabilir.



İstenirse öteki bileşenlere de benzer yolla atamalar yapılabilir.

Yukarıdaki üç aşamayı birleştirerek iki ya da bir adımda hepsini bitirebiliriz. Örneğin,

```
int [] sicilNo = new int[] {1234, 2746, 1785, 8732, 1079};
```

deyimi üç aşamayı birden yapar.

JAVA ARRAY

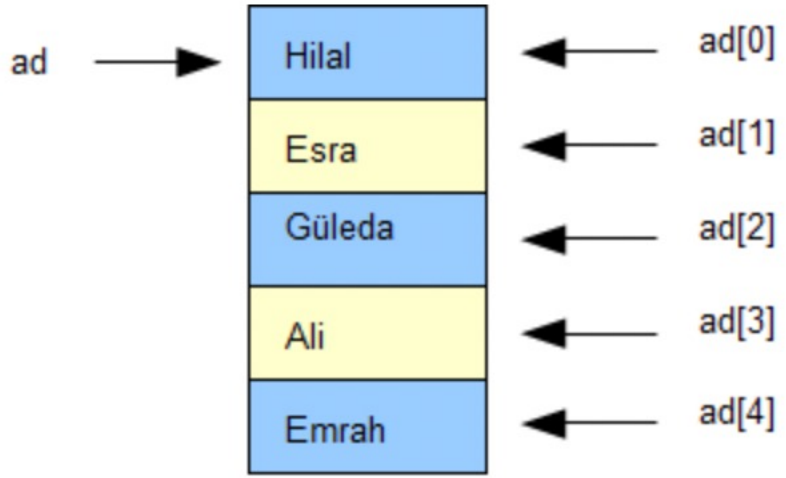
Java dilinde her veri tipinden array yaratılabilir. Örneğin,

```
String [] ad = new String [] {“Hilal”, “Esra”, “Güleda”, “Ali”, “Emrah”};
```

deyimi `String` tipinden bir array sınıfı bildirmiş, onun bir nesnesini yaratmış ve o nesnenin bileşenlerine `String` tipi değerler atamıştır. Bu atama

```
ad[0] = “Hilal” ; ad[1] = “Esra” ; ad[2] = “Güleda” ; ad[3] = “Ali” ; ad[4] = “Emrah” ;
```

atamalarına denktir.



JAVA ARRAY

Artık, array yaratmak için genel sözdizimini yazabiliriz:

```
veriTipi [] arrayAdı ; (array bildirimi)
arrayAdı = new veriTipi [bileşen sayısı]; (array nesnesini yaratma)
```

Array nesnesi yaratmak demek, ana bellekte arryin bileşenlerine birer yer (bellek adresi) ayırmak demektir. Array nesnesi yaratıldıktan sonra onun bileşenlerine değer atama eylemi, diğer değişkenlere değer atama gibidir. İstenen bileşen indeks sayısı ile seçilerek seçkili (random) değer atanabilir. Örneğin, yukarıda yapıldığı gibi,

```
ad[2] = "Güleda"
```

deyimi ad adlı arrayin 3-üncü bileşenine "Güleda" değerini atamaktadır.

JAVA ARRAY

Array bildiriminde, yukarıda yaptığımız gibi, arrayin uzunluğunu (bileşenlerinin sayısını) nesneyi yaratırken belirtmeliyiz. Ancak, bazı durumlarda, nesneyi yaratırken arrayin uzunluğunu tam bilmiyor olabiliriz. Bu durumda, Array sınıfı yerine `ArrayList, LinkedList ya da Vector` sınıflarından birisini kullanmalıyız. Tabii, bu sınıfların nitelikleri birbirlerinden farklıdır. Yapmak istediğimiz işe hangisi uyuyorsa onu seçmeliyiz.

Örneğin,

```
short [] sıraNo = new short[];
```

bildirimi java dilinde geçerli değildir. Derleyici bu bildirime

```
array dimension missing
```

hata iletisini yollayacaktır.

Onun yerine

```
short [] sıraNo = new short[28];
```

gibi, bileşen sayısını belirten deyim yazılmalı ya da { } bloku içinde bileşenlere değerleri atanmalıdır. Tabii, bu şekilde bileşenlere değer atanınca, arrayin uzunluğu (bileşen sayısı) kendiliğinden belli olmaktadır.

JAVA ARRAY

Arrayin Bileşenlerine Değer Atama Yöntemleri

Aşağıdaki üç yöntem aynı işi yapar.

1.Yöntem. Arrayin uzunluğunu belirleyip seçkili (random) değer atama

```
int[] arr;
arr = new int[10];
arr[6]=70; arr[1]=20; arr[2]=30; arr[7]=80; arr[4]=50;
arr[3]=40; arr[0]=10; arr[8]=90; arr[9]=100; arr[5]=60;
```

deyimi 10 bileşenli bir array yaratır sonra bileşenlerine seçkili (random) değer atar.

2.Yöntem. Nesneyi istemli yaratıp sıralı değerler atama

```
int[] arr;
arr = new int[] { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 };
```

deyimi 10 bileşenli bir array yaratır ve bileşenlerine sıralı değerler atar. New int[] ifadesi arrayin uzunluğunu belirlemez; ancak {} bloku içine sırayla yazılan değerler array uzunluğunu ve her bileşene atanan değeri kesinkes belirler. {} içindeki değerlerin yazılış sırası ile bileşenlerin sırası uyumludur. Örneğin, {} içindeki 7-inci değer 7-inci bileşene aittir. Tabii, 7-inci bileşenin damgasının 6 olduğunu biliyoruz; çünkü damgalama işlemi 1 den değil 0 dan başlar. O nedenle `arr[6] = 7` dir. Arraylerde bu özeliği daima anımsamalıyız.

3.Yöntem. Nesneyi istemsiz yaratıp sıralı değerler atama

```
int[] arr { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 };
```

deyimi yukarıdaki deyimin yaptığı işin aynısını yapar; ancak new operatörünü istemli (explicit) kullanmaz, nesne istemsiz (implicit) olarak yaratılır.

JAVA ARRAY

Arrayin Uzunluğunu Bulma

Java dilinde bir arrayin uzunluğunu bulmak için length özgenini (attribute) kullanırız. Çoğumuz array uzunluğunu bulan bir metot (fonksiyon) olması gerektiğini sanabiliriz. Ama length bir metot değil, bir özgendir ve arrayin uzunluğunu belirlememize yarar.

```
public class ArrayUzunluğu
{
public static void main(String[] args)
{
String[] meyve = {"elma", "çilek", "kiraz"};
int arrayUzunluğu = meyve.length;
System.out.format("Bileşen sayısı: %d ", arrayUzunluğu);
}
}
```

Çıktı:

```
Bileşen sayısı: 3
```

JAVA ARRAY

Arrayin Bileşenlerine Erişim

Arrayin bileşenleri değişken olduklarından, onlara istendiğinde değer atanabileceği, istenirse atanan değerlerin değiştirilebileceği açıktır. Yukarıdaki `sicilNo[2] = 1785` atama deyimi, arraylerin üstün bir niteliğini ortaya koyar. Arrayin istenen bileşenine damga (indeks) sayısı ile doğrudan (seçkili, random) erişmek mümkündür. Her bileşen bir değişken olduğu için, o bileşen tipiyle yapılabilen her işlem onlar için de yapılabilir, değişkenlerle ilgili kurallar bileşenler için de aynen geçerlidir. `SicilNo[2] = 1785;` deyimi indeksi 2 olan bileşene 1785 değerini atamıştır.

Bileşen değerleri aynı veri tinden başka değişkenlere aktarılabilir ve tabii bu işin tersi de yapılabilir. Örneğin, `x` aynı tipten bir değişken ise

```
x = sicilNo[2];
```

ataması, `sicilNo[2]` bileşeninin değerini `x` değişkenine aktarır; öyleyse, bu atama deyimi

```
x = 1785 ;
```

atamasına denktir.

Tersine olarak, y aynı tipten bir değişken ise değeri bileşene ktarılabilir. Örneğin,

```
int y = 123;
sicilNo[2] = y;
```

ataması geçerlidir. Bu atama y nin değerini sicilNo[2] bileşenine aktarır; dolayısıyla,

```
sicilNo[2] = 123;
```

atamasına demktir.

JAVA ARRAY

İstemli (explicit) dönüşüm yaptırırken, programcı ne yaptığının farkında olmalıdır. Aksi halde mantıksal hata dediğimiz öldürücü hatalar oluşur. Bu hataların öldürücü sayılmasının nedeni açıktır.

Programda sözdizimi (syntax) hatası olmadığı için derlenecek ve yorumlanacaktır. Yürüyen bir programın hata yaptığını anlamak zordur. Oysa sözdizimi hatası içeren programlar masumdur; onlar asla çalışmayacağı için, programcısından başkasına zarar veremezler.