

MTK467 Nesneye Yönelik Programlama

Hafta7- Diziler

Zümra Kavafoğlu
<https://zumrakavafoglu.github.io/>

Dizilerin metod parametresi olarak kullanılması

Bir yönteme parametre olarak verilen dizi, metodun tanımlanması sırasında, yine metod isminden sonra parantez içine tipi belirtilerek ve dizi parantezleri kullanılarak yazılır. Ancak parantezlerin içine dizinin boyu yazılmaz.

- Yöntem çağırılırken de dizi ismi parantezsiz olarak yazılır.

Dizilerin metod parametresi olarak kullanılması

```
public class PrintArrayWithMethod {  
    public static void printArray(int[] arr){  
        int n = arr.length;  
        for(int i=0; i<n; i++)  
            System.out.print(arr[i] + " ");  
    }  
  
    public static void main(String[] args){  
        int[] oddNumbers = {1,3,5,7,9};  
        printArray(oddNumbers);  
    }  
}
```

çıktı

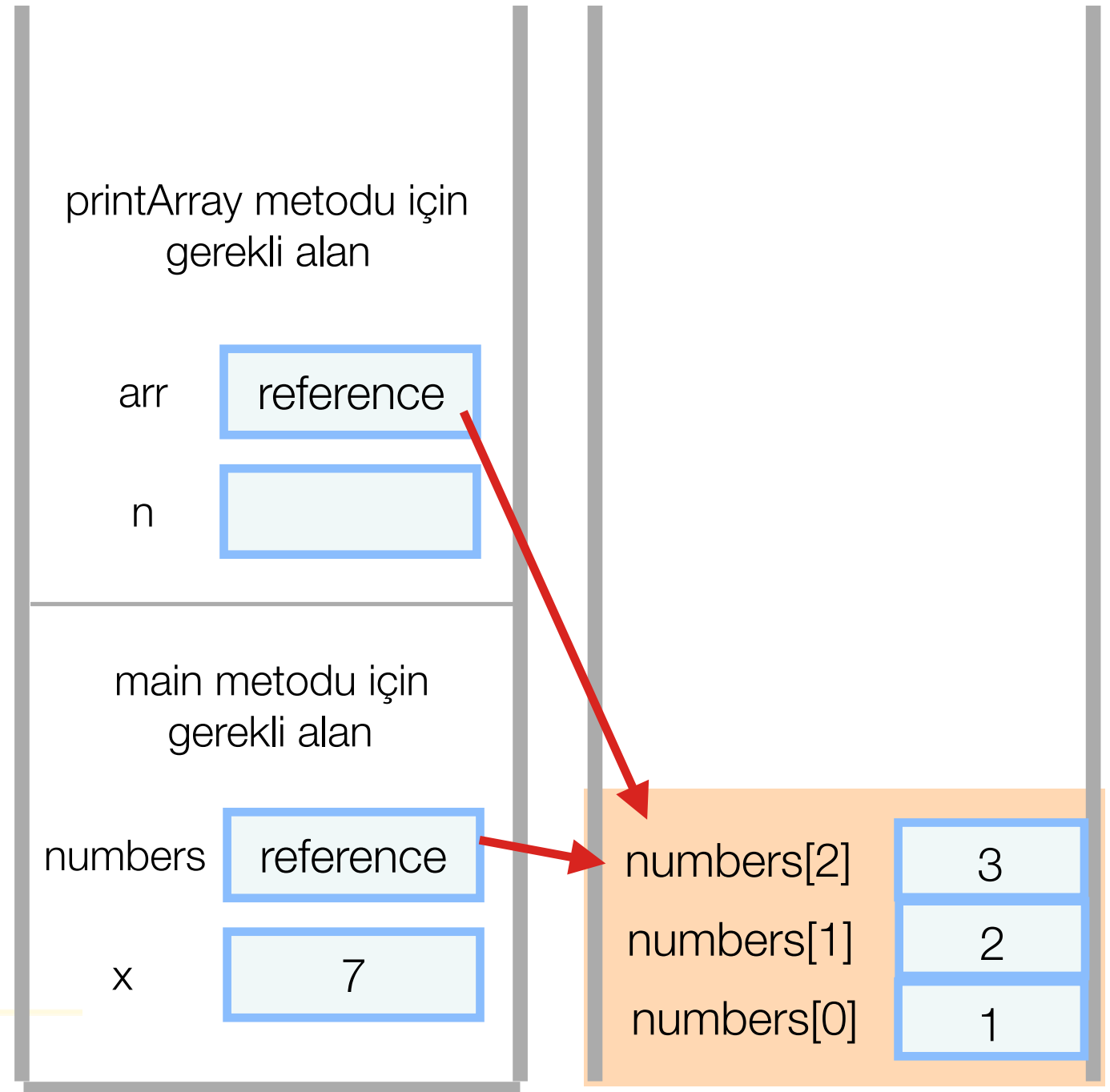
1 3 5 7 9

Dizilerin metod parametresi olarak kullanılması : Pass by reference

Diziler metotlara parametre olarak verilirken pass by reference yapılır. pass by value'nun tersine metotlar dizinin bir kopyası değil direkt kendisi üzerinde işlem yapar. Dolayısıyla parametre olarak verilen diziye metot içinde yapılan değişiklikler, dışarıya da yansır.

Dizilerin metod parametresi olarak kullanılması : Pass by reference

```
public class PassByReferenceDemo {  
  
    public static void printArray(int[] arr){  
  
        int n = arr.length;  
  
        for(int i=0; i<n; i++){  
            System.out.print(arr[i] + " ");  
        }  
  
        public static void main(String[] args){  
  
            int x = 7;  
  
            int[] numbers = {1,2,3};  
  
            System.out.println("x: " + x);  
            System.out.print("Numbers: ");  
            printArray(numbers);  
        }  
    }  
}
```



stack

Dizilerin metod parametresi olarak kullanılması : Pass by reference

```
public static void changeValues(int[] arr, int x ){  
    x += 10;  
  
    int n = arr.length;  
  
    for(int i=0; i<n; i++){  
        arr[i] += 10;  
    }  
}
```

Dizilerin metod parametresi olarak kullanılması : Pass by reference

```
public static void main(String[] args){  
    int x = 7;  
  
    int[] numberArray = {1, 2, 3};  
  
    System.out.println("Before calling changeValues method: ");  
  
    System.out.println("x: "+x);  
  
    System.out.print("Number Array: ");  
    for(int i=0; i<numberArray.length; i++){  
        System.out.print(numberArray[i]+" ");  
    }  
  
    changeValues(numberArray,x);  
  
    System.out.println("\n\nAfter calling changeValues method: ");  
  
    System.out.println("x: "+x);  
  
    System.out.print("Number Array: ");  
    for(int i=0; i<numberArray.length; i++){  
        System.out.print(numberArray[i]+" ");  
    }  
}
```

Dizilerin metod parametresi olarak kullanılması : Pass by reference

changeValues metodu çağırıldıktan sonra primitif tipli argümanın değeri değişmezken, dizi argümanın değeri değişir.

Before calling changeValues method:

x: 7

Number Array: 1 2 3

After calling changeValues method:

x: 7

Number Array: 11 12 13

Primitif tiplerden farklı olarak diziler heap adı verilen farklı bir hafıza bölümünde tutulur. Stack'de ise dizinin heapdeki adresini belirten bir referans tutulur.

Dizilerin metod parametresi olarak kullanılması : Pass by reference

changeValues metodu numberArray dizisiyle çağırıldığında, numberArray dizisinin referans değeri metod parametresi arr'ye aktarılır. Dolayısıyla hem numberArray hem de arr heap'te tutulan aynı değerlere işaret eder. Bu durumda arr'nin elemanlarında değişiklik yapmakla numberArray'in elemanlarında değişiklik yapmak aynıdır.

Dizilerin metod parametresi olarak kullanılması : Pass by reference

```
public class PassByReferenceDemo2 {  
    public static void changeValues(int[] arr, int x ){  
        x += 10;  
        int n = arr.length;  
        for(int i=0; i<n; i++){  
            arr[i] += 10;  
        }  
    }  
  
    public static void main(String[] args){  
        int x = 7;  
        int[] numberArray = {1, 2, 3};  
        changeValues(numberArray,x);  
    }  
}
```

main metodu için
gerekli alan

x

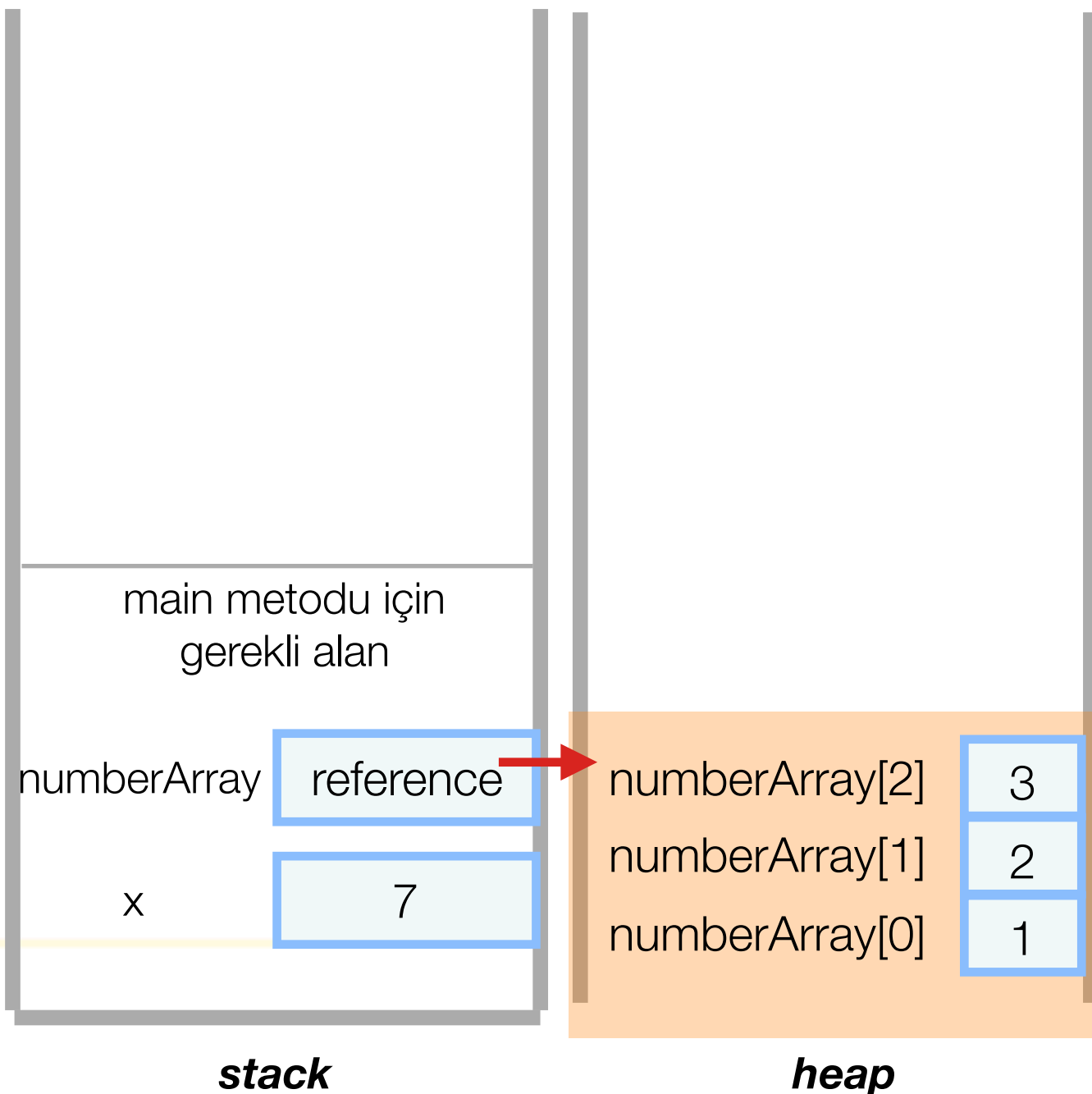
7

stack

heap

Dizilerin metod parametresi olarak kullanılması : Pass by reference

```
public class PassByReferenceDemo2 {  
    public static void changeValues(int[] arr, int x ){  
        x += 10;  
        int n = arr.length;  
        for(int i=0; i<n; i++){  
            arr[i] += 10;  
        }  
    }  
  
    public static void main(String[] args){  
        int x = 7;  
        int[] numberArray = {1, 2, 3};  
        changeValues(numberArray,x);  
    }  
}
```



Dizilerin metod parametresi olarak kullanılması : Pass by reference

```
public class PassByReferenceDemo2 {
```

```
    public static void changeValues(int[] arr, int x ){
```

```
        x += 10;
```

```
        int n = arr.length;
```

```
        for(int i=0; i<n; i++){
            arr[i] += 10;
        }
```

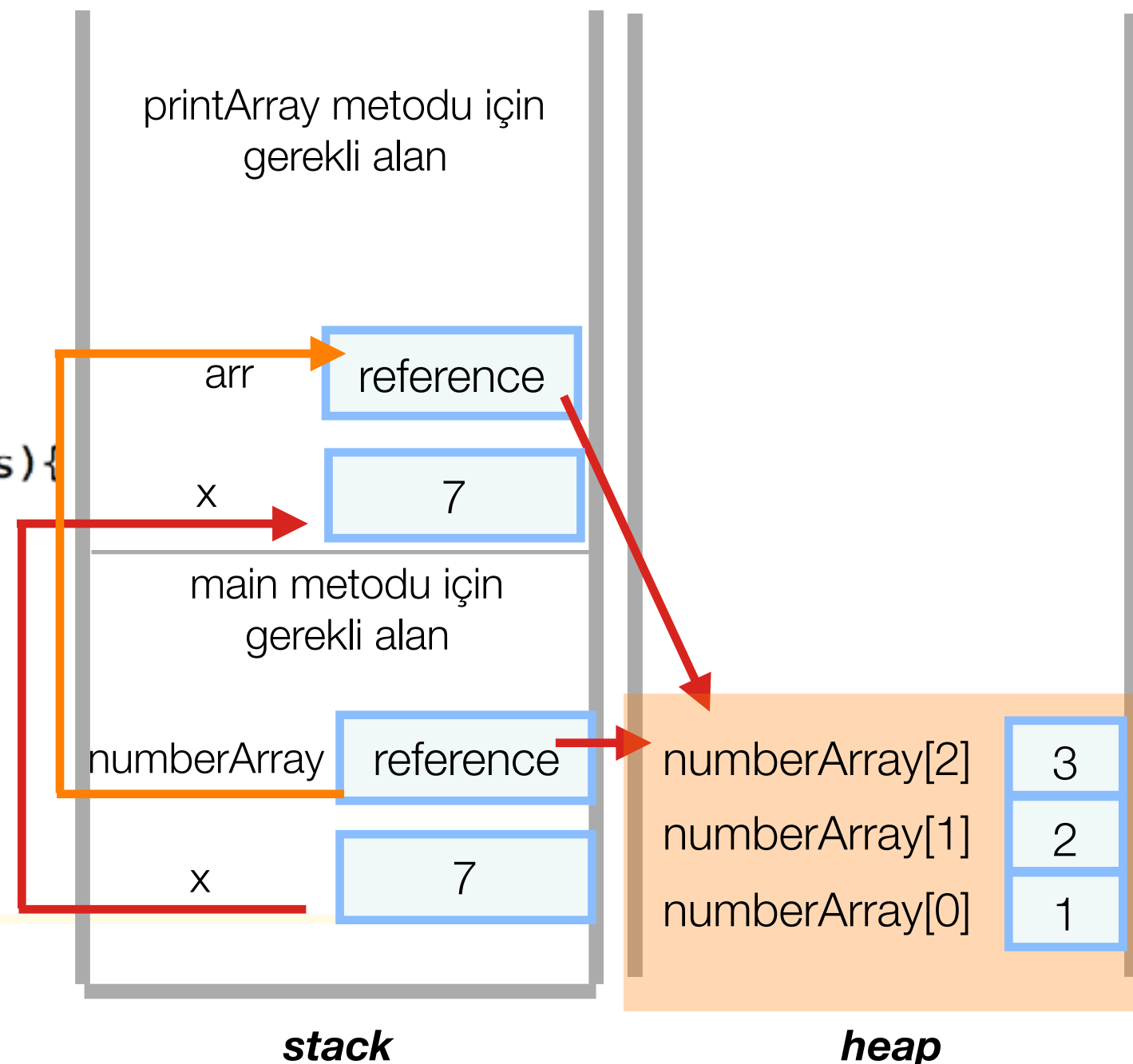
```
    public static void main(String[] args){
```

```
        int x = 7;
```

```
        int[] numberArray = {1, 2, 3};
```

```
        changeValues(numberArray, x);
```

```
    }
```



Dizilerin metod parametresi olarak kullanılması : Pass by reference

```
public class PassByReferenceDemo2 {
```

```
    public static void changeValues(int[] arr, int x ){
```

```
        x += 10;
```

```
        int n = arr.length;
```

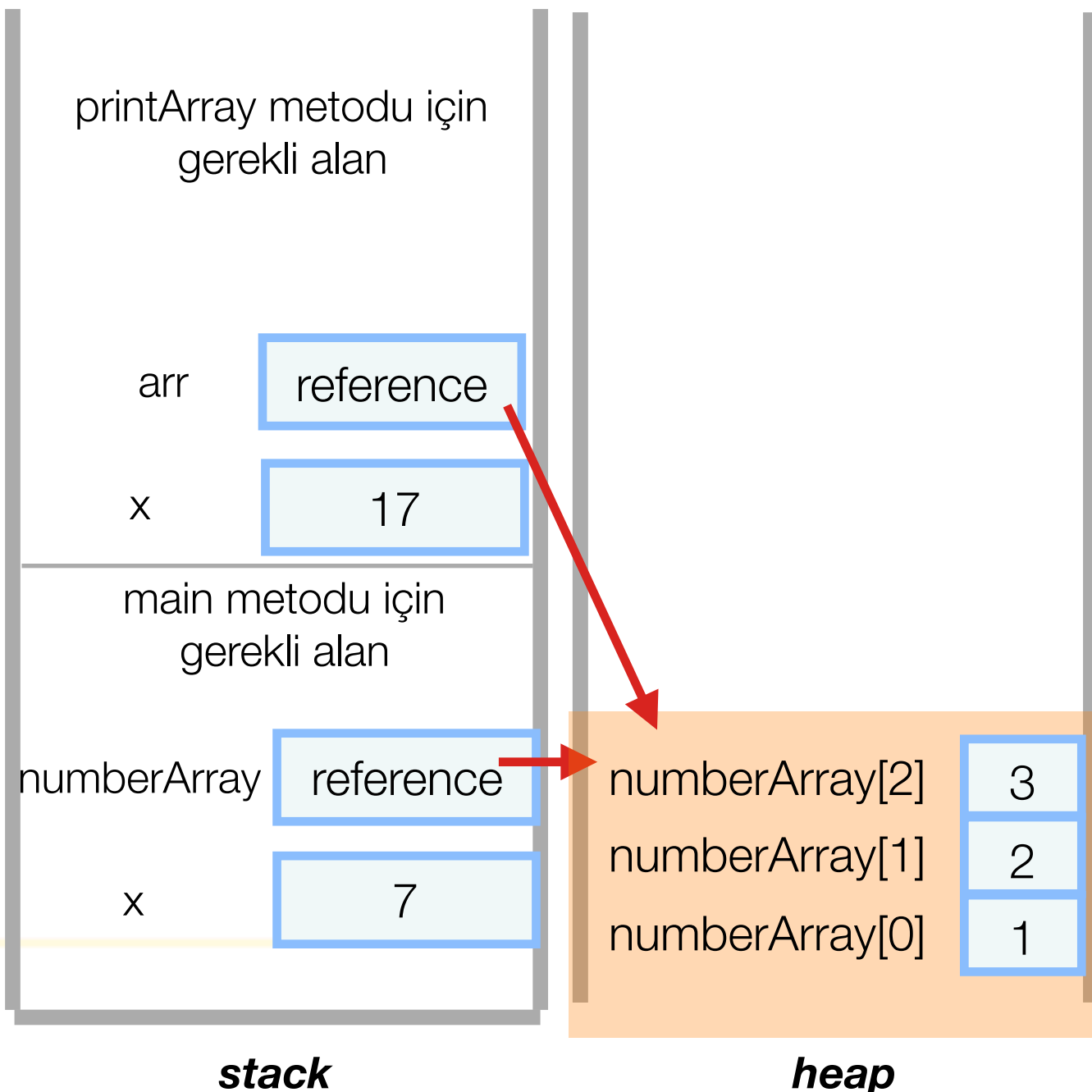
```
        for(int i=0; i<n; i++){
            arr[i] += 10;
        }
    }
```

```
    public static void main(String[] args){
```

```
        int x = 7;
```

```
        int[] numberArray = {1, 2, 3};
```

```
        changeValues(numberArray, x);
    }
```



Dizilerin metod parametresi olarak kullanılması : Pass by reference

```
public class PassByReferenceDemo2 {
```

```
    public static void changeValues(int[] arr, int x ){
```

```
        x += 10;
```

```
        int n = arr.length;
```

```
        for(int i=0; i<n; i++){
            arr[i] += 10;
        }
```

```
    public static void main(String[] args){
```

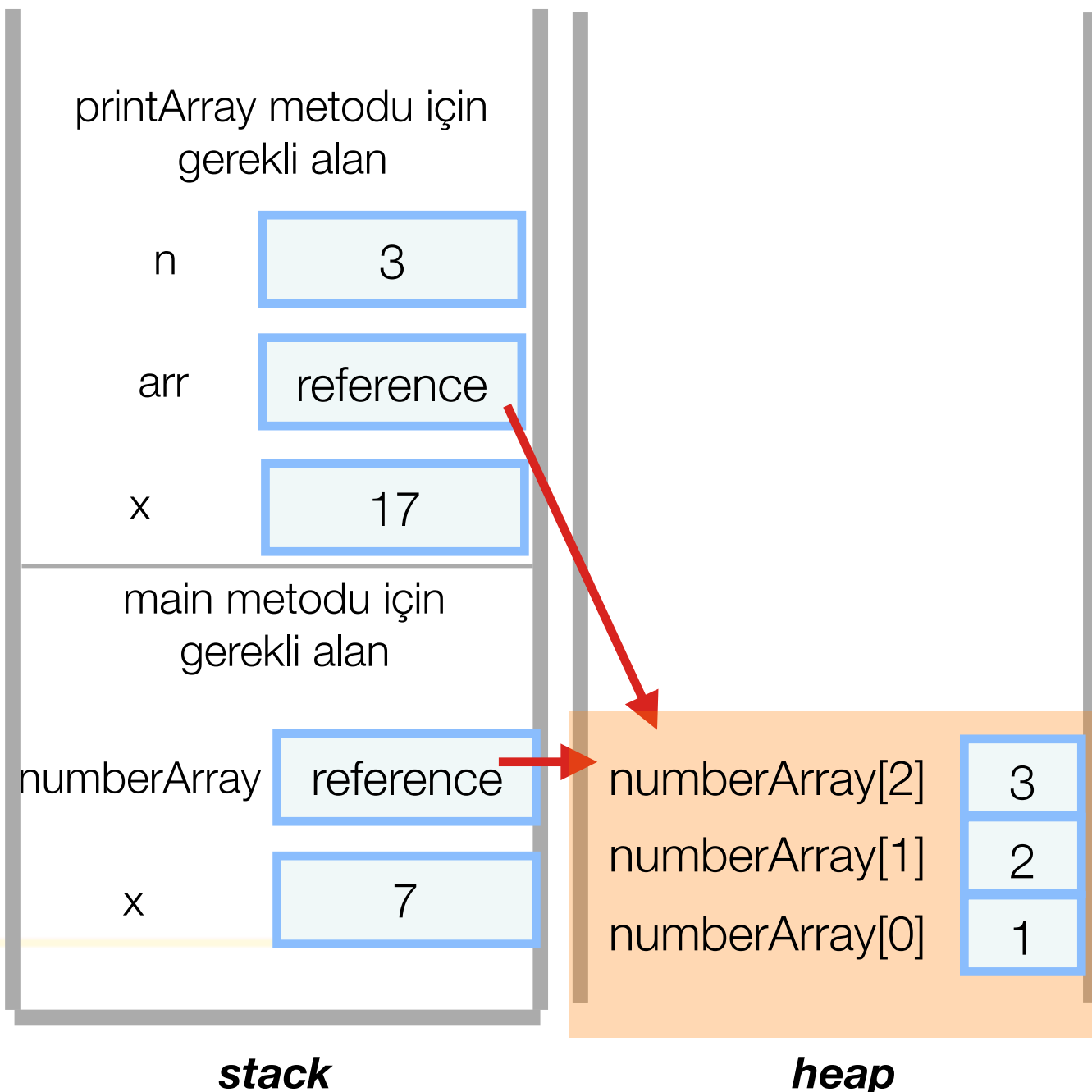
```
        int x = 7;
```

```
        int[] numberArray = {1, 2, 3};
```

```
        changeValues(numberArray,x);
```

```
    }
```

```
}
```



Dizilerin metod parametresi olarak kullanılması : Pass by reference

```
public class PassByReferenceDemo2 {
```

```
    public static void changeValues(int[] arr, int x ){
```

```
        x += 10;
```

```
        int n = arr.length;
```

```
        for(int i=0; i<n; i++){
            arr[i] += 10;
        }
```

```
    public static void main(String[] args){
```

```
        int x = 7;
```

```
        int[] numberArray = {1, 2, 3};
```

```
        changeValues(numberArray,x);
```

```
    }
```

printArray metodu için
gerekli alan

i 0

n 3

arr reference

x 17

main metodu için
gerekli alan

numberArray reference

x 7

numberArray[2] 3
numberArray[1] 2
numberArray[0] 1

stack

heap

Dizilerin metod parametresi olarak kullanılması : Pass by reference

```
public class PassByReferenceDemo2 {
```

```
    public static void changeValues(int[] arr, int x ){
```

```
        x += 10;
```

```
        int n = arr.length;
```

```
        for(int i=0; i<n; i++){  
            arr[i] += 10;  
        }
```

```
    public static void main(String[] args){
```

```
        int x = 7;
```

```
        int[] numberArray = {1, 2, 3};
```

```
        changeValues(numberArray,x);
```

```
    }
```

printArray metodu için
gerekli alan

i

0

n

3

arr

reference

x

17

main metodu için
gerekli alan

numberArray

reference

x

7

numberArray[2]

3

numberArray[1]

2

numberArray[0]

11

stack

heap

Dizilerin metod parametresi olarak kullanılması : Pass by reference

```
public class PassByReferenceDemo2 {
```

```
    public static void changeValues(int[] arr, int x ){
```

```
        x += 10;
```

```
        int n = arr.length;
```

```
        for(int i=0; i<n; i++){
            arr[i] += 10;
        }
```

```
    public static void main(String[] args){
```

```
        int x = 7;
```

```
        int[] numberArray = {1, 2, 3};
```

```
        changeValues(numberArray,x);
```

```
    }
```

printArray metodu için
gerekli alan

i

1

n

3

arr

reference

x

17

main metodu için
gerekli alan

numberArray

reference

x

7

numberArray[2]

3

numberArray[1]

2

numberArray[0]

11

stack

heap

Dizilerin metod parametresi olarak kullanılması : Pass by reference

```
public class PassByReferenceDemo2 {
```

```
    public static void changeValues(int[] arr, int x ){
```

```
        x += 10;
```

```
        int n = arr.length;
```

```
        for(int i=0; i<n; i++){  
            arr[i] += 10;  
        }
```

```
    public static void main(String[] args){
```

```
        int x = 7;
```

```
        int[] numberArray = {1, 2, 3};
```

```
        changeValues(numberArray,x);
```

```
    }
```

printArray metodu için
gerekli alan

i

1

n

3

arr

reference

x

17

main metodu için
gerekli alan

numberArray

reference

x

7

numberArray[2]

3

numberArray[1]

12

numberArray[0]

11

stack

heap

Dizilerin metod parametresi olarak kullanılması : Pass by reference

```
public class PassByReferenceDemo2 {
```

```
    public static void changeValues(int[] arr, int x ){
```

```
        x += 10;
```

```
        int n = arr.length;
```

```
        for(int i=0; i<n; i++){
            arr[i] += 10;
        }
```

```
    public static void main(String[] args){
```

```
        int x = 7;
```

```
        int[] numberArray = {1, 2, 3};
```

```
        changeValues(numberArray,x);
```

```
    }
```

printArray metodu için
gerekli alan

i

2

n

3

arr

reference

x

17

main metodu için
gerekli alan

numberArray

reference

x

7

numberArray[2]

3

numberArray[1]

12

numberArray[0]

11

stack

heap

Dizilerin metod parametresi olarak kullanılması : Pass by reference

```
public class PassByReferenceDemo2 {
```

```
    public static void changeValues(int[] arr, int x ){
```

```
        x += 10;
```

```
        int n = arr.length;
```

```
        for(int i=0; i<n; i++){  
            arr[i] += 10;  
        }
```

```
    public static void main(String[] args){
```

```
        int x = 7;
```

```
        int[] numberArray = {1, 2, 3};
```

```
        changeValues(numberArray,x);  
    }
```

printArray metodu için
gerekli alan

i

2

n

3

arr

reference

x

17

main metodu için
gerekli alan

numberArray

reference

x

7

stack

numberArray[2]

13

numberArray[1]

12

numberArray[0]

11

heap

Dizilerin metod parametresi olarak kullanılması : Pass by reference

```
public class PassByReferenceDemo2 {
```

```
    public static void changeValues(int[] arr, int x ){
```

```
        x += 10;
```

```
        int n = arr.length;
```

```
        for(int i=0; i<n; i++){
            arr[i] += 10;
        }
```

```
    public static void main(String[] args){
```

```
        int x = 7;
```

```
        int[] numberArray = {1, 2, 3};
```

```
        changeValues(numberArray,x);
```

```
    }
```

printArray metodu için
gerekli alan

i

3

n

3

arr

reference

x

17

main metodu için
gerekli alan

numberArray

reference

x

7

numberArray[2]

13

numberArray[1]

12

numberArray[0]

11

stack

heap

Dizilerin metod parametresi olarak kullanılması : Pass by reference

```
public class PassByReferenceDemo2 {
```

```
    public static void changeValues(int[] arr, int x ){
```

```
        x += 10;
```

```
        int n = arr.length;
```

```
        for(int i=0; i<n; i++){
            arr[i] += 10;
        }
```

```
    public static void main(String[] args){
```

```
        int x = 7;
```

```
        int[] numberArray = {1, 2, 3};
```

```
        changeValues(numberArray, x);
```

```
    }
```

false : döngüden çık

printArray metodu için
gerekli alan

i

3

n

3

arr

reference

x

17

main metodu için
gerekli alan

numberArray

reference

x

7

numberArray[2]

13

numberArray[1]

12

numberArray[0]

11

stack

heap

Dizilerin metod parametresi olarak kullanılması : Pass by reference

```
public class PassByReferenceDemo2 {
```

```
    public static void changeValues(int[] arr, int x ){
```

```
        x += 10;
```

```
        int n = arr.length;
```

```
        for(int i=0; i<n; i++){  
            arr[i] += 10;  
        }
```

```
    }
```

```
    public static void main(String[] args){
```

```
        int x = 7;
```

```
        int[] numberArray = {1, 2, 3};
```

```
        changeValues(numberArray,x);
```

```
    }
```

**changeValues metodundan
çıkıldıktan sonra arr referansı
hafızadan silindi ama heap'deki
değişmiş dizi değerleri olduğu gibi
kaldı**

main metodu için
gerekli alan

numberArray reference

x

7

numberArray[2] 13

numberArray[1] 12

numberArray[0] 11

stack

heap

Metod dönüş tipi olarak diziler

- Bir dizi bir metodun dönüş tipi olabilir.
- Örneğin aşağıdaki metod, parametre olarak aldığı pozitif tamsayı kadar rasgele sayılar üretir ve bunları bir dizi halinde döndürür.

```
public static double[] generateRandomNumbers(int n){  
    double[] randomNumbers = new double[n];  
  
    for(int i=0; i<n; i++){  
        randomNumbers[i] = Math.random();  
    }  
  
    return randomNumbers;  
}
```


Metod dönüş tipi olarak diziler:

RandomNumbers.java

- generateRandomNumbers yöntemi kullanılarak kullanıcının istediği adette rasgele ondalık sayılar ekrana yazdıran aşağıdaki program yazılabilir.

```
public static void main(String[] args){  
    int n;  
  
    System.out.print("How many numbers do you want to generate?: "  
  
    Scanner input = new Scanner(System.in);  
  
    n = input.nextInt();  
  
    System.out.println("Generated numbers are: ");  
    printArray(generateRandomNumbers(n));  
}
```

Geliştirilmiş for döngüsü (for each)

JDK 1.5 ile birlikte bir diziyi indis değişkeni kullanmadan dolaşmayı sağlayan yeni bir for döngüsü yapısı tanımlandı.

Örneğin bir arr dizisinin tüm elemanlarını yazdırmak istediğimizde

```
for(int i=0; i < arr.length; i++){  
    System.out.println(arr[i]);  
}
```

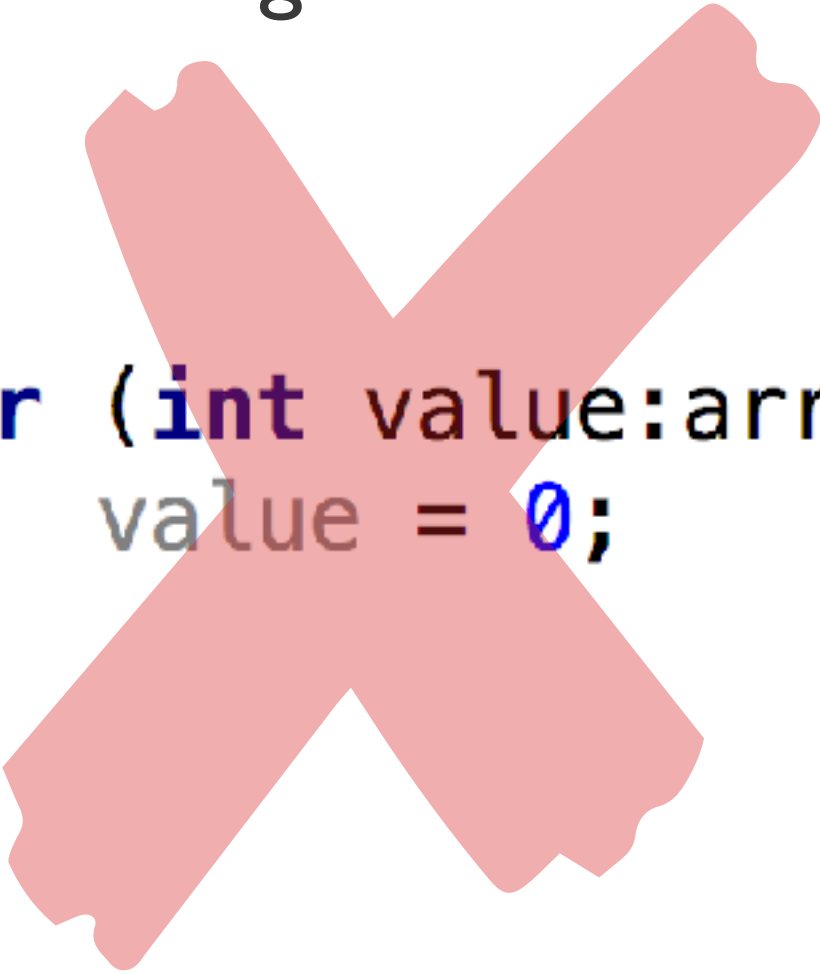
yerine kısaca

```
for (int value:arr) {  
    System.out.println(value);  
}
```

ifadesini kullanabiliriz.

Geliştirilmiş for döngüsü (for each)

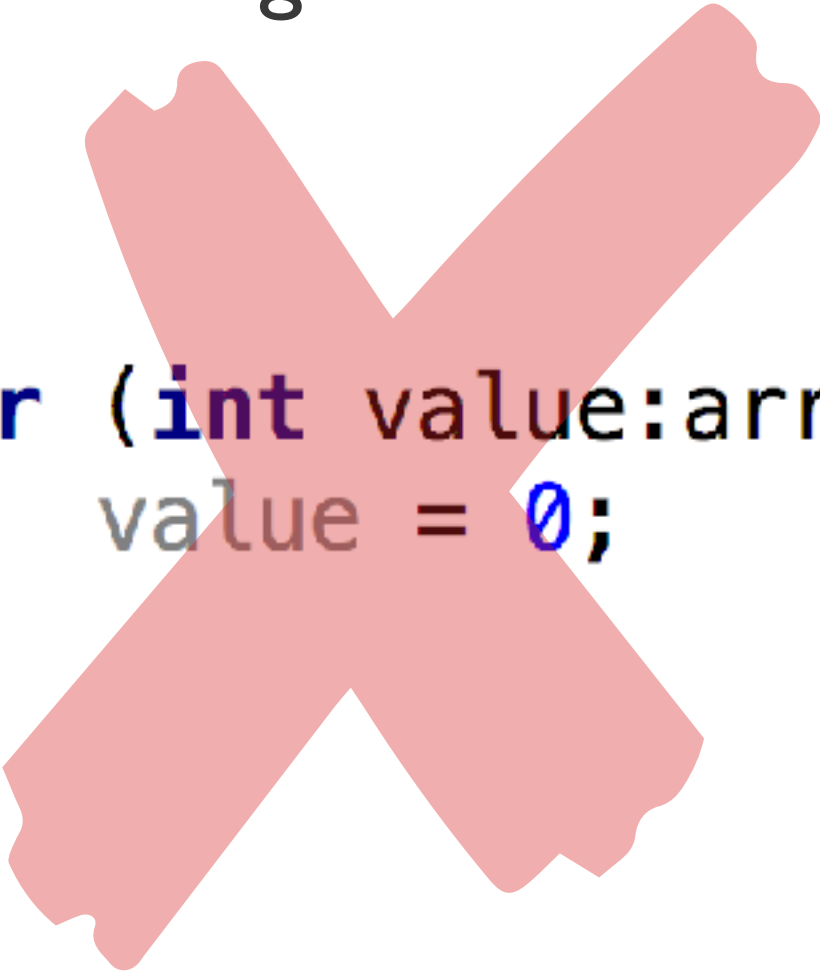
Gelişmiş for döngüsü dizi elemanlarını değiştirmek için kullanılamaz, yalnızca onlara erişmek için kullanılır. Örneğin dizi elemanlarının hepsinin değerini 0 yapmak için aşağıdaki gibi foreach döngüsü kullanmak hatalıdır.



```
for (int value:arr) {  
    value = 0;  
}
```

Geliştirilmiş for döngüsü (for each)

Gelişmiş for döngüsü dizi elemanlarını değiştirmek için kullanılamaz, yalnızca onlara erişmek için kullanılır. Örneğin dizi elemanlarının hepsinin değerini 0 yapmak için aşağıdaki gibi foreach döngüsü kullanmak hatalıdır.



```
for (int value:arr) {  
    value = 0;  
}
```

Dizi elemanlarının yerini değiştirme (Swapping)

```
public static void swapElements(int[] arr1, int index1, int index2){  
    int temp = arr1[index1];  
    arr1[index1] = arr1[index2];  
    arr1[index2] = temp;  
}
```

Dizi elemanlarının yerini değiştirme (Swapping)

SwapElements metodunu kullanarak bir dizinin 1.ve 4. elemanlarının yerini değiştiren program örneği:

```
public static void main(String args[]){  
    int[] intArray = {1, 3, 5, 7, 9};  
  
    System.out.println("Array before swapping");  
    printArray(intArray);  
  
    swapElements(intArray, 0, 3);  
  
    System.out.println("Array after swapping");  
    printArray(intArray);  
}
```

Çıktı

```
Array before swapping  
1 3 5 7 9  
Array after swapping  
7 3 5 1 9
```

Örnek 1: ShiftElements.java

Verilen bir dizinin elemanlarının yerlerini 1 adım geriye doğru kaydıran bir Java programı yazınız.

Örnek 2: GuessThePassword.java

Bir kasanın 5 tamsayı değerden oluşan şifresini rasgele karıştırarak ekrana yazdıran ve kullanıcıdan şifreyi tahmin etmesini isteyen bir Java programı yazınız.

Sıralama algoritmaları

Çoğu bilimsel çalışmada bir dizi nesnenin artan ya da azalan şekilde sıralanabilmesi büyük önem taşır. Biz bu sıralamayı kendi mantığımızla çok basit bir şekilde yapabiliriz, ancak fazla sayıda veriyi sıralamak biraz zamanımızı alabilir. Java programlama diliyle sıralama yapmak için bazı yöntemler vardır.

Selection Sort(Seme Sıralaması)

Başlangı adımı $i=0$ olarak ata. Dizinin uzunluęu n olsun.

1. adım: dizideki i . elemandan sonraki, i . elemandan küçük en küçük elemanın yerini bul.
2. Böyle bir en küçük eleman varsa bu elemanla i . elemanın yerini deęiştir.
3. adım : $i < n-1$ ise i 'yi bir arttır ve 1. adıma git, deęilse sıralamayı sonlandır.

SELECTION SORT $O(n^2)$

5 1 12 -5 16 2 12 14

5 1 12 -5 16 2 12 14

-5 1 12 5 16 2 12 14

-5 1 12 5 16 2 12 14

-5 1 2 5 16 12 12 14

-5 1 2 5 16 12 12 14

-5 1 2 5 12 16 12 14

-5 1 2 5 12 12 16 14

-5 1 2 5 12 12 14 16

Selection Sort(Seme Sıralaması)

```
public static void selectionSort(int[] arr) {  
    int i, j, minIndex;  
    int n = arr.length;  
    for (i = 0; i < n - 1; i++) {  
        minIndex = i;  
        for (j = i + 1; j < n; j++)  
            if (arr[j] < arr[minIndex])  
                minIndex = j;  
        if (minIndex != i) {  
            swapElements(arr,minIndex,i);  
        }  
    }  
}
```

Bubble Sort (Kabarcık Sıralaması)

$j = 0$ ve dizinin uzunluğu n olsun .

1. adım: $i = 0$ olsun.
2. adım: i . elemanla $(i+1)$ nci elemanı karşılaştır. i . eleman $(i+1)$. elemandan büyükse yerlerini değiştir.
3. adım : i , $n-j$ 'den küçükse i 'yi bir arttır ve 2. adıma git.
3. adım: Eğer j adımı için 2. adımda bir değişiklik yapıldıysa j 'yi 1 arttır ve 1. adıma git. Yapılmadıysa sıralamayı sonlandır.

Bubble Sort (Kabarcık Sıralaması)

BUBBLE
SORT $O(n^2)$

5	1	12	-5	16	unsorted
5	1	12	-5	16	5 > 1, swap
1	5	12	-5	16	5 < 12, ok
1	5	12	-5	16	12 > -5, swap
1	5	-5	12	16	12 < 16, ok
1	5	-5	12	16	1 < 5, ok
1	5	-5	12	16	5 > -5, swap
1	-5	5	12	16	5 < 12, ok
1	-5	5	12	16	1 > -5, swap
-5	1	5	12	16	1 < 5, ok
-5	1	5	12	16	-5 < 1, ok
-5	1	5	12	16	sorted

Bubble Sort (Kabarcık Sıralaması)

```
static void bubbleSort(int[] arr) {  
  
    boolean swapped = true;  
    int j = 0;  
  
    while (swapped) {  
        swapped = false;  
        j++;  
        for (int i = 0; i < arr.length - j; i++) {  
            if (arr[i] > arr[i + 1]) {  
                swapElements(arr, i, i + 1);  
                swapped = true;  
            }  
        }  
    }  
}
```

Arrays sınıfının sort metodları

Sıralama programlamada çok ihtiyaç duyulan bir işlem olduğundan Java overloaded sort metodunu otomatik olarak sağlamaktadır.

```
double[] numbers = {6.0, 4.4, 1.9, 2.9, 3.4, 3.5};  
java.util.Arrays.sort(numbers);
```

```
char[] chars = {'a', 'A', '4', 'F', 'D', 'P'};  
java.util.Arrays.sort(chars);
```


Arama Algoritmaları

Bir dizinin içinde belirli bir elemanın olup olmadığını ve bu elemanın yerini bulmamızı gerektiren birçok uygulamaya ihtiyaç duyulur. Bunun için çeşitli arama algoritmaları geliştirilmiştir.

Arama Algoritmaları: Doğrusal Arama (Linear Search)

Doğrusal arama algoritması aranan elemanı dizinin her elemanı ile karşılaştırır. ($O(n)$)

```
public static int linearSearch(int key, int[] arr){  
    int n = arr.length;  
    for(int i=0; i<n; i++){  
        if(key == arr[i]){  
            return i;  
        }  
    }  
  
    return -1;  
}
```

Arama Algoritmaları: İkili Arama (Binary Search)

Bir diğer yaygın arama algoritması ikili arama algoritmasıdır. İkili aramanın çalışabilmesi için dizinin sıralı olması gerekmektedir. Dizinin azalmayan biçimde sıralı olduğunu varsayalım. İkili arama önce aranan elemanla(anahtar) dizinin tam ortasındaki elemanı karşılaştırır. Şu üç durumu göz önüne alalım:

- Eğer anahtar ortadaki elemandan küçükse, anahtarı yalnızca dizinin ilk yarısında aramalısınız.
- Eğer anahtar ortadaki elemana eşitse arama başarılı olmuştur.
- Eğer anahtar ortadaki elemandan büyükse, anahtarı yalnızca dizinin ikinci yarısında aramalısınız.

Aynı işlemleri dizinin aramaya devam ettiğimiz yarısıyla tekrarlayarak algoritma devam eder.

BINARY SEARCH $O(\log_2 n)$

$\{-1, 5, 6, 18, 19, 25, 46, 78, 102, 114\}$ dizisinde 6'yı bulalım:

Step 1 (middle element is $19 > 6$):	-1	5	6	18	19	25	46	78	102	114
Step 2 (middle element is $5 < 6$):	-1	5	6	18	19	25	46	78	102	114
Step 3 (middle element is $6 == 6$):	-1	5	6	18	19	25	46	78	102	114

$\{-1, 5, 6, 18, 19, 25, 46, 78, 102, 114\}$ dizisinde 103'ü bulalım:

Step 1 (middle element is $19 < 103$):	-1	5	6	18	19	25	46	78	102	114
Step 2 (middle element is $78 < 103$):	-1	5	6	18	19	25	46	78	102	114
Step 3 (middle element is $102 < 103$):	-1	5	6	18	19	25	46	78	102	114
Step 4 (middle element is $114 > 103$):	-1	5	6	18	19	25	46	78	102	114
Step 5 (searched value is absent):	-1	5	6	18	19	25	46	78	102	114

Arama Algoritmaları: İkili Arama (Binary Search)

```
public class BinarySearch {  
  
    public int binarySearch(int[] arr, int key) {  
  
        int low = 0;  
        int high = arr.length - 1;  
  
        while (low <= high) {  
  
            int mid = (low + high)/2;  
  
            if (arr[mid] == key)  
                return mid;  
            else if (arr[mid] < key)  
                low = mid + 1;  
            else  
                high = mid - 1;  
  
        }  
        return -1 - low;  
    }  
}
```

Arama Algoritmaları: İkili Arama (Binary Search)

- `binarySearch` metodu anahtar değerle aynı değere sahip dizi elemanı varsa o elemanın indisini döndürür. Aksi takdirde, `-insertionPoint-1` değerini döndürür.
- `insertionPoint` dizinin sıralamasına göre anahtar değerın yer alacağı indis değeridir.