



Java ile Nesne-Merkezli Programlamaya Giriş

Bölüm 6 - Diziler



Eğitmen:

Akın Kaldıroğlu

Çevik Yazılım Geliştirme ve Java Uzmanı

Konular



- **Diziler**
 - Dizi Tanımlamak
 - Elemanlara Erişim
 - for each
 - Parametre Olarak Dizi
 - Çok Boyutlu Diziler
- **Arrays Sınıfı**
- **main Metoda Parametre Geçme**

Dizi er



 selsof

build better, deliver

Torbalar (Collections)

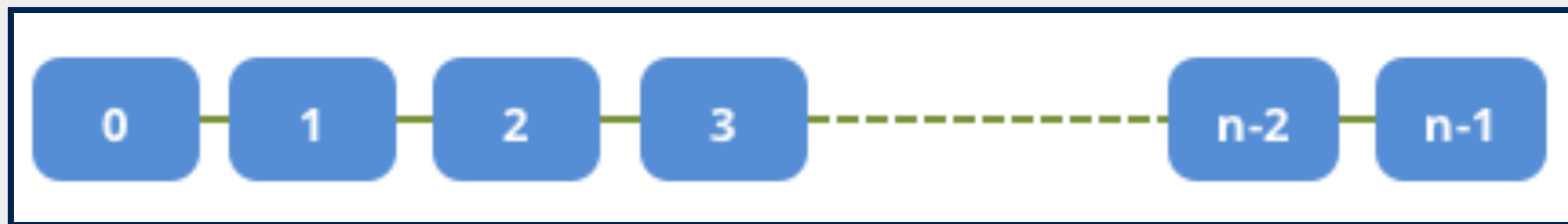


- Bütün dillerde olduğu gibi Java'da da, birden fazla ilkel değişkeni ya da nesneyi yönetmeyi sağlayan **torba (collection)** yapıları vardır.
- Java'nın `java.util` paketinde yetenekli bir **torba çerçevesi (collection framework)** vardır,
 - Bu torbaları ileride ayrı bir bölümde ele alacağız.
- Bu 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.



build better, deliver faster

Dizi Tanimlamak

Dizi Tanıtımı - I



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

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

Dizi Tanıtımı - II



- [] işaretini nerede olduğunun önemi yoktur, tipten ya da referanstan sonra olabilir.
- Yaygın kullanım tipten sonra olmalıdır.

```
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 çağrıları tanımda yapılmaz, bu çağrıyı JVM halleter.

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[] = new Student[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 seylerdir.
- 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 Zamani!





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 eksigine 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.



- 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 yapmaktadır.
 - 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 sayı = randomInt % 100;
    intArray[i] = sayı;
}
```

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 tanıtıma 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 Zamani!





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 Zamani!





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!
```

ArrayAsParameter



- org.javaturk.oopj.ch06.ArrayAsParameter

Soru ve Cevap Zamani!





build better, deliver faster

Çok Boyutlu Diziler

Çok Boyutlu Diziler - I



- Çok boyutlu diziler (multi-dimensional arrays), dizi içinde dizi olarak ifade edilirler.
- Çok boyutlu diziler, ilk dizinin her bir odasına bir başka dizi konarak elde edilir.
- Çok boyutlu diziler, matris gibi çok boyutlu yapıları ifade için kullanılabilir.

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

Çok Boyutlu Diziler - II



- Tanımlarken, sadece esas ya da ilk dizinin boyutu verilmelidir.
- Tanımlama sırasında içteki dizilerin boyutlarının verilmesine ihtiyaç yoktur.
- Çünkü içteki diziler farklı boyutlarda olabilirler.
- İki boyutlu dizi söz konusuysa, dizi, matris olmak zorunda da değildir, içteki diziler farklı boyutlarda olabilir.

Çok Boyutlu Diziler - III



- Çok boyutlu diziler kısa yolla da, doğrudan elemanları verilerek tanımlanabilir.

```
ElementType[][] name = {{...}, {...}, ..., {...}}  
  
int[][]coordinates = {{3,7}, {13,2}, {8,5}};  
  
int[][]points = {{{1,2,3}, {4,5,6}, {7,8,9}}};
```

Çok Boyutlu Dizileri İşlemek - I



- Çok boyutlu dizilerde, iç içe diziler vardır,
- En dıştaki dizinin elemanları da ayrı birer dizidir.
- Örneğin 2 boyutlu bir dizide, en dıştaki dizinin elemanları tek boyutlu bir dizidir.

```
int[][] coordinates = {{3,7}, {13,2}, {8,5}};
int[] coordinates1 = coordinates[0]; // {3,7}
int[] coordinates2 = coordinates[1]; // {3,7}
int[] coordinates3 = coordinates[2]; // {8,5}

int i1 = coordinates[0][0]; // 3
int i2 = coordinates[0][1]; // 7
int i3 = coordinates[1][0]; // 13
int i4 = coordinates[1][1]; // 2
int i5 = coordinates[2][0]; // 2
int i6 = coordinates[2][1]; // 2
```

Çok Boyutlu Dizileri İşlemek - II



- Çok boyutlu dizilerin işlenmesi için boyut sayısı kadar iç içe döngüye ihtiyaç vardır.
 - İç döngülerde, iç dizinin boyutu alınmalıdır.

Çok Boyutlu Dizileri İşlemek - III



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

```
void print(int[][][] array){  
    for(int i = 0; i < array.length; i++)  
        for(int j = 0; j < array[i].length; j++)  
            for(int k = 0; k < array[i][j].length; k++)  
                System.out.println(array[i][j][k]);  
}
```

MultiDimArray



- org.javaturk.oopj.ch06.MultiDimArray

Uygulama



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

Uygulama



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

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

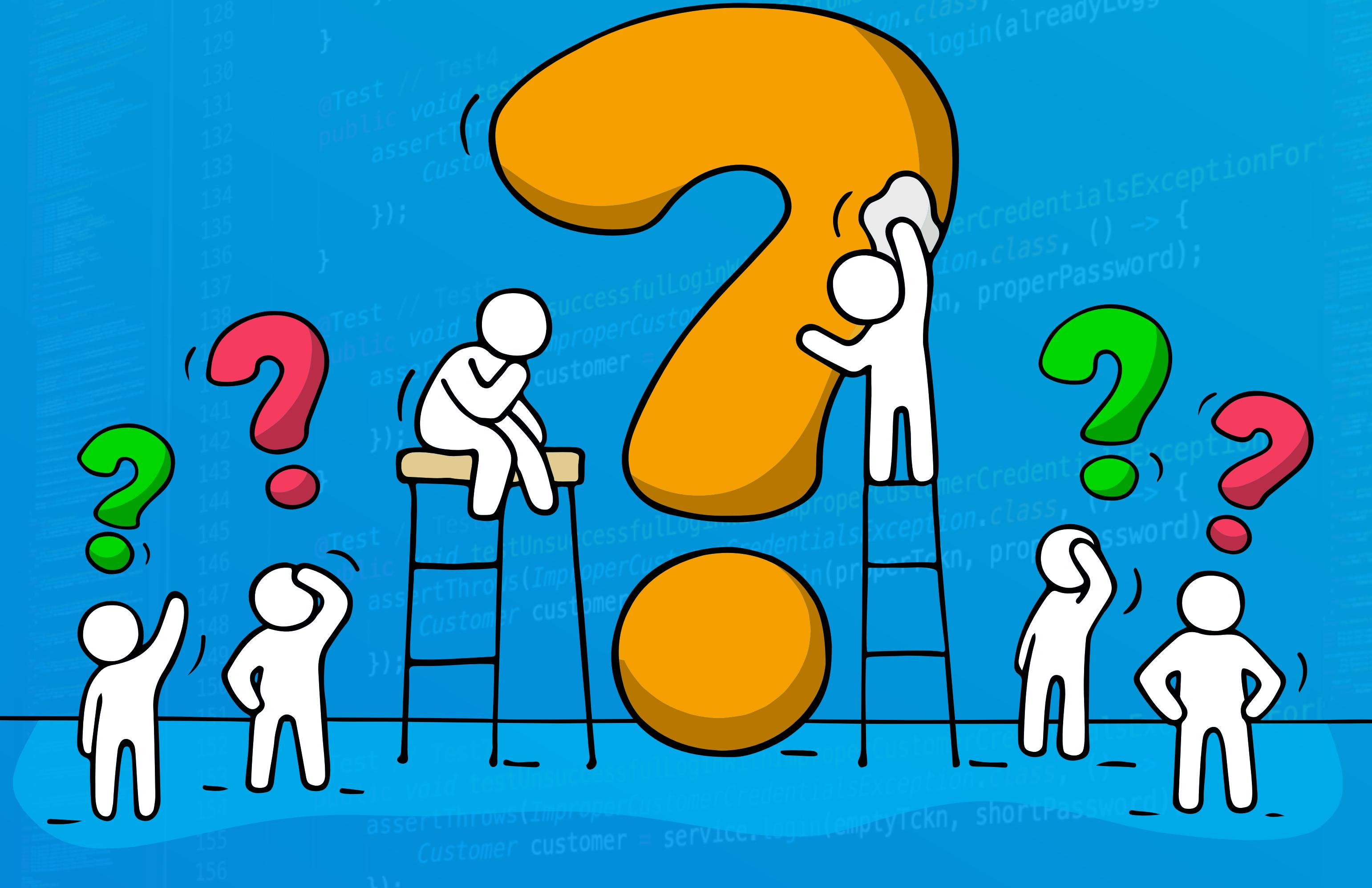
```
*  
***  
****  
*****  
*****  
*****  
*****  
*****  
*****  
*****
```

Uygulama



- Matris çarpımı yapan bir program yazınız.
 - Bu amaçla önce boyut bilgisini sonra da buna uygun olarak elemanlarını dışarıdan alarak iki tane matris oluşturun.
 - Sonra matrisin çarpımını yapın.

Soru ve Cevap Zamani!



Arrays; Sinif



selsoft

build better, deliver fast

Arrays Sınıfı



- `java.util` paketindeki **Arrays** sınıfı, dizilerle ilgili kolaylık sağlayıcı (utility) metodlara sahiptir:
 - `copyOf`
 - `copyOfRange`
 - `fill`
 - `sort`
 - `binarySearch`



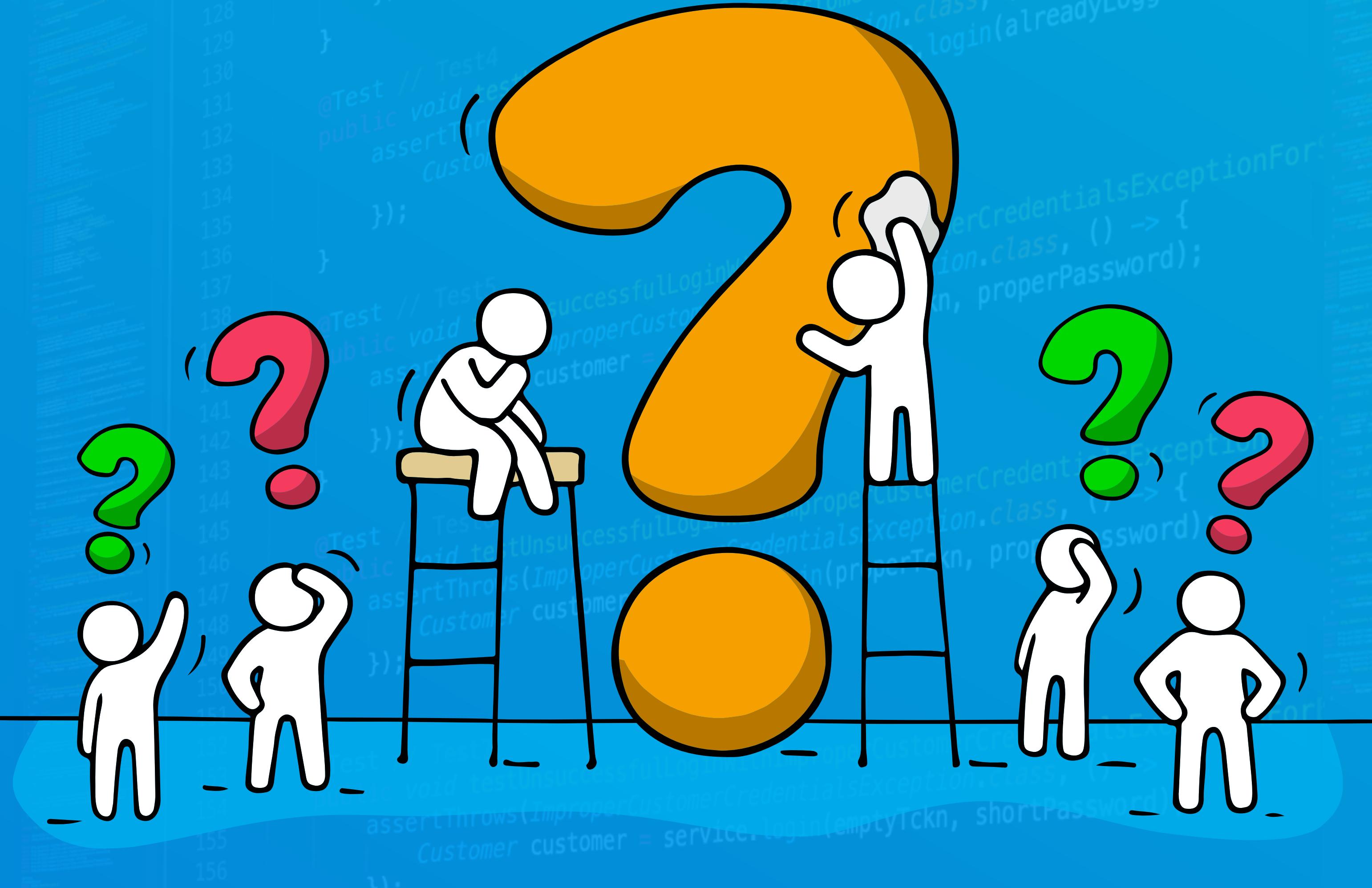
- `org.javaturk.oopj.ch06.ArrayDemo`
- **Arrays** sınıfının API'sine bakarak, metodlarının hangi türden tipler aldıklarına dikkat edin.

Sıralanamayan Sınıflar



- İlkel tipler tabi olarak sıralanabilen (naturally orderable) tiplerdir.
- Örneğin
 - `1 < 2, a < b` ya da `false < true`
- `Student` ya da `Car` gibi sınıfların nesnelerinin nasıl sıralanabilecekleri ileride ele alınacaktır.

Soru ve Cevap Zamani!



main Metota Parametre Geçme

main Metoda Parametre Geçme - I



- **main** metoda parametre geçilebilir.
- Geçilen parametreler, **main** metodun argümanı olan **String** dizisinde saklanır.

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

```
public static void main(String . . . args)
```

- Bu amaçla **main** metotlu sınıf çalıştırılırken komut satırında ya da Eclipse gibi IDE'lerde parametreler verilmelidir.

main Metoda Parametre Geçme - I



- **main** metoda geçilen parametrelerin 0'dan başlayan indislerle **String** olarak alınabileceğine, gerekirse **length** ile uzun kontrolü yapmak gerekiğine dikkat edin.

```
public static void main(String[] args){  
    if(args.length == 3){  
        String name = args[0];  
        String lastName = args[1];  
        String age = args[2];  
    }  
    else{  
        System.out.println("Please provide three arguments!");  
        System.exit(1);  
    }  
}
```

MainDemo



- `org.javaturk.oopj.ch06.MainDemo`
- Bu örneği hem komut satırından hem de Eclipse ile çalıştırın.



build better, deliver faster

Ödevler



1. Girilen bir dizideki elemanları, sıralarını bozmadan ama tekil olarak yeni bir diziye aktaran bir program yazın. Örneğin girdi {3, 3, 87, 56, 1, 87, 3, 2 } ise çıktı {3, 87, 56, 1, 2 } olmalıdır.
2. 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.



3. Bir tam sayının bölenlerini bulan bir program yazınız.
4. Bir tam sayının asal bölenlerini bulan bir program yazınız.

Fundamental theorem of arithmetic (the unique factorization theorem or the unique-prime-factorization theorem): Every integer greater than 1 either is prime itself or is the product of prime numbers, and that this product is unique, up to the order of the factors.

https://en.wikipedia.org/wiki/Fundamental_theorem_of_arithmetic

$$100 = 2^2 * 5^2, \quad 1000 = 2^3 * 5^3, \quad 17248 = 2^5 * 7^2 * 11$$

Ödevler - III



5. Yandaki arayüzü yerine getiren bir stack sınıfı yazın.

- Stack ve StackTest sınıfları assignment paketindedir.
- Stack, LIFO çalışır.

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

// Put element on the top
public boolean 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
public void showElements() {}
```

Ödevler - IV



6. Yandaki arayüzü yerine getiren bir queue sınıfı yazın.

- Queue ve QueueTest sınıfları assignment paketindedir.
- Queue, FIFO çalışır.

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

// Insert element at the bottom
public boolean 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() {...}

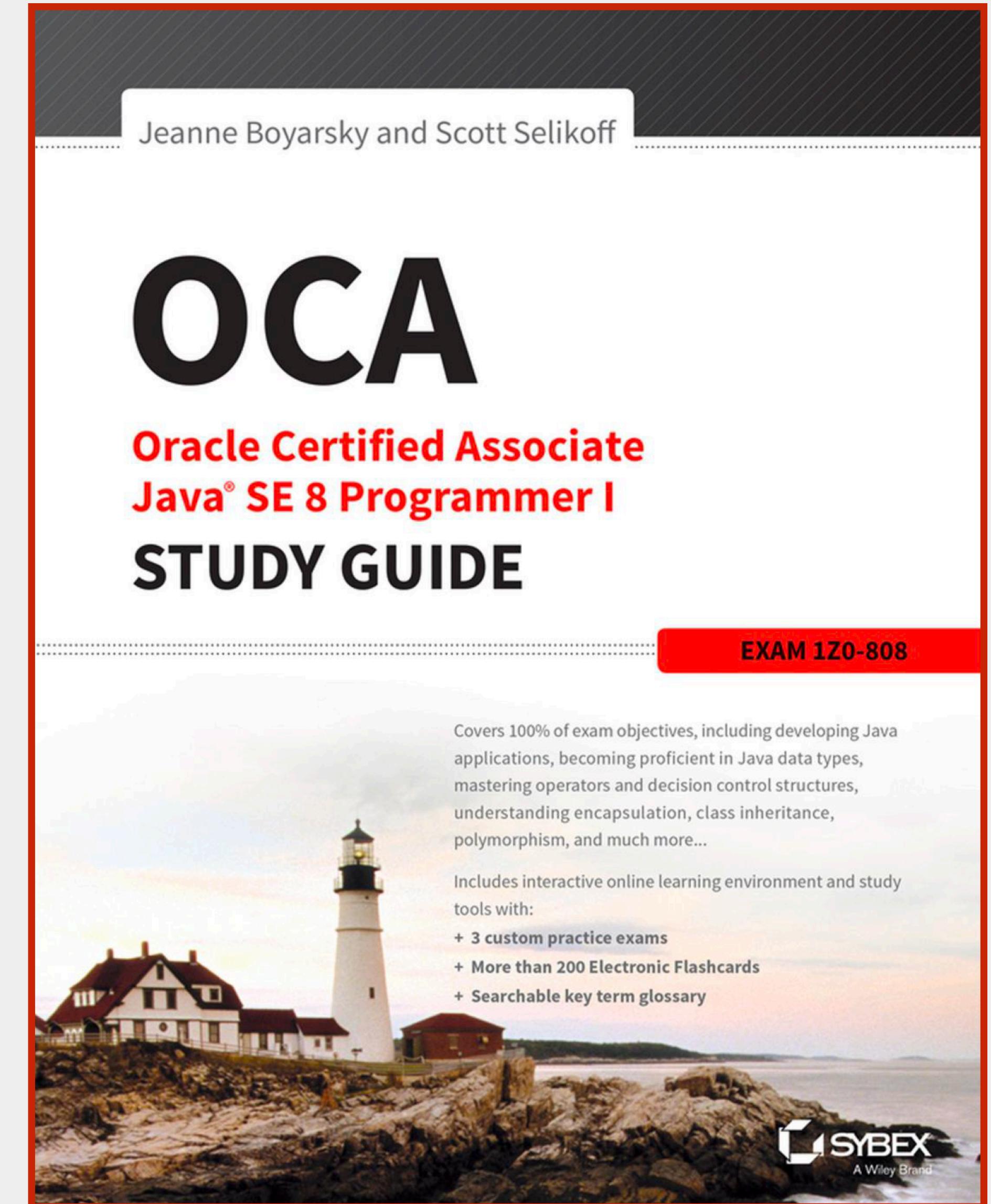
// Capacity of queue?
public int getCapacity() {...}

// Outputs the elements in the queue.
public void showElements() {}
```



7. OCA kitabından aşağıdaki soruları çözün:

- a. Kolay: (Chapter 1) 8, (Chapter 3) 15, 16
- b. Orta: (Chapter 3) 22
- c. Zor:





8. OCA/OCP kitabından aşağıdaki soruları çözün:
- Kolay: (*Chapter 1 Java Basics*) 29,
(*Chapter 3 Using Operators and Decision Constructs*) 34, 46
 - Orta:
 - Zor:

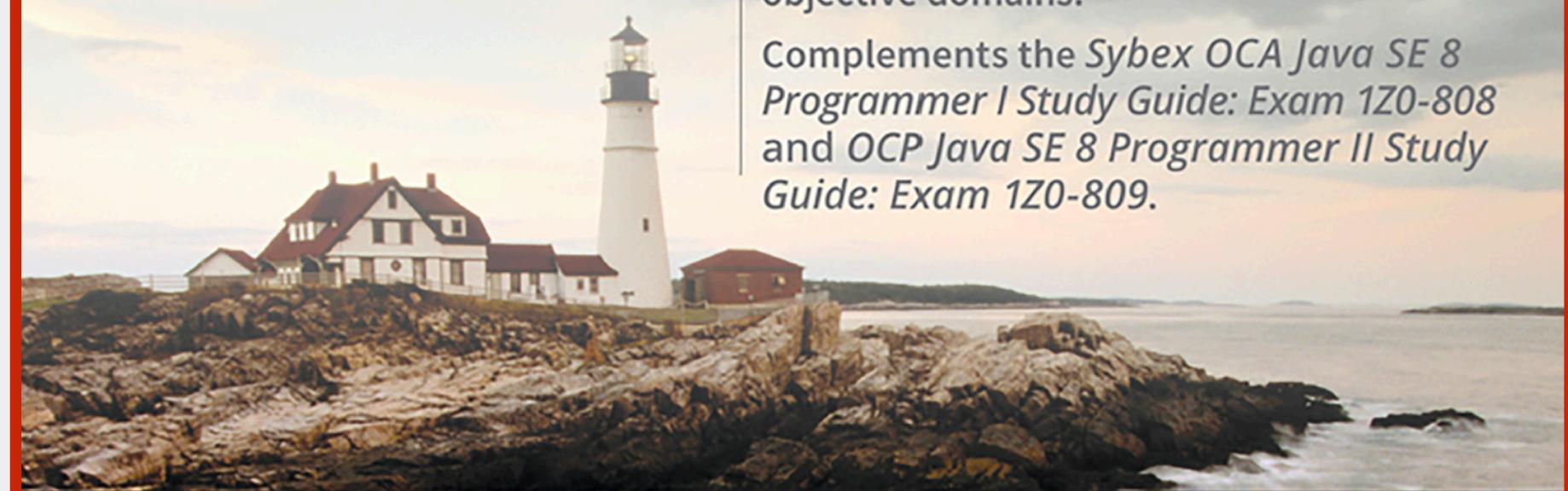
OCA/OCP JAVA® SE 8 PROGRAMMER

PRACTICE TESTS

SCOTT SELIKOFF AND JEANNE BOYARSKY

Provides 1,000 questions that include 2 practice exams covering all sections of the OCA and OCP Java SE 8 Programmer objective domains.

Complements the *Sybex OCA Java SE 8 Programmer I Study Guide: Exam 1Z0-808* and *OCP Java SE 8 Programmer II Study Guide: Exam 1Z0-809*.



Bölüm Sonu

Soru ve Cevap Zamani!

