**YMH111   
ALGORiTMA**

**ve**

**PROGRAMLAMA**

**13.HAFTA**

**(27-31 Aralık 2021)**

**STRING(SÖZCÜK) İŞLEMLERİ**

* Matematiksel bir anlam taşımayan sayı, harf veya noktalama işaretlerini ihtiva eden verilere **string** adı verilir.
* **Karakter dizisi** anlamına gelen stringler özellikle C#’ ta birer dizi olarak kullanılabilirler.
* Metin arama/düzeltme, metin doğrulama, metin biçimlendirme ve metin şifreleme gibi birçok işlemin temelinde string işlemleri vardır.
* **String** birden fazla karakter dizisini tutabilirken **char** sadece tek bir karakteri hafızada tutmak için kullanılır.
* **String veriler çift tırnak(“ “), char veriler ise tek tırnak(‘ ‘) içerisinde yazılırlar.**
* **Örnek: String S = “ METAVERSE”**

**char C = ‘E’**

**S** değişkenin gösterdiği hafıza gözü gösterimi:

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Eleman Sırası | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |  |
| Eleman İçeriği | M | E | T | A | V | E | R | S | E | **S String değişkeni hafıza yerleşim** |

**NOT: String verileri karakterlerden oluştuğu için sanki char tipinde tanımlanmış birer dizi gibi de düşünülebilir.** Javada char tipindeki dizileri string tipine çevirmek için çeşitli metotlara sahiptir.

**Örnek:**

**1.yol:**

**char [ ] Dizi** **= {‘M’,’E’,’T’,’A’};** //char başlangıç değerleri

**String S= “VERSE”;** //string başlangıç değerleri

**S= String.copyValueOf(Dizi);** //char dizisi elemanları S Stringine kopyalanıyor.

**System.out.println(S); //Ekrana META yazılır.**

**2.yol:**

**char [ ] Dizi** **= {‘M’,’E’,’T’,’A’};** //char başlangıç değerleri

**String S=new String(Dizi);** //char dizisi String değişkene

aktarılıyor.

**System.out.println(S); //Ekrana META yazılır.**

* String tiplerin istenen karakterine ulaşılabilir.
* Aşağıda tanımlı S stringi içerisinde METAVERSE verisi bulunmaktadır.
* Stringlerde karakterlere erişimi Java’da **charAt()** metodu icra eder.
* Bu gibi yöntemler **ReadOnly(sadece okunabilir)** özelliktedir.
* **Örnek:** **S.chartAt(2)** // S değişkeninin 2. elemanı(T)
* String verilerde işlem amacıyla sadece toplama operatörü kullanılır.
* Stringlerde toplamanın değişme özelliği yoktur. Aşağıdaki yapıda s3=!s4 olmadığına dikkat ediniz:
* **Örnek:** **String s1= “Meta”, s2=”Verse”, s3,s4;**

**s3= s1 + s2; // MetaVerse**

**s4= s2 + s1; // VerseMeta**

**NOT:** Cümleler içerisindeki boşluklar tıpkı herhangi bir harf veya rakam gibi hafızada yer tutar ve kapladığı bir hacim vardır.

* **Örnek:** **String s1= “Meta”, s2=”Verse”, s3,s4;**

**s3= s1 + s2; // MetaVerse=toplam 9 karakter**

**s4= s1+ “ “ + s2; // Meta Verse= toplam 10 karakter**

**NOT:** String değişkenleri sıfırlamak ve içini boşaltmak için çift tırnaklar, aralarına hiçbir değer almadan kullanılır.

* Örnek: String s1 = “ MetaVerse”;

s1=””; //s1 değişkeninin içi boşaltıldı.

**NOT:** String veriler sayı dahi içerseler matematiksel bir anlamı yoktur. Yani başta dört işlem olmak üzere diğer matematiksel fonksiyonlara da tabi tutulamazlar. Örneğin; telefon numaraları, IP adresleri, TC kimlik numaraları rakam içermelerine karşın matematiksel bir anlamı yoktur. Bilindiği üzere matematikte baştaki sıfırların bir önemi yok ve sayıların başlarına sıfır yazılamazlar. Ama bir cep telefonunda(0555…) baştaki sıfırın bir anlamı vardır ve yazılması gerekmektedir. **Yani, her sayısal tip sayısal veri içerir ama her sayı içeren veri sayısal tipten olmayabilir.**

**STRING METOTLARI**

**String S= “meta”, YS=”VERSE”;**

Tanımlamalar dikkate alınarak aş. tabloda string metotları verilmiştir.

|  |  |  |  |
| --- | --- | --- | --- |
| **İŞLEV** | **METOT** | **KULLANIM BİÇİMİ** | **ÇIKTI** |
| **Uzunluğu**  **(Karakter Sayısı)** | **length()** | **S.length()** | **4** |
| **Büyük Harfe Çevirmek** | **toUpperCase()** | **S. toUpperCase()** | **META** |
| **Küçük Harfe Çevirmek** | **toLowerCase()** | **YS. toLowerCase()** | **verse** |
| **Parça Kopyalama 1.yol: a. karakterden sonuna kadar** | **substring(a)** | **S. substring(1)** | **ETA** |
| **Parça Kopyalama 2.yol: a. Karakterden b. ye kadar** | **substring(a,b)** | **YS. substring(2,4)** | **RS** |
| **Stringleri Değiştirmek. d stringini e ile değiştir.** | **replaceAll(d,e)** | **YS. replaceAll(“E”,”A”)** | **VARSA** |

**SORU 19:** Dışarıdan girilen bir kelimenin uzunluğunu (Length metodunu kullanmadan ) bulan programı ekrana yazan algoritmanın akış diyagramını çiziniz ve java kodunu yazınız.

## İpucu: “Evrenin gizemini anlamak istiyorsanız, enerji, frekans ve titreşim cinsinden düşünün.” Girilen cümlenin sonuna sonlandırıcı sıfır veya başka bir karakter ekleyerek bu karakteri bulana kadar döngüyü devam ettiririz. Sona eklenecek sonlandırma karakteri ASCII sıfırın karakter karşılığıdır. Sıfırın karakter karşılığını bulmak için “(char)0” dönüşümü yapılır. Bulduğumuzdaki harf indisimiz kelimenin uzunluğunu verecektir.

**ÇÖZÜM:**

Giriş:

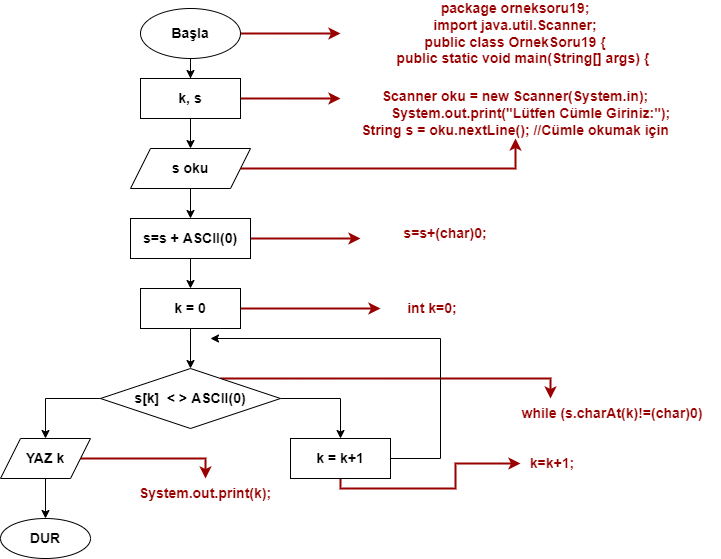
Çıkış:

Matematik:

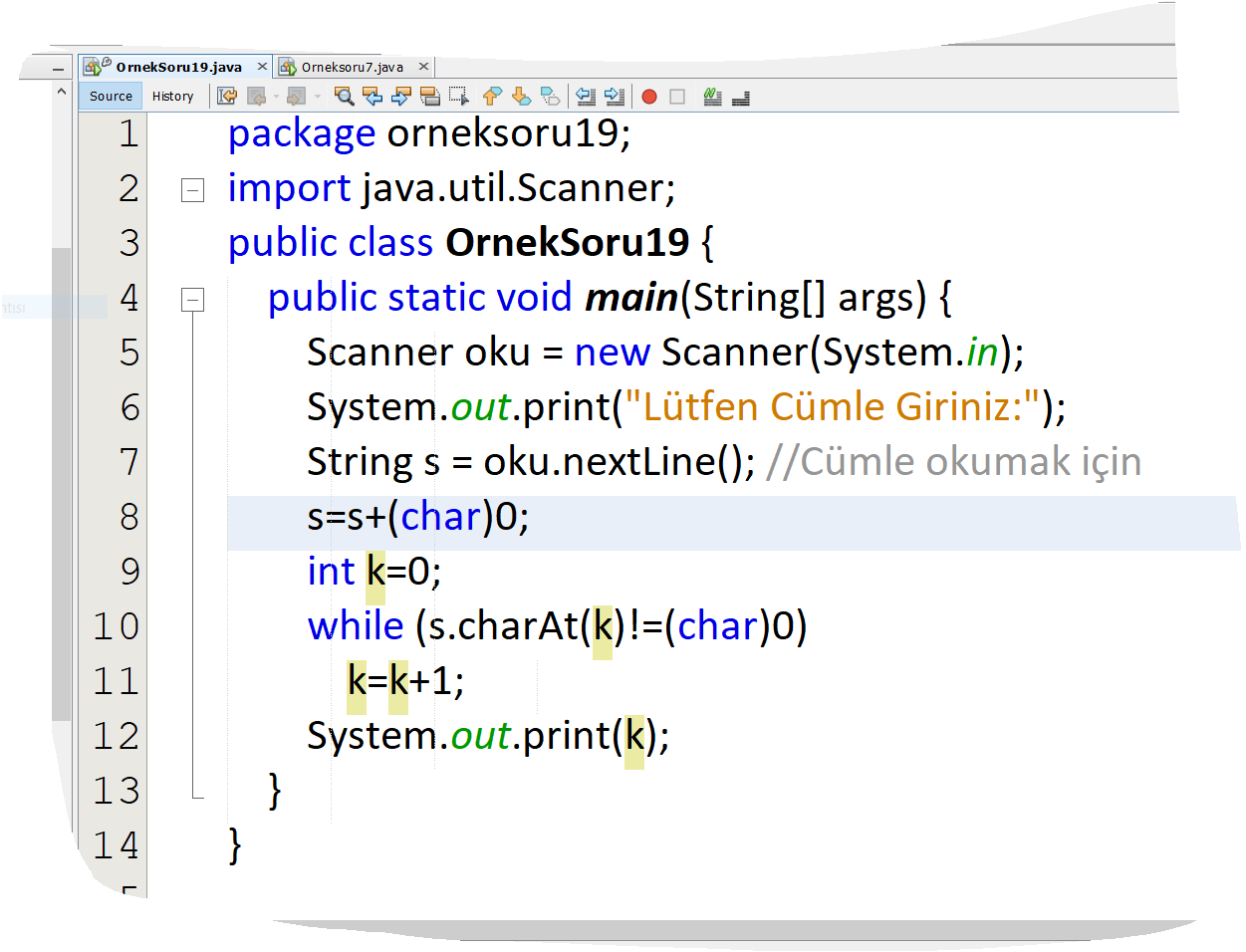
Koşullu Yürütme:

Tekrar:

* Kontrol Değişkeni (Alt Sınır):
* Kontrol Koşulu (Üst Sınır):
* Döngü Gövdesi:
* Sonlandırma İfadesi:

**AKI Ş DİYAGRAMI ve KOD BLOĞU:**

**KOD:**



**SORU 20:** Dışarıdan girilen bir cümleyi ekrana kelime kelime yazan programın akış şemasını çiziniz ve java kodunu yazınız.

**İpucu: “JAMES WEBB SPACE TELESCOPE”** Cümle içerisinde kelimeleri ayıran karakter boşluklar olduğu için, cümle içerisinde boşluğa rastlayınca alt satıra geçmek problemi çözecektir.

**ÇÖZÜM:**

Giriş:

Çıkış:

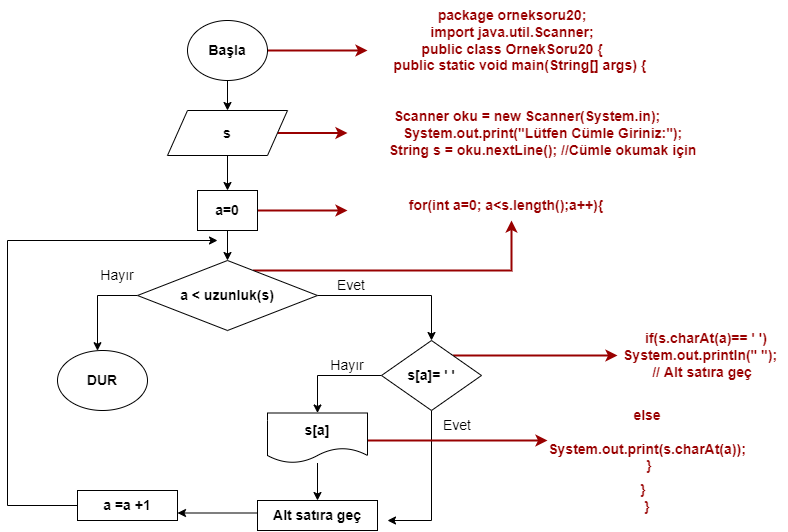
Matematik:

Koşullu Yürütme:

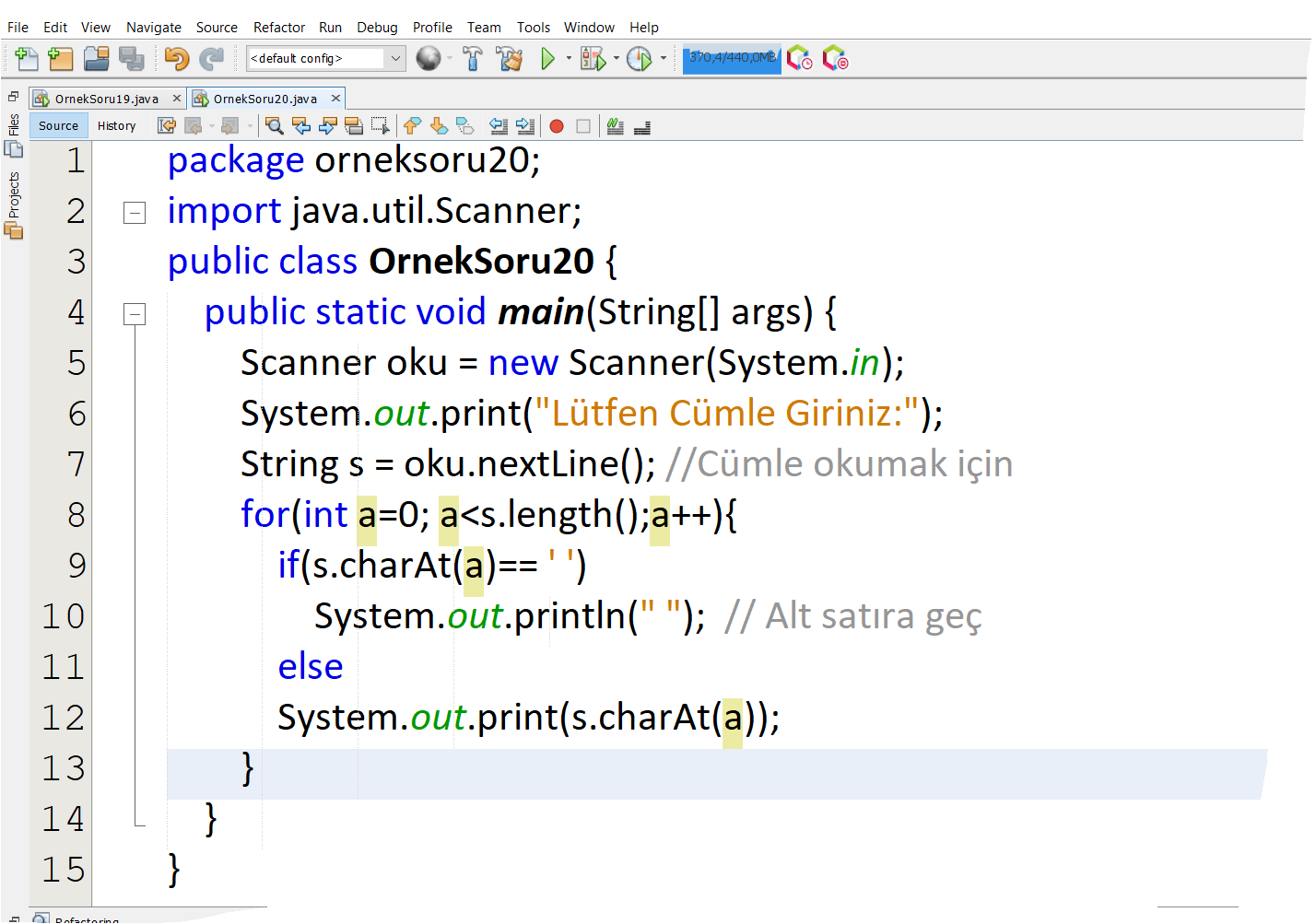
Tekrar:

* Kontrol Değişkeni (Alt Aralık):
* Kontrol Koşulu (Üst Aralık):
* Döngü Gövdesi:
* Sonlandırma İfadesi:

**AKIŞ DİYAGRAMI ve KOD BLOĞU:**



**KOD**:



**DİZLER VE MATRİS NEDİR?**

* Bellekte **ardışıl** olarak yer kaplayan veri kümesine **dizi(array)** denir.
* Diziyi oluşturan veri kümeleri(elemanlar) **hafızada bitişik olarak sıralı bir şekilde yerleşir.**
* Diziler birden fazla değişken tanımı gerektiğinde çok büyük kolaylık sağlar.
* Dizi ve değişken arasında tanımlanma açısından fark vardır. **Her ikisi de hafıza gözünü(adresini)** gösterir.
* **Değişken** **yalnız tek bir hafıza gözünü gösterirken, dizi birden fazla bitişik hafıza gözünü gösterir.**
* **Örneğin,** aşağıdaki tamsayı içeren hafıza gözlerindeki verileri, ilk olarak değişkenle tanımlandığını düşünelim. Bildiğimiz gibi x’ in değeri 25, y’ nin değeri 300…şeklinde ifade edilir. Dizi şeklinde ifade edilseydi tüm verileri A dizisinin elemanları(5 elemanlı) şeklinde düşünüp, A’ nın sıfırıncı elemanı 25, A’ nın birinci elemanı 300…şeklinde tanımlıyoruz.
* A’ nın elemanlarını gösteren **indisler** köşeli parantezler **“[ ]”** arasına yazılır.
* **Dizi indisi(adresi/göstericisi) başlangıç değeri olarak sıfırdan başlar.** **Yani ilk elemanın adresi A dizisi için [0] dır.**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **x y z t k** | | | | |
| **25** | **300** | **1209** | **0** | **45** |
| **A[0] A[1] A[2] A[3] A[4]** | | | | |

**Değişken yapı ile gösterim:** x, y, z, t, k…

**Veri hafıza gözeleri:** 25, 300, …

**Dizi yapısı ile gösterim:** A[0], A[1], …

* Diziler çoklu değişken kullanımlarında büyük kolaylıklar sağladığı için tercih edilir.

Örnek: Sınıfımızdaki 40 kişinin isimlerini ve fizik notlarını programa girip **isteğe göre**, en yüksek notu, notlara göre sıralamayı, ilk üç kişiyi, notların ortalamasını ve kaç kişinin kalıp geçtiğini bulmak istiyoruz. Bunun için 40 isim ve 40 fizik notunu **hafızada devamlı tutmamız gerekecektir.** Değişken ve dizi yaklaşımıyla problemin çözümü:

|  |  |  |
| --- | --- | --- |
| **Aradaki Farklar** | **Değişken Yapı** | **Dizi Yapı** |
| Tanımlama farkı | **String** isim1, isim2, … ,isim40  **int** not1, not2, …, not40 | **String**[ ] isim=new String[40]  **int**[ ] notlar=new int [40] |
| İsim ve not giriş farkı | Oku isim1; Oku not1;  Oku isim2; Oku not2;  … …  Oku isim40; Oku not40; | for(a=0; a<=39; a++){  Oku isim [a];  Oku notlar [a];  } |
| En büyük notu bulma farkı | if (buyuk<not1)  ….  if (buyuk<not40) | for(a=0; a<=39; a++){  if (buyuk<not[a]){  buyuk=not[a];}} |

Yukarıdaki farklardan da görüleceği üzere dizi tanımlamanın değişken tanımlamadan bir farkı yok fakat avantajı çoktur. Dizi isterse 100.000 elemanlı olsun kullanımı basit. Sadece sayıların değerini değiştirmek yeterli.

* Diziler yukarıdaki gibi tek boyutlu olabileceği gibi birden fazla boyuta da sahip olabilir. Dizinin bir elemanını kaç indis ile gösterirsek dizi o kadar boyutludur.
* Örneğin iki boyutlu diziler yaygın olarak resim işlemede kullanılır. Resmi oluşturan piksellerin yerini tarif etmek için x ve y parametrelerine ihtiyaç duyarız. Dolayısıyla bir resmin tüm piksel değerleri 2 boyutlu bir dizi içerisine yerleşebilir.
* Aşağıda 3 x 3’ luk A dizisini ele alalım. **A[1,2] elemanında 1 rakamı satırı ve 2 rakamı da sütunu** gösterir. Bu eleman 1. satır ve 2. sütun elamanıdır. Aşağıda bu hücre 1.satır ve 2.sütunun kesişim noktası olarak renklendirilmiştir. Satırlar **yatay,** sütunlar ise **düşeyde** ki bloklardır. Şimdi **A dizisi, 3x3’ lük bir matrisi temsil etmek için kullanılabilir**.

|  |  |  |  |
| --- | --- | --- | --- |
|  | 0.sütun | 1.sütun | 2.sütun |
| 0.satır | A[0,0] | A[0,1] | A[0,2] |
| 1.satır | A[1,0] | A[1,1] | A[1,2] |
| 2.satır | A[2,0] | A[2,1] | A[2,2] |

3x3 lük 2 boyutlu matris yapısı

* Bilgisayardaki dizi kavramının matematikteki karşılığı **matrislerdir.**
* Daha çok resim işleme uygulamalarında kullanılırlar. Resmi hafızaya almak, efektler vermek, döndürmek gibi işlemlerde,…
* Bunun dışında matematikçiler için lazım olan bir matrisin tersi, transpozu, matrislerin çarpımı gibi tüm problemlerin ve lineer denklem sistemlerinin çözümünde,…

**DİZİLERİN JAVADA TANIMI**

**1.YOL:** Dizilere sabit bir eleman sayısının verildiği tanımlama şeklidir. Eleman sayısı sonradan **değiştirilemez.** Diziler, elemanı kadar tanımlı olduğu tipe ait verileri tutar.

Dizi\_Tipi **[ ]** Dizi\_Adı = **new** Dizi\_Tipi **[**Eleman\_Sayısı**]**

Aşağıda örnek dizi tanımları verilmiştir.

**int** [ ] A= **new int** [10]; //10 elemanlı tek boyutlu integer tipinde A dizisi.

**double** X [ ] = **new double**[100]; //100 elemanlı tek boyutlu double tipinde X dizisi.

**String** [ ] S= **new String**[40]; //40 elemanlı tek boyutlu String tipinde S dizisi.

**Char** harf [ ][ ] = **new char** [3][4]; //3x4=12 elemanlı iki boyutlu char tipinde harf dizisi.

**int** [ ] [ ] R= **new int** [10][10]; //10 X10=100 elemanlı iki boyutlu integer tipinde R dizisi.

**ÖNEMLİ NOT:** **Köşeli parantezler dizi adının önünde veya arkasında yer alabilir. Java’ da bu durum hataya yol açmaz.**

**2.YOL:** Dizinin hafıza gözlerine başlangıç değerlerinin atanıp bu değer sayısı kadar eleman sayısının ima edildiği tanımlama şeklidir.

**int** D [] = **{**1,2,3,4,5**}**; //5 elemanlı tek boyutlu integer tipinde D dizisi.

**String** [] isim = **{**“Ali”, “Veli”, “Ahmet”, “Mehmet”**}**; //4 elemanlı tek boyutlu String tipinde isim dizisi.

**int** K[] [] = **{{**1,2,3**}**,**{**4,5,6**}}**; //6 elemanlı iki boyutlu integer tipinde K dizisi.

**DİZİLERİN KULLANIMI**

1. Diziler hafızaya toplu halde yerleştikleri için tanımlanan bir dizinin eleman sayısı programın hafızada kapladığı alana doğrudan etki edecektir. Dizi boyutunu belirlerken ince eleyip sık dokumak gerekir.
2. Elemanları gösteren sayıların (indis) **sıfırdan başlayıp**, **eleman sayısının bir eksiğine kadar** devam etmesidir. Bu alt ve üst sınırlara azami dikkat edilmesi gerekir.
3. Dizilere eleman atarken ve dizinin elemanlarını kullanırken dizi boyutu sayısınca iç içe for döngüsü kullanma gereği hemen aklımıza gelmelidir. Örneğin **iki boyutlu** dizilere değer atamak için, **iki for** döngüsü iç içe kullanılmalıdır.

**DİZİLERE AİT METOTLAR**

1. **Dizinin eleman sayısını bulmak için length metodu**

**kullanılır. Dizi\_Adı.length**

**Örnek:**

**int [ ] A = new int [10];**

**System.out.println(A.length); // Ekrana 10 sayısını yazacaktır.**

1. **A dizisinin, B dizisine istenilen kısmını kopyalamak için:**

**Hedef\_Dizi = Arrays.copyOf(Kaynak\_Dizi, Uzunluk)**

**Not: Bu metodu kullanmak için “java.util.Arrays” import edilmelidir.**

**Örnek:**

**int[ ] A = {1, 2, 3, 4, 5};**

**int[ ] B = new int[5];**

**B = Arrays.copyOf(A,5);**

**System.out.println(B[2]); //Ekrana 3 sayısını yazacaktır.**

**SORU 21: 1- 99** arasında girilen sayıları yazı ile yazdıran programın akış şemasını çiziniz ve java kodunu yazınız.

**İpucu: 1-9 arasındaki rakamların yazı karşılığını Bir dizisine 10,20,30, … ,90’ kadarkilerin yazı karşılığını On dizisine alalım. Sonra sayımızın basamaklarındaki sayıları bularak bunlara karşılık gelen On ve Bir dizilerindeki elemanların değerlerini yazdıralım. 20,30,…, gibi sayılardaki son basamaktaki sıfırı yazmamak için şartla belirtmemiz gerekmektedir.**

ÇÖZÜM:

Giriş:

Çıkış:

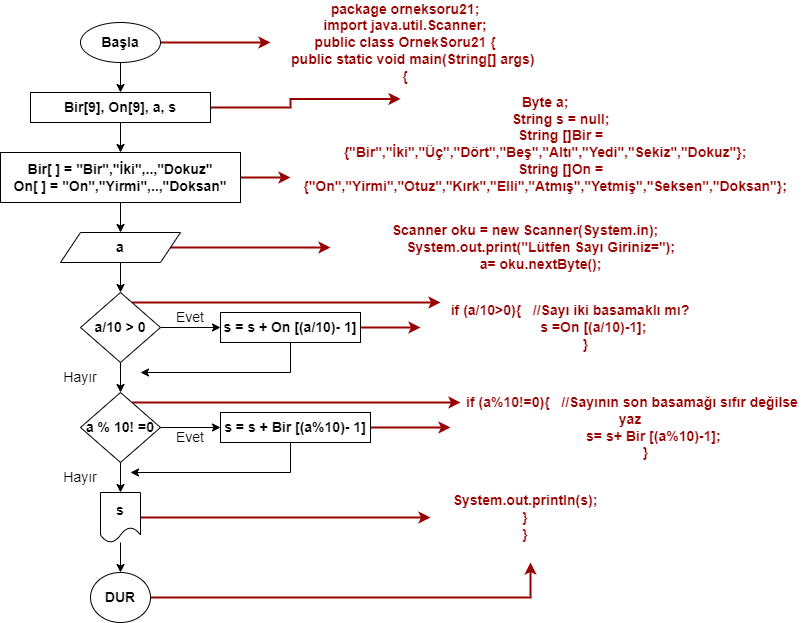
Matematik:

Koşullu Yürütme:

Tekrar:

* Kontrol Değişkeni (Alt Aralık):
* Kontrol Koşulu (Üst Aralık):
* Döngü Gövdesi:
* Sonlandırma İfadesi:

**AKIŞ DİYAGRAMI ve KOD BLOĞU:**



**KOD:**

