

İleri Programlama

String İşlemler

Hüseyin Ahmetoğlu

Kodlama Hataları

- ▶ Derleme Hatası
- ▶ Çalışma Zamanı Hatası
- ▶ Mantıksal Hatalar

Derleme Hatası

- *Compilation Error (or Syntax Error)*: Program derlenemiyor. Bu hata, derleme hata mesajlarını kontrol ederek kolayca düzeltilebilir.

```
// System instead of Sys
Sys.out.print("Hello");
error: package Sys does not exist
Sys.out.print("Hello");
    ^

// Missing semi-colon
System.out.print("Hello")
error: ';' expected
System.out.print("Hello")
                        ^
```

Çalışma zamanı hatası

- *Runtime Error*: Program derlenebilir, ancak başarıyla çalışmaz. Bu, çalışma zamanı hata mesajlarını kontrol ederek de kolayca düzeltilebilir.

```
// Divide by 0 Runtime error
int count = 0, sum = 100, average;
average = sum / count;
Exception in thread "main" java.lang.ArithmeticException: / by zero

// Wrong conversion code, %d for integer
System.out.printf("The sum is: %d\n", 1.23);
Exception in thread "main" java.util.IllegalFormatConversionException: d != java.lang.Double
```

Mantıksal hatalar

- ▶ *Logical Error*: Program derleyip çalıştırabilir ancak yanlış sonuçlar üretir (her zaman veya bazen). Hata mesajı olmadığından düzeltilmesi en zor hatadır - çıktının kontrolüne güvenmeniz gerekir. Programın her zaman yanlış çıktı üretip üretmediğini tespit etmek kolaydır. Programın çoğu zaman doğru sonucu üretip üretmediği durumlarda düzeltmek son derece zordur.
- ▶ Hata mesajı üretimden önce yakalanmazsa bu tür hatalar çok ciddidir. İyi programlar yazmak, bu hataları en aza indirmeye ve tespit etmede son derece önemlidir. Programın doğruluğunu tespit etmek için iyi bir test stratejisine ihtiyaç vardır.

```
// Can compile and run, but give wrong result - sometimes!  
if (mark > 50) {  
    System.out.println("PASS");  
} else {  
    System.out.println("FAIL");  
}
```

char – aritmetik işlemler

- ▶ Java'da her bir karakter 16 bitlik bir Unicode numarasıyla temsil edilir. Karakter '0' kod numarası 48 (30H), karakter '1' 49 (31H), karakter 'A' 65 (41H) karakter 'a' 97 (61H) ile temsil edilir. Char '0' ın int 0 olmadığını, char '1'in int 1 olmadığını unutmayın.
- ▶ Karakterler aritmetik işlemlerde yer alabilir. Bir karakter aritmetik işlemlerde temel alınan int ([0, 65535] aralığında) olarak kabul edilir. Başka bir deyişle, char ve int birbirinin yerine kullanılabilir. karakter '0' \Leftrightarrow int 48, karakter '1' \Leftrightarrow int 49, karakter 'A' \Leftrightarrow int 65, karakter 'a' \Leftrightarrow int 97.

char – aritmetik işlemler

```
char c1 = '0';      // Code number 48
char c2 = 'A';      // Code number 65
char c3;

// char <-> int (interchangeable)
System.out.println((int)c1);  // Print int 48
System.out.println((int)c2);  // Print int 65
c3 = 97;                  // Code number for 'a'
System.out.println(c3);      // Print char 'a'
System.out.println((char)97); // Print char 'a'
```

char – aritmetik işlemler

- ▶ Aritmetik işlemlerde, char (ve byte ve short) önce int türüne dönüştürülür. Java'da aritmetik işlemler yalnızca int, long, float veya double olarak gerçekleştirilir
- ▶ Bu nedenle, $\text{char} \oplus \text{char} \Rightarrow \text{int} \Rightarrow \text{int} \Rightarrow \text{int}$, burada \oplus bir ikili aritmetik işlemi belirtir (+, -, *, / ve % gibi). Sonuçta ortaya çıkan int'i karaktere geri çevirmeniz gerekebilir.

```
char c1 = '0';    // Code number 48
char c2 = 'A';    // Code number 65
char c3;

// char + char -> int + int -> int
//c3 = c1 + c2;    // error: RHS evaluated to "int", cannot assign to LHS of "char"
c3 = (char)(c1 + c2); // Need explicit type casting, return char 'q' (code number 113)
System.out.println(c3);    // Print 'q', as c3 is a char
System.out.println(c1 + c2); // Print int 113
System.out.println((char)(c1 + c2)); // Print char 'q'
```


char – aritmetik işlemler

- Benzer, $\text{char} \oplus \text{int} \Rightarrow \text{int} \oplus \text{int} \Rightarrow \text{int}$. Sonuçta ortaya çıkan int'i açık tip dönüşümü ile karaktere geri çevirmeniz gerekebilir. Örneğin,

```
char c1 = '0';    // Code number 48
char c2 = 'A';    // Code number 65
char c3;

// char + int -> int + int -> int
//c3 = c1 + 5;    // error: RHS evaluated to "int", cannot assign to LHS of "char"
c3 = (char)(c1 + 5); // Need explicit type casting, return char '5' (code number 53)
System.out.println(c3);    // Print '5', as c3 is a char
System.out.println(c1 + 5); // Print int 53

// Print the code number for 'a' to 'z'
for (int codeNum = 'a'; codeNum <= 'z'; ++codeNum) {
    System.out.println((char)codeNum + ": " + codeNum);
}
```

char – aritmetik işlemler

- Bileşik atama operatörleri için (+=, -=, *=, /=, %= gibi) değerlendirme int olarak gerçekleştirilir, ancak sonuç otomatik olarak LHS'ye geri döndürülür.
- Karakter artışı (++) ve eksiltme (--) için (ve byte ve short), int'e yükseltme yoktur.

```
char c4 = '0';           // Code number 48
c4 += 5;                 // Automatically cast back to char '5'
System.out.println(c4); // Print char '5'
```

```
// Print char '0' to '9' via increment
for (char c = '0'; c <= '9'; ++c) { // ++c remains as "char"
    System.out.println(c);
}
```

String'ler

String değişkenler, içerisinde her türlü karakterin (alfabetik, sayısal, vb.) saklandığı sözel tip değişkenlerdir. Sözel tip değişkenler, “String” ya da ‘char’ veri tipi ile ifade edilirler. Daha önce de belirtildiği gibi String veri tipi birden fazla karakteri, ‘char’ veri tipi ise tek bir karakteri ifade eder. **String veriler** çift tırnak içerisinde “...” yazılırken, **char veriler** tek tırnak ‘...’ içerisinde yazılırlar. Stringler’de diziler gibi her bir elemanı ayrı bir nesne gibi algılanır.

`String Ad= "HANZAR";` şeklindeki bir string ifadenin her bir elemanına aynı dizilerde olduğu gibi indis numarası ile erişebiliriz. Ad değişkeninin her bir karakterine `Ad.charAt(indis_no)` deyimi ile erişebiliriz.

İndis no	0	1	2	3	4	5
String ifade	H	A	N	Z	A	R

String değer atama ve birleştirme

String'leri ilk yapılandırırken `new` deyimini kullanmanız gerekmez. Java otomatik olarak String nesnesini yapılandırır, yani `" "` (çift tırnak) işaretleri arasında kalan ifade `"="` operatörü ile değişkene aktarılır. Aşağıdaki iki tanımlamada aynı işlevi yapar ve `Ad` değişkenine `"HANZAR"` ifadesi aktarılır.

```
String Ad = "HANZAR";  
// Ad değişkenine "HANZAR" ismi aktarılır  
String Ad = new String("HANZAR");  
// Ad değişkenine "HANZAR" ismi aktarılır
```

String ifadeleri birleştirmek için `+` operatörü kullanılır.

Örneğin;

```
System.out.println("Ada"+"Pazarı"); //Ekrana AdaPazarı yazar.  
String gun = "31 ";  
String ay = "Aralık";  
String bugün = gun + ay;           // Sonuç "31 Aralık"
```

veya

```
String date = "31";  
String date += "Aralık";           // Sonuç "31 Aralık"
```

String işlemleri

- ▶ ***str.length ()***: str uzunluğunu döndürür.
- ▶ ***str.charAt (int index)***: String'in index konumundaki karakteri döndürür. Dizinin 0'dan başladığını ve str.length () - 1'e kadar olduğunu unutmayın.
- ▶ ***str1.equals (str2)***: str1 ve str2 içeriklerini karşılaştırmak için. İki String'i karşılaştırmak için "str1 == str2" kullanamayacağınızı unutmayın. Bunun nedeni, "==" ifadesinin yalnızca ilkel türler için geçerli olması, ancak String 'in ilkel bir tür olmamasıdır.

String işlemleri

```
String str = "Java is cool!";  
System.out.println(str.length());           // return int 13  
System.out.println(str.charAt(2));           // return char 'v'  
System.out.println(str.charAt(5));           // return char 'i'  
  
// Comparing two Strings  
String anotherStr = "Java is COOL!";  
System.out.println(str.equals(anotherStr));   // return boolean false  
System.out.println(str.equalsIgnoreCase(anotherStr)); // return boolean true  
System.out.println(anotherStr.equals(str));   // return boolean false  
System.out.println(anotherStr.equalsIgnoreCase(str)); // return boolean true  
// (str == anotherStr) to compare two Strings is WRONG!!!
```

String işlemleri

```
String str = "Java is cool!";  
System.out.println(str.length());           // return int 13  
System.out.println(str.charAt(2));           // return char 'v'  
System.out.println(str.substring(0, 3));     // return String "Jav"  
System.out.println(str.indexOf('a'));        // return int 1  
System.out.println(str.lastIndexOf('a'));    // return int 3  
System.out.println(str.endsWith("cool!"));  // return boolean true  
System.out.println(str.toUpperCase());       // return a new String "JAVA IS COOL!"  
System.out.println(str.toLowerCase());      // return a new String "java is cool!"
```

contains(String degisken) : String ifade içersinde istenilen bir karakter veya kelimenin bulunup bulunmadığını geriye **true/false** değerini döndürerek bildiren bir fonksiyondur.

replace(x,y) : String içindeki x ile tanımlı her karakteri, y ile tanımlı karakterle değiştirir.

trim() : String ifadenin başındaki ve sonundaki boşluk ve “tab” karakterlerini kaldırır.

codepointAt(indis) : String ifadede **indis** pozisyonundaki harfin byte değerini geri döndürür. İlk karakterin pozisyonu 0’dır.

copyValueOf(Değişken) : Bir karakter dizisinin tamamını ya da belli bir parçasını String değişkene kopyalamak için kullanılır.

startsWith ve endsWith: Bir string ifadenin başlangıç ve bitişini kontrol etmek veya başlangıç ve bitiş değerine göre işlem yapmak istediğimizde bu metotları kullanırız.

Örnek

Örnek 6.42: Klavyeden girilen bir cümledeki toplam kelime sayısını ve “ve” bağlaçlarının sayısını bulan programı yazınız.

Çözüm:

```
import java.util.Scanner;
public class KelimeSayisi {
    public static void main(String[] args) {
        Scanner tara=new Scanner (System.in);
        System.out.println("Cümleyi giriniz");
        String cumle=tara.nextLine();
        int s=1,s1=0;
        for(int i=0;i<cumle.length();i++)
        {
            // kelime sayısını elde etmek için
            if(cumle.charAt(i)==' ')s++;
            //ve bağlaç sayısı için
            if(i<=cumle.length()-4) if(cumle.charAt(i)==' '
            && cumle.charAt(i+1)=='v' &&
            cumle.charAt(i+2)=='e' && cumle.charAt(i+3)==' ')
                s1++;
        }
        System.out.println("Toplam "+s+" kelime vardır.");
        System.out.println("Toplam "+s1+" ve bağlacı vardır.");
    }
}
```


String'i primitive'e dönüştürme

```
String inStr = "5566";
int number = Integer.parseInt(inStr);    // number <- 5566
// Input to Integer.parseInt() must be a valid integer literal
// number = Integer.parseInt("abc");      // Runtime Error: NumberFormatException
```

```
String inStr = "55.66";
float aFloat = Float.parseFloat(inStr);    // aFloat <- 55.66f
double aDouble = Double.parseDouble("1.2345"); // aDouble <- 1.2345
aDouble = Double.parseDouble("1.2e-3");    // aDouble <- 0.0012
// Input to Integer.parseInt() must be a valid double literal
// aDouble = Double.parseDouble("abc");     // Runtime Error: NumberFormatException
```

```
// Extract each char
String msg = "Hello, world";
char msgChar;
for (int idx = 0; idx < msg.length(); ++idx) {
    msgChar = msg.charAt(idx);
    // Do something about the extracted char
    .....
}
```

```
String boolStr = "true";
boolean done = Boolean.parseBoolean(boolStr); // done <- true
boolean valid = Boolean.parseBoolean("false"); // valid <- false
```

String Tip Dönüşümleri

Java’da her temel (basit) veri tipinin bir **nesnesel veri tipi** (*wrapper class*) karşılığı vardır. Bunların listesi aşağıda verilmiştir. Tip dönüşümlerinde de bu nesnesel veri tipi karşılıkları kullanılmaktadır. Örneğin, **integer** veri tipini **String** veri tipine dönüştürürken ‘int’ değil de ‘Integer’ kullanılır. (“Integer.toString(parametre)” gibi.)

Temel Veri Tipleri	Nesnesel Veri Tip Karşılığı (Wrapper Class)
byte	Byte
short	Short
int	Integer
long	Long
double	Double
float	Float
boolean	Boolean
char	Character

Primitive'i String'e dönüştürme

```
// Using String concatenation operator '+' with an empty String (applicable to ALL primitive types)
String str1 = 123 + "";    // int 123 -> String "123"
String str2 = 12.34 + "";  // double 12.34 -> String "12.34"
String str3 = 'c' + "";    // char 'c' -> String "c"
String str4 = true + "";   // boolean true -> String "true"

// Using String.valueOf(aPrimitive) (applicable to ALL primitive types)
String str5 = String.valueOf(12345);    // int 12345 -> String "12345"
String str6 = String.valueOf(true);      // boolean true -> String "true"
String str7 = String.valueOf(55.66);     // double 55.66 -> String "55.66"

// Using toString() for each primitive type
String str8 = Integer.toString(1234);    // int 1234 -> String "1234"
String str9 = Double.toString(1.23);     // double 1.23 -> String "1.23"
String str10 = Character.toString('z');  // char 'z' -> String "z"
```

Örnek: Tersten yazdırma

```
import java.util.Scanner;
/**
 * Prompt user for a string; and print the input string in reverse order.
 */
public class ReverseString {
    public static void main(String[] args) {
        // Declare variables
        String inStr;    // input String
        int inStrLen;    // length of the input String
        Scanner in = new Scanner(System.in);

        // Prompt and read input as "String"
        System.out.print("Enter a String: ");
        inStr = in.next();
        inStrLen = inStr.length();

        System.out.print("The reverse is: ");
        // Use a for-loop to extract each char in reverse order
        for (int inCharIdx = inStrLen - 1; inCharIdx >= 0; --inCharIdx) {
            System.out.print(inStr.charAt(inCharIdx));
        }
        System.out.println();
        in.close();
    }
}
```

Enter a String: abcdefg
The reverse is: gfedcba

Örnek: Binary Kontrol

```
import java.util.Scanner;
/**
 * Check if the input string is a valid binary string.
 */
public class ValidateBinString {
    public static void main(String[] args) {
        // Declare variables
        String inStr;    // The input string
        int inStrLen;    // The length of the input string
        char inChar;     // Each char of the input string
        boolean isValid; // "is" or "is not" a valid binary string?
        Scanner in = new Scanner(System.in);

        // Prompt and read input as "String"
        System.out.print("Enter a binary string: ");
        inStr = in.next();
        inStrLen = inStr.length();

        isValid = true; // Assume that the input is valid, unless our check fails
        for (int inCharIdx = 0; inCharIdx < inStrLen; ++inCharIdx) {
            inChar = inStr.charAt(inCharIdx);
            if (!(inChar == '0' || inChar == '1')) {
                isValid = false;
                break; // break the loop upon first error, no need to continue for more errors
                       // If this is not encountered, isValid remains true after the loop.
            }
        }
        System.out.println "\"" + inStr + "\" is " + (isValid ? "" : "NOT ") + "a binary string");
        in.close();
    }
}
```

Enter a binary string: **1011000**
"1011000" is a binary string

Enter a binary string: **10001900**
"10001900" is NOT a binary string

Örnek: Binary Kontrol - Çözüm 2

```
import java.util.Scanner;
/**
 * Check if the input string is a valid binary string.
 */
public class ValidateBinStringV2 {
    public static void main(String[] args) {
        // Declare variables
        String inStr;    // The input string
        int inStrLen;    // The length of the input string
        char inChar;     // Each char of the input string
        Scanner in = new Scanner(System.in);

        // Prompt and read input as "String"
        System.out.print("Enter a binary string: ");
        inStr = in.next();
        inStrLen = inStr.length();

        for (int inCharIdx = 0; inCharIdx < inStrLen; ++inCharIdx) {
            inChar = inStr.charAt(inCharIdx);
            if (!(inChar == '0' || inChar == '1')) {
                System.out.println "\"" + inStr + "\" is NOT a binary string");
                return; // exit the program upon the first error detected
            }
        }
        // for-loop completed. No error detected.
        System.out.println "\"" + inStr + "\" is a binary string");
        in.close();
    }
}
```

Enter a binary string: **1011000**
"1011000" is a binary string

Enter a binary string: **10001900**
"10001900" is NOT a binary string

Örnek: Binary to Decimal

```
import java.util.Scanner;
/**
 * Prompt user for a binary string, and convert into its equivalent decimal number.
 */
public class Bin2Dec {
    public static void main(String[] args) {
        // Declare variables
        String binStr; // The input binary string
        int binStrLen; // The length of binStr
        int dec = 0; // The decimal equivalent, to accumulate from 0
        char binChar; // Each individual char of the binStr
        Scanner in = new Scanner(System.in);

        // Prompt and read input as "String"
        System.out.print("Enter a binary string: ");
        binStr = in.next();
        binStrLen = binStr.length();

        // Process char by char from the right (i.e. Least-significant bit)
        // using exponent as loop index.
        for (int exp = 0; exp < binStrLen ; ++exp) {
            binChar = binStr.charAt(binStrLen - 1 - exp);
            // 3 cases: '1' (add to dec), '0' (valid but do nothing), other (error)
            if (binChar == '1') {
                dec += (int)Math.pow(2, exp); // cast the double result back to int
            } else if (binChar == '0') {
            } else {
                System.out.println("error: invalid binary string \"" + binStr + "\"");
                return; // or System.exit(1);
            }
        }

        // Print result
        System.out.println("The equivalent decimal for \"" + binStr + "\" is " + dec);
        in.close();
    }
}
```

Enter a binary string: **10001001**

The equivalent decimal for "10001001" is 137

binStr	:	1	0	1	1	1	0	0	1
charAt(idx)	:	0	1	2	3	4	5	6	7
Math.pow(2, exp)	:	7	6	5	4	3	2	1	0

(idx increases from the left)
(exp increases from the right)

```
binStr.length() = 8
idx + exp = binStr.length() - 1
```

```
number = in.nextInt(2); // Input in binary e.g., 10110100
System.out.println(number); // 180
```

Örnek: Hexadecimal to Decimal

```
import java.util.Scanner;
/**
 * Prompt user for the hexadecimal string, and convert to its equivalent decimal number
 */
public class Hex2Dec {
    public static void main(String[] args) {
        // Declare variables
        String hexStr; // The input hexadecimal String
        int hexStrLen; // The length of hexStr
        int dec = 0; // The decimal equivalent, to accumulate from 0
        Scanner in = new Scanner(System.in);

        // Prompt and Read input as "String"
        System.out.print("Enter a Hexadecimal string: ");
        hexStr = in.next();
        hexStrLen = hexStr.length();

        // Process char by char from the left (most-significant digit)
        for (int charIdx = 0; charIdx < hexStrLen; ++charIdx) {
            char hexChar = hexStr.charAt(charIdx);
            int expFactor = (int) Math.pow(16, hexStrLen - 1 - charIdx);
            // 23 cases: '0'-'9', 'a'-'f', 'A'-'F', other (error)
            if (hexChar == '0') {
                // Valid but do nothing
            } else if (hexChar >= '1' && hexChar <= '9') {
                dec += (hexChar - '0') * expFactor; // Convert char '0'-'9' to int 0-9
            } else if (hexChar >= 'a' && hexChar <= 'f') {
                dec += (hexChar - 'a' + 10) * expFactor; // Convert char 'a'-'f' to int 10-15
            } else if (hexChar >= 'A' && hexChar <= 'F') {
                dec += (hexChar - 'A' + 10) * expFactor; // Convert char 'A'-'F' to int 10-15
            } else {
                System.out.println("error: invalid hex string \"" + hexStr + "\"");
                return; // or System.exit(1);
            }
        }
        System.out.println("The equivalent decimal for \"" + hexStr + "\" is " + dec);
        in.close();
    }
}
```

Enter a Hexadecimal string: 10aB
The equivalent decimal for "10aB" is 4267

StringBuilder Sınıfı

StringBuilder, **string** (*metinsel*) ifadeleri birleştirme ve birbirine ekleme amaçlı kullanılan bir sınıftır. Normal string komutlarla yapılan işlemlere göre daha hızlı bir performans gösterir.

Kullanım şekli:

```
StringBuilder değişken = new StringBuilder();
```

Temel string değişkenlerin sahip olduğu metot ve özelliklere sahiptir, ek olarak **append**, **insert**, **reverse()**, **capacity()**, **delete()** gibi metotlara sahiptir. **append**, karakter dizisinin en sonuna ekleme yapılırken, **insert** ile karakter dizisinin istenen yerine ekleme yapılabilir. **reverse()** ise karakter dizisini ters çevirir.

StringBuilder Sınıfı

Bazı StringBuilder Metotları

StringBuilder	append(): Karakter dizisinin en sonuna ekleme yapar.
İnt	capacity(): Geçerli kapasite değerini döndürür
StringBuilder	delete(int başlangıç, int bitiş): Başlangıç ve bitiş indisleri belirtilen aralıktaki karakterleri siler.
StringBuilder	deleteCharAt(int indis): İndis numarası verilen karakteri siler.
StringBuilder	insert(int ekleme yeri, String str): Karakter dizisinin istenen yerine ekleme yapılır.
StringBuilder	reverse(): Karakter dizisini ters çevirir.
Void	setCharAt(int indis, char ch): İndis numarası ile belirtilen yere istenen karakteri ekler.
Void	setLength(int yeniBoyut): Karakter dizisinin uzunluğunu artırır veya kısaltır.

Örnek

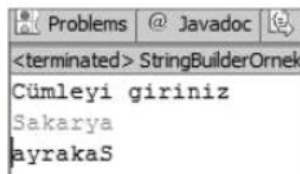
```
StringBuilder a = new StringBuilder("Sen kendini bil");
a.append("Bülent"); // "Sen kendini bil Bülent" sonucunu üretir.
a.insert( 4, "Bülent"); // "Sen Bülent kendini bil" sonucunu üretir
a.reverse(); // "lib inidnek seN" sonucunu üretir
a.setCharAt(3, '+'); // "Sen+kendini bil" sonucunu üretir
a.delete(2,9); // "Seni bil" sonucunu üretir
a.setLength(11); // "Sen kendini" sonucunu üretir
```

Örnek 6.43: Klavyeden girilen metni ters çeviren bir programı yazınız.

Çözüm:

```
import java.util.Scanner;
public class StringBuilderOrnek {
    public static void main(String[] args) {
        Scanner tara=new Scanner (System.in);
        System.out.println("Cümleyi giriniz");
        String cumle=tara.nextLine(); //Klavyeden girilen cümle
        StringBuilder sb = new StringBuilder(cumle);
        sb.reverse(); // cümleyi ters çevir
        System.out.println(sb);
    }
}
```

Programın ekran çıktısı: “Sakarya” girdiğimizde
ayrakas elde edildi.



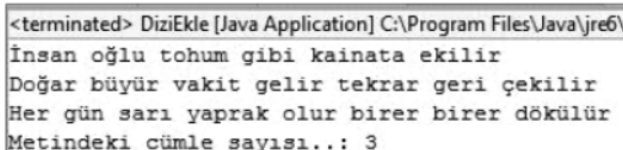
Örnek

Örnek 6.44: Bir metinde geçen cümleleri ve cümle sayısını ekranda gösteren programı yazınız.

Çözüm: Her cümle nokta ile sonlandığına göre, '.' Karakterini esas alarak bir metin içerisindeki cümle sayısını bulabiliriz.

```
public class DiziEkle {  
    public static void main(String args[]) throws Exception {  
        String metin = "İnsan oğlu tohum gibi kâinata ekilir. Doğar büyür vakit ↓  
        gelir tekrar geri çekilir. Her gün sarı yaprak olur birer birer dökülür.";   
        String[] cumle = metin.split("\\.");  
        int i = 0;  
        for (String x : cumle) {  
            i += 1;  
            System.out.println(x);  
        }  
        System.out.println("Metindeki cümle sayısı..: " + i);  
    }  
}
```

Programın ekran çıktısı:



```
<terminated> DiziEkle [Java Application] C:\Program Files\Java\jre6\  
İnsan oğlu tohum gibi kainata ekilir  
Doğar büyür vakit gelir tekrar geri çekilir  
Her gün sarı yaprak olur birer birer dökülür  
Metindeki cümle sayısı..: 3
```

KAYNAKLAR

- ▶ Programming Notes. (2021, March 11). Retrieved from <https://www3.ntu.edu.sg/home/ehchua/programming/index.html>
- ▶ Çobanoğlu B. (2020) Java ile Programlama ve Veri Yapıları. İstanbul Pusula Yayıncılık. 978-605-2359-84-6