

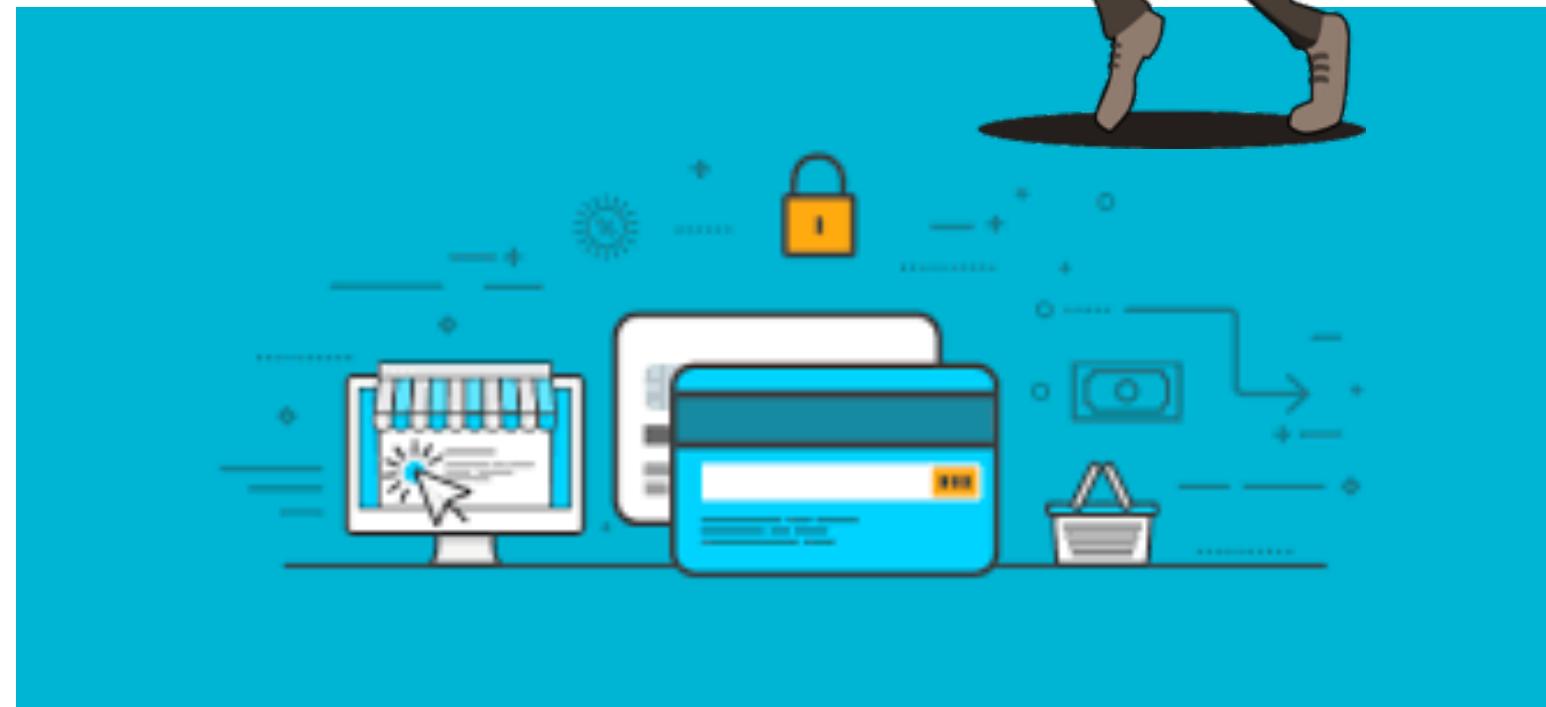
CENG 102

# Nesneye Yönelik Programlama

Elif Haytaoğlu

# Online Alışveriş: Güvenlik (Security)

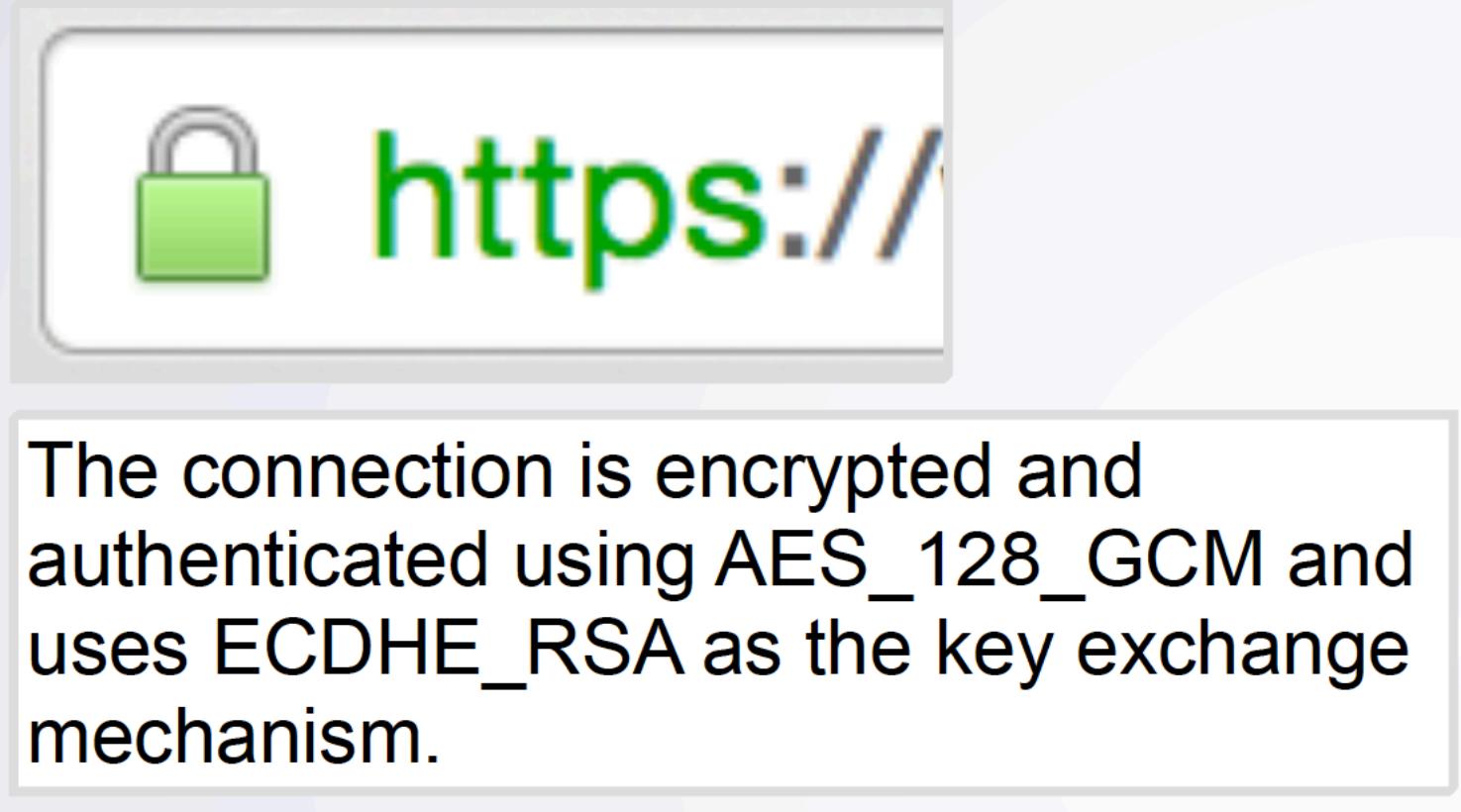
- Bilgisayar Güvenliği hakkında
- Online alışveriş yapmak istiyorsunuz.



# Bilgisayar göndermeden önce şifreliyor



# Modern Kriptografi: https



Güvenli bağlantı: Bu siteye gönderilen bilgileriniz (örneğin şifreler veya kredi kartı numaraları) gizli.

Uses modern cryptography: RSA +AES

# Modern Cryptography: https

| <https://support.apple.com>

https=secure

Uses modern cryptography: RSA +AES

Secure Connection:

Your information (for example, passwords or credit card numbers) is private when it is sent to this site.

~~$n = pq$~~   
 ~~$\gcd(e, (p-1)(q-1)) = 1$~~   
 ~~$c = m^e \pmod n$~~   
 ~~$GF(2^8)$~~

# Ancient History to Modern Times



present

| <https://support.apple.com>

- Modern cryptography: güvenilir, ileri derecede matematik
- Classical cryptography: güvenilir değil, basit matematik

# Ancient History to Modern Times



Egypt

1900 BCE

present

| <https://support.apple.com>

- Modern cryptography: güvenilir, ileri derecede matematik
- Classical cryptography: güvenilir değil, basit matematik

# Ancient History to Modern Times



- Modern cryptography: güvenilir, ileri derecede matematik
- Classical cryptography: güvenilir değil, basit matematik

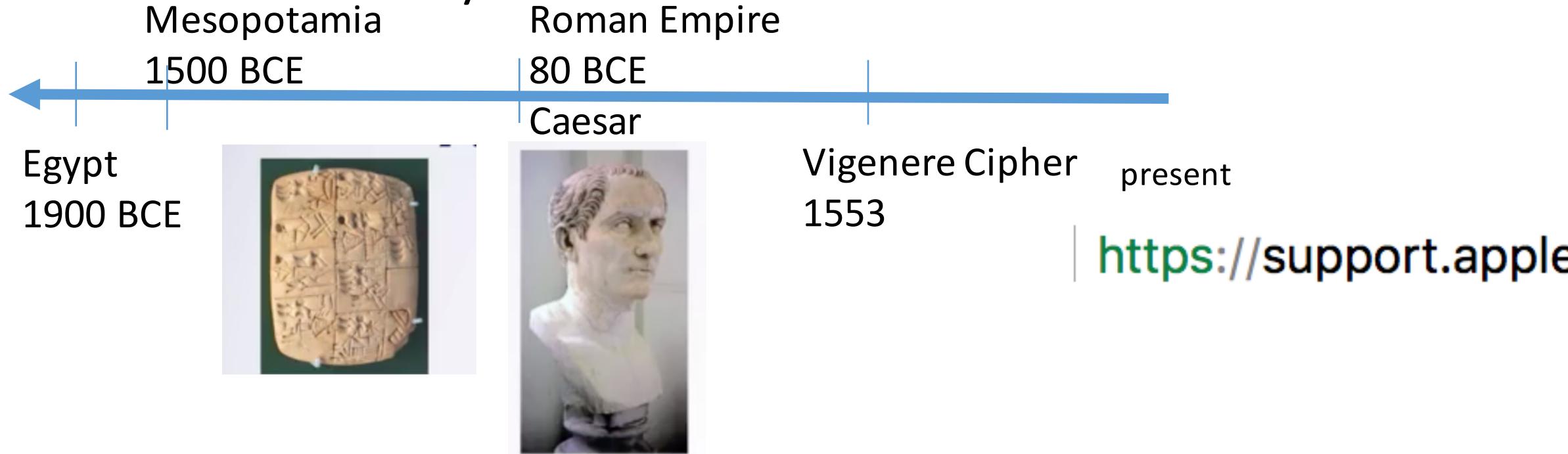
# Ancient History to Modern Times



- Modern cryptography: güvenilir, ileri derecede matematik
- Classical cryptography: güvenilir değil, basit matematik

<https://support.apple.com>

# Ancient History to Modern Times



- Modern cryptography: güvenilir, ileri derecede matematik
- Classical cryptography: güvenilir değil, basit matematik

# Ancient History to Modern Times



# Caesar Cipher (Sezar Şifrelemeyi Geçekleştirmek)

- Tanıtım

# Caesar Cipher (Sezar Şifreleme)



FIRST LEGION ATTACK EAST FLANK

CFOPQ IBDFLK XQQXZH BXPQ CIXKH

# Caesar Cipher (Sezar Şifreleme)



FIRST LEGION ATTACK EAST FLANK

CFOPQ IIBDFLK XQQXZH BXPQ CIXKH

- Named after Julius Caesar
- Encryption: substitute letter with  $(\text{letter} + N)$ 
  - Caesar :  $N=23$  (i.e. 3 letters prior)

# Caesar Cipher (Sezar Şifreleme)



FIRST LEGION ATTACK EAST FLANK

ABCDEFIGHJKLMNOPQRSTUVWXYZ

A blue arrow points from the letter 'F' in the text above to the letter 'I' in the Caesar cipher key below.

- Julius Caesar yüzünden bu şekilde isimlendirilmiştir.
- Şifreleme: harf kydırma şeklinde gerçekleşmektedir. (letter + N)
  - Caesar : N=23 (ya da 3 harf öncesi)

# Caesar Cipher (Sezar Şifreleme)



FIRST LEGION ATTACK EAST FLANK

F

ABCDEFGHIJKLMNOPQRSTUVWXYZ

- Julius Caesar yüzünden bu şekilde isimlendirilmiştir.
- Şifreleme: harf kydırma şeklinde gerçekleşmektedir. (letter + N)
  - Caesar : N=23 (ya da 3 harf öncesi)

# Caesar Cipher (Sezar Şifreleme)



FIRST LEGION ATTACK EAST FLANK

F

ABCDEFGHIJKLMNPQRSTUVWXYZ

C

- Julius Caesar yüzünden bu şekilde isimlendirilmiştir.
- Şifreleme: harf kydırma şeklinde gerçekleşmektedir. (letter + N)
  - Caesar : N=23 (ya da 3 harf öncesi)

# Caesar Cipher (Sezar Şifreleme)



FIRST LEGION ATTACK EAST FLANK

ABCDEFGHIJKLMNOPQRSTUVWXYZ

C

- Julius Caesar yüzünden bu şekilde isimlendirilmiştir.
- Şifreleme: harf kydırma şeklinde gerçekleşmektedir. (letter + N)
  - Caesar : N=23 (ya da 3 harf öncesi)

# Caesar Cipher (Sezar Şifreleme)



FIRST LEGION ATTACK EAST FLANK

|

ABCDEFGHIJKLMNOPQRSTUVWXYZ

C

- Julius Caesar yüzünden bu şekilde isimlendirilmiştir.
- Şifreleme: harf kydırma şeklinde gerçekleşmektedir. (letter + N)
  - Caesar : N=23 (ya da 3 harf öncesi)

# Caesar Cipher (Sezar Şifreleme)



FIRST LEGION ATTACK EAST FLANK

|

ABCDEFGHIJKLMNOPQRSTUVWXYZ

CF

- Julius Caesar yüzünden bu şekilde isimlendirilmiştir.
- Şifreleme: harf kydırma şeklinde gerçekleşmektedir. (letter + N)
  - Caesar : N=23 (ya da 3 harf öncesi)

# Caesar Cipher (Sezar Şifreleme)



FIRST LEGION ATTACK EAST FLANK

ABCDEFGHIJKLMNOPQRSTUVWXYZ

CFOPQ IBDFLK

- Julius Caesar yüzünden bu şekilde isimlendirilmiştir.
- Şifreleme: harf kydırma şeklinde gerçekleşmektedir. (letter + N)
  - Caesar : N=23 (ya da 3 harf öncesi)

# Caesar Cipher (Sezar Şifreleme)



FIRST LEGION **A**TACK EAST FLANK

**A**B C D E F G H I J K L M N O P Q R S T U V W X Y Z

C F O P Q I B D F L K

- Julius Caesar yüzünden bu şekilde isimlendirilmiştir.
- Şifreleme: harf kydırma şeklinde gerçekleşmektedir. (letter + N)
  - Caesar : N=23 (ya da 3 harf öncesi)

# Caesar Cipher (Sezar Şifreleme)



FIRST LEGION ATTACK EAST FLANK

A

A ABCDEFGHIJKLMNOPQRSTUVWXYZ

CFOPQ IBDFLK X

- Julius Caesar yüzünden bu şekilde isimlendirilmiştir.
- Şifreleme: harf kydırma şeklinde gerçekleşmektedir. (letter + N)
  - Caesar : N=23 (ya da 3 harf öncesi)

# Caesar Cipher (Sezar Şifreleme)



FIRST LEGION ATTACK EAST FLANK

ABCDEFGHIJKLMNOPQRSTUVWXYZ

CFOPQ IBDFLK X

- One way: math on letters
  - Everything is a number
  - 'F'-3='C'
  - 'A'-3 ?
    - Need to wrap around

# Caesar Cipher (Sezar Şifreleme)



FIRST LEGION ATTACK EAST FLANK

ABCDEFGHIJKLM NOPQRSTUVWXYZ

ABCDEFGHIJKLM NOPQRSTUVWXYZ

- Another way: pre-shift alphabet
  - Compute shifts of each letter at start

# Caesar Cipher (Sezar Şifreleme)



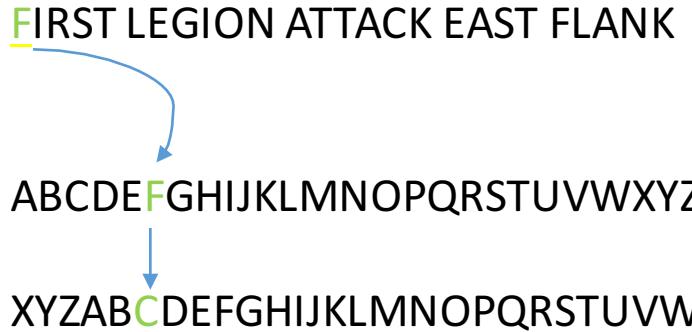
FIRST LEGION ATTACK EAST FLANK

ABCDEFGHIJKLMNOPQRSTUVWXYZ

XYZABCDEFGHIJKLMNPQRSTUVWXYZ

- Another way: pre-shift alphabet
  - Compute shifts of each letter at start
  - Lookup each letter

# Caesar Cipher (Sezar Şifreleme)



- Diğer Yöntem: alfabeliyi önceden kaydır
  - Tüm harflerin karşılığını önceden belirle
  - Herbirine teker teker bak

If you have a String variable called `alphabet = "ABCDEFGHIJKLM NOPQRSTUVWXYZ"`, then which method would find the F in that String?

---

- `alphabet.position("F");`
- `alphabet.find("F");`
- `alphabet.indexOf("F");`
- `alphabet.location("F");`

# Yeni Kavramlar

- Uygulamaya geçmeden önce birkaç yeni kavram
  - String Manipülasyonları
  - Range aralığı kadar dönen for döngüleri

# Strings

```
CGGACACACAAAAAGAAAAAGTTTTAAGATTGGTGTGCAGTAACATGAGGAAGATTAACAG  
TTTCCTCAGTTAAGGTATACTGAAATTGAGATTGAGATTCTCCTCTTGCTATTCTGAACTTCC  
CTGGTTGTGACAATTGAATCAGTTTATCTATTACCAATTACCATCAACATGGTATGTCTAGTGATCTG  
GGACTCTTCTTCATCTGGTTTCTAGAGCTCTGAATCTATTGTGAGAAGTTCATCCAAACGACCCA
```



DNA

```
<div class="split-3-layout layout theme-base">  
<h2 class="section-heading">  
</h2>  
<div class="column">  
<article class="story theme-summary"
```

**HTML**



```
TimeEST, TemperatureF, Dew PointF, Humidity, Sea Level PressureIn, VisibilityMPH,...  
12:51 AM, 30.9, 25.0, 79, 30.36, 10.0, Calm, Calm, -, N/A, , Overcast, 0, 2014-01-01 05:51:00  
1:51 AM, 30.9, 25.0, 79, 30.37, 10.0, Calm, Calm, -, N/A, , Overcast, 0, 2014-01-01 06:51:00  
2:51 AM, 30.9, 26.1, 82, 30.38, 10.0, Calm, Calm, -, N/A, , Overcast, 0, 2014-01-01 07:51:00  
3:51 AM, 30.0, 26.1, 85, 30.37, 10.0, Calm, Calm, -, N/A, , Overcast, 0, 2014-01-01 08:51:00  
4:51 AM, 30.0, 27.0, 88, 30.37, 10.0, Calm, Calm, -, N/A, , Overcast, 0, 2014-01-01 09:51:00  
5:51 AM, 30.9, 26.1, 82, 30.37, 10.0, Calm, Calm, -, N/A, , Overcast, 0, 2014-01-01 10:51:00
```



# Stringler

- Concatenation
  - Stringleri birleştirme
  - + operatörü ile yapılabilir.

```
"XYZ" + "ABCDEFGHIJKLMNPQRSTUVWXYZ"
```



```
"XYZABCDEFGHIJKLMNPQRSTUVWXYZ"
```

# Sezar Şifreleme için

- Önceden Düzenlenmiş bir alfabe oluşturabiliriz
  - substring metodu ile iki farklı string parçası elde edilebilir.
  - İkisi birleştirilebilir.

alphabet: ABCDEFGHIJKLMNOPQRSTUVWXYZ

encr:

# Sezar Şifreleme için

- Önceden Düzenlenmiş bir alfabe oluşturabiliriz
  - substring metodu ile iki farklı string parçası elde edilebilir.
  - İkisi birleştirilebilir.

alphabet: ABCDEFGHIJKLMNOPQRSTUVWXYZ

encri: XYZABCDEFGHIJKLMNOPQRSTUVWXYZ

```
String encri = alphabet.substring(23);
encri = encri + alphabet.substring(0,23);
```

How would you generalize the following code, which creates a String variable named **encr** that is a shifted alphabet for encryption, to be used for any value of **key**?

```
1 String encr = alphabet.substring(23);
2 encr = encr + alphabet.substring(0, 23);
```

1 String encr = alphabet.substring(23);
2 encr = encr + alphabet.substring(0, key);

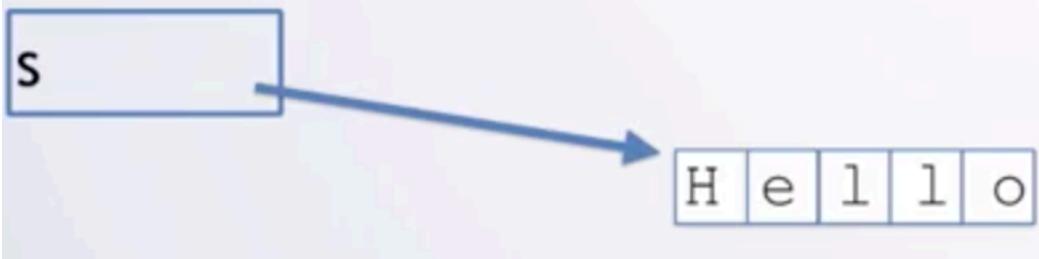
1 String encr = alphabet.substring(key);
2 encr = encr + alphabet.substring(0, key);

1 String encr = alphabet.substring(key);
2 encr = encr + alphabet.substring(0, 23);

# String'ler değiştirilemezdir. (immutable)

- Değiştiremez.
- Yenilerini yapabiliriz.

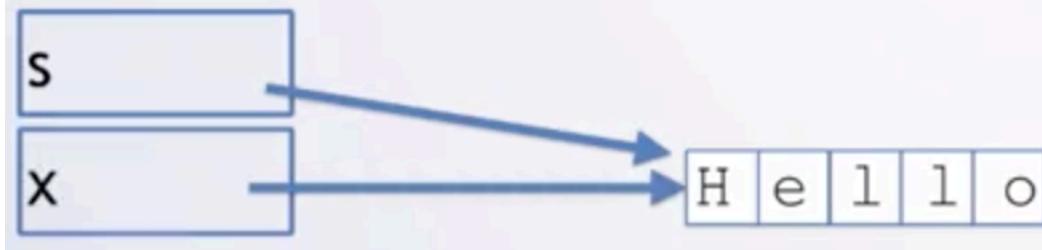
```
String s = "Hello";
```



# String'ler değiştirilemezdir.

- Değiştiremez.
- Yenilerini yapabiliriz.

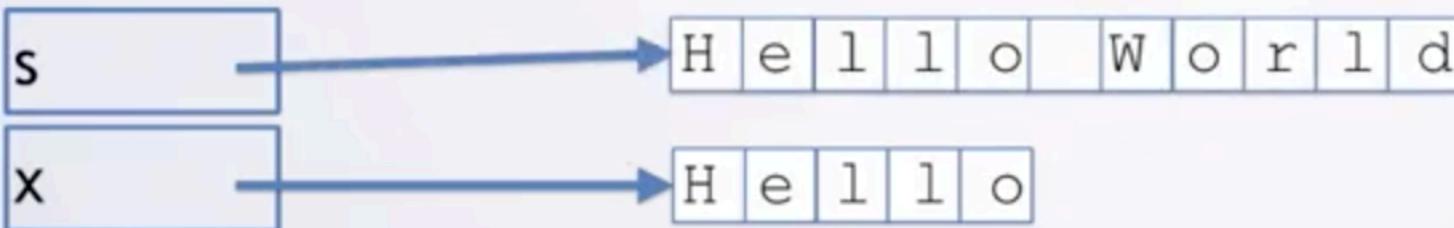
```
String s = "Hello";  
String x = s;
```



# String'ler değiştirilemezdir.

- Değiştiremez.
- Yenilerini yapabiliriz.

```
String s = "Hello";  
String x = s;  
s = s + " World";
```



What are the values of String **s** and **m** after the following code executes?

```
1 String s = "blue";
2 String m = s + "moon";
3 s = s + m;
4 m = "low" + s;
```

- 
- m is lowbluebluemoon and s is bluebluemoon
  - s is bluemoonbluemoon and m is lowbluemoonbluemoon
  - s is bluemoon and m is lowbluemoon
  - s and m are both lowbluebluemoon

```
public class StringConstructors {  
    public static void main(String[] args) {  
        char[] charArray = {'b', 'i', 'r', 't', 'h', ' ', 'd', 'a', 'y'};  
        String s = new String("hello");  
  
        // use String constructors  
        String s1 = new String();  
        String s2 = new String(s);  
        String s3 = new String(charArray);  
        String s4 = new String(charArray, 6, 3);  
  
        System.out.printf(  
            "s1 = %s\ns2 = %s\ns3 = %s\ns4 = %s\n", s1, s2, s3, s4);  
    }  
}
```

```
public class StringMiscellaneous {
    public static void main(String[] args) {
        String s1 = "hello there";
        char[] charArray = new char[5];

        System.out.printf("s1: %s", s1);

        // test length method
        System.out.printf("\nLength of s1: %d", s1.length());

        // loop through characters in s1 with charAt and display reversed
        System.out.printf("\nThe string reversed is: ");

        for (int count = s1.length() - 1; count >= 0; count--) {
            System.out.printf("%c ", s1.charAt(count));
        }

        // copy characters from string into charArray
        s1.getChars(0, 5, charArray, 0);
        System.out.printf("\nThe character array is: ");

        for (char character : charArray) {
            System.out.print(character);
        }

        System.out.println();
    }
}
```

# String Karşılaştırma

- compareTo metodu
- ==
- equals metodu
- (Figure 14.3)

# Karakterlerin String içerisindeki Yerleri

```
public class StringIndexMethods {  
    public static void main(String[] args) {  
        String letters = "abcdefghijklmabcdefghijklm";  
  
        // test indexOf to locate a character in a string  
        System.out.printf(  
            "'c' is located at index %d\n", letters.indexOf('c'));  
        System.out.printf(  
            "'a' is located at index %d\n", letters.indexOf('a', 1));  
        System.out.printf(  
            "'$' is located at index %d\n", letters.indexOf('$'));  
  
        // test lastIndexOf to find a character in a string  
        System.out.printf("Last 'c' is located at index %d\n",  
            letters.lastIndexOf('c'));  
        System.out.printf("Last 'a' is located |at index %d\n",  
            letters.lastIndexOf('a', 25));  
    }  
}
```

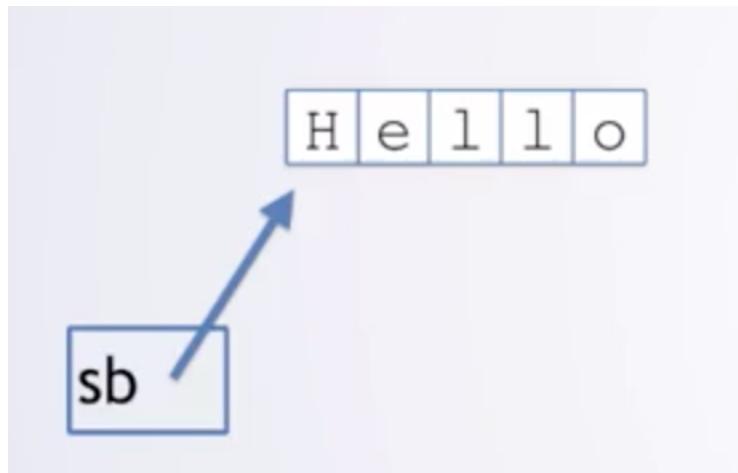
# StringBuilder Sınıfı

- Java'da stringler üzerinde değişiklik yapabilmek için StringBuilder sınıfı
  - Vardır.
  - Değiştirilebilir karakter dizileri oluşturabilirsiniz.

Method Name	Functionality
append	Put String, int, char, etc.. on end
insert	Insert String, int, char, etc... into middle
charAt	Gets character at specified index
setCharAt	Changes the character at specified index
toString	Get back String that you made

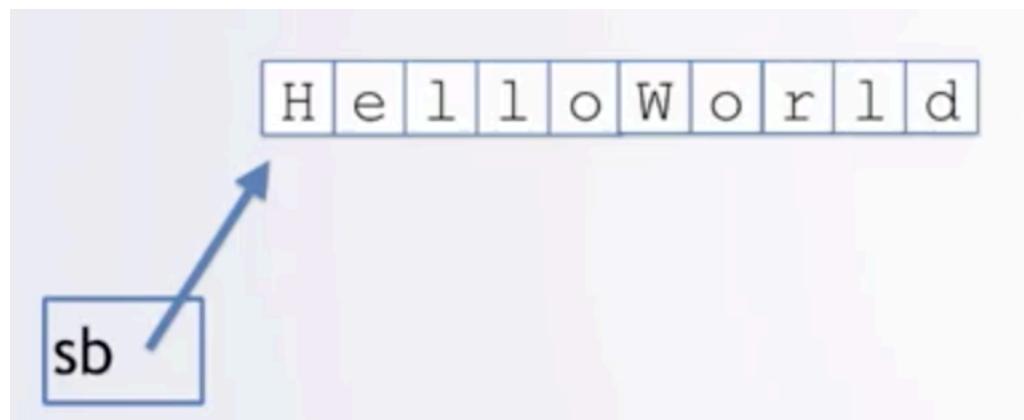
# StringBuilder Sınıfı

- `StringBuilder sb=new StringBuilder("Hello");`



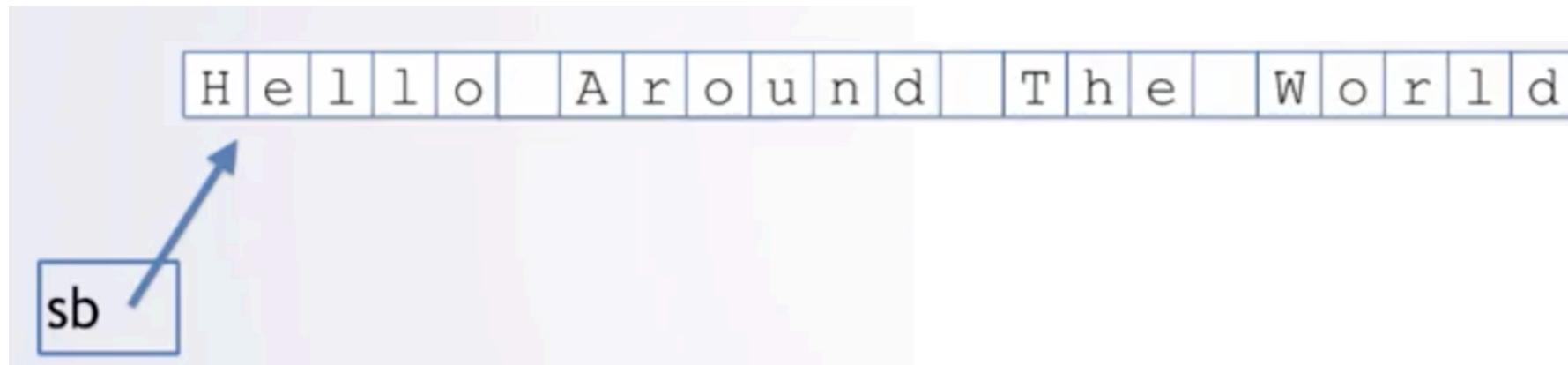
# StringBuilder Sınıfı

- `StringBuilder sb=new StringBuilder("Hello");`
- `sb.append("World");`



# StringBuilder Sınıfı

- `StringBuilder sb=new StringBuilder("Hello");`
  - `sb.append("World");`
  - `sb.insert(5, " Around The ");`



Which of the following options would produce the same printed output as the following code?

```
1 StringBuilder sb = new StringBuilder("start");
2 sb.insert(4,"le");
3 System.out.println(sb);
```

- ```
1 String s = "start";
2 s.substring(0,4) + "le" + s.substring(4);
3 System.out.println(s);
```
- ```
1 String s = "start";
2 s = s.substring(0) + "le" + s.substring(4);
3 System.out.println(s);
```
- ```
1 String s = "start";
2 s = s.substring(0,4) + "le" + s.substring(4);
3 System.out.println(s);
```

# Loops, Indexes Kavramlarını Gözden Geçirme

- Farklı problem türleri için farklı for döngüleri kullanabiliriz.
- Bir String içerisindeki kolamaları ya da tagları bulmak için
  - while (true) {... break ...}
  - FileResource nesnesi kullanarak dosya satırlarını kullanırken  
`for(String s :fr.lines()){..}`

Karakter(lerin) indexisini bulurken  
“cgatga”.indexOf(“atg”);  
“cgatga”.substring(1,4);

For the following assignment to the variable **dna** of type String:

```
1 dna = "cgatga";
```

What is the result of these two method calls?

```
1 dna.indexOf("atg");
2 dna.substring(1,4);
```

- 3 and "gat"
- 2 and "gatg"
- 3 and "cgat"
- 3 and "gatg"
- 2 and "gat"

# String içerisindeki karakterlere erişmek

- “CGATTA” stringinin tersi (reverse’si) “ATTAGC”’dir.
  - Biyoinformatik alanında kullanışlı
- Palindromlar
  - Eh, ça va, la vache?
  - Draw, O Caesar erase a coward

# String indexleri

- Üç aşamalı döngüler
  - ; ile ayrılır
  - İllkendirme Bölümü (Initialization (happens once, before guard))
  - Kontrol Bölümü (Guard evaluated before loop body)
  - Değişim Bölümü.

```
public String reverse(String s){  
    String ret = "";  
    for(int k=0; k < s.length(); k += 1){  
        ret = s.charAt(k) + ret;  
    }  
    return ret;  
}
```

```
public String reverse(String s){  
    String ret = "";  
    for(int k=0; k < s.length(); k += 1){  
        ret = s.charAt(k) + ret;  
    }  
    return ret;  
}
```

```
public String reverse(String s){  
    String ret = "";  
    int k=0;  
    while (k < s.length()){  
        ret = s.charAt(k) + ret;  
        k += 1;  
    }  
    return ret;  
}
```

# Anatomy of a for loop

Consider the call reverse “pit”



```
public String reverse(String s){  
    String ret = "";  
    for(int k=0; k < s.length(); k += 1){  
        ret = s.charAt(k) + ret;  
    }  
    return ret;  
}
```

```
1 public String reverse(String s){  
2     String ret = "";  
3     for(int k=0; k < s.length(); k +=1){  
4         ret = s.charAt(k) + ret;  
5     }  
6     return ret;  
7 }
```

In the for loop, if **k+=1** is changed to **k+=2**, then what is the result of the following call?

```
1 System.out.println(reverse("computer"));
```

# Character Sınıfı

- char ilkel (primitive) bir tiptir, tek tırnak içerisinde gösterilmektedir.
  - 'a', '1', ... But using "a" is a String
- The Character sınıfının farklı metodları bulunmaktadır.
  - Örn: Character.toLowerCase('G')

| Method Name     | Functionality                           |
|-----------------|-----------------------------------------|
| isLowerCase(ch) | returns boolean if ch is 'a', 'b' ...   |
| isDigit(ch)     | returns boolean if ch is '0','1',...'9' |
| toLowerCase(ch) | returns lowercase version of ch         |
| toUpperCase(ch) | returns uppercase version of ch         |

```
public class CharacterDemo {  
    public void digitTest() {  
        String test = "ABCabc0123456789';#!";  
        for(int k=0; k < test.length(); k++){  
            char ch = test.charAt(k);  
            if (Character.isDigit(ch)){  
                System.out.println(ch+ " is a digit");  
            }  
            if (Character.isAlphabetic(ch)){  
                System.out.println(ch+ " is alphabetic");  
            }  
        }  
    }  
}
```

```
public void conversionTest(){
    String test = "ABCDEFabcdef123!#";
    for(int k=0; k < test.length(); k++){
        char ch = test.charAt(k);
        char uch = Character.toUpperCase(ch);
        char lch = Character.toLowerCase(ch);
        System.out.println(ch+ " "+uch+ " "+lch);
    }
}
```

# Caesar Cipher

```
public String encrypt(String input, int key) {
    //Make a StringBuilder with message (encrypted)
    StringBuilder encrypted = new StringBuilder(input);
    //Write down the alphabet
    String alphabet = "ABCDEFGHIJKLMNOPQRSTUVWXYZ";
    //Compute the shifted alphabet
    String shiftedAlphabet = alphabet.substring(key)+alphabet.substring(0,key);
    //Count from 0 to < length of encrypted, (call it i)
    for(int i = 0; i < encrypted.length(); i++) {
        //Look at the ith character of encrypted (call it currChar)
        char currChar = encrypted.charAt(i);
        //Find the index of currChar in the alphabet (call it idx)
        int idx = alphabet.indexOf(currChar);
        //If currChar is in the alphabet
        if(idx != -1){
            //Get the idxth character of shiftedAlphabet (newChar)
            char newChar = shiftedAlphabet.charAt(idx);
            //Replace the ith character of encrypted with newChar
            encrypted.setCharAt(i, newChar);
        }
        //Otherwise: do nothing
    }
    //Your answer is the String inside of encrypted
    return encrypted.toString();
}
```

# Test

```
public void testCaesar() {
    int key = 17;
    FileResource fr = new FileResource();
    String message = fr.asString();
    String encrypted = encrypt(message, key);
    System.out.println(encrypted);
    String decrypted = encrypt(encrypted, 26-key);
    System.out.println(decrypted);
}
```

# Caesar Şifrelemeyi Kırmak

- Caesar cipher’I Java’da kodladık.
  - Şifreleme için basit bir yöntem.
  - Şifrelemek için anahtar kullandık, peki nasıl şifreyi çözeceğiz?
  - Karşı taraf anahtarı biliyor olmalı
    - Encrypt with 7, decyprt with 19
- Nasıl Kırabiliriz?
  - Brute force( kaba kuvvet) ile mi?

# What is the Encrypted Message?

**Lujyfwapvu huk zljbypaf hyl mbukhtluahs whyaz vm avkhf'z Pualyula**

- 0 Lujyfwapvu huk zljbypaf hyl mbukhtluahs whyaz vm avkhf'z Pualyula.
- 1 Mvkzgxbqvv ivl amkcqbqg izm ncvliumvbit xizba wn bwlig'a Qvbmzvmb.
- 2 Nwlahycrxw jwm bnldarch jan odwmjvnwcju yjacb xo cxmjhb'c Rwcnavnc.
- 3 Oxmbizdsyx kxn comebsdi kbo pexnkwoxdkv zkbdc yp dynki'c Sxdoxbod.
- 4 Pyncjaetzy lyo dpnfctej lcp qfyolxpyelw alced zq ezolj'd Tyepcype.
- 5 Qzodkbfuaz mzp eqogdufk mdq rgzpmqzfmx bmdfe ar fapmk'e Uzfqdzqf.
- 6 Rapelcgvba naq frphevgl ner shaqnzragny cnegf bs gbqlf'f Vagrearg.
- 7 Sbqfmdhwcb obr gsqifwhm ofs tibroasbhoz dofhg ct hcrom'g Wbhsfbsh.
- 8 Tcrgneixdc pcs htrjgxin pgt ujcspbtcipa epgih du idspn'h Xcitgcti.
- 9 Udshofjyed qdt iuskhijo qhu vkdtqcudjqb fqhji ev jetqo'i Ydjuhduj.
- 10 Vetipgkzfe reu jvtlizkp riv wleurdvekrc grikj fw kfurp'j Zekviewk.
- 11 Wfujqhlagf sfv kwumjalq sjw xmfvsewflsd hsjlk gx lgvsq'k Aflwjfwl.
- 12 Xgvkrimbhg tgw lxvnkbmr tkx yngwtxgmte itkml hy mhwtr'l Bgmxkgxm.
- 13 Yhwlsjncih uhx mywolcns uly zohxugyhnu julnm iz nixus'm Chnylhyn.
- 14 Zixmtkodji viy nzxpmdot vmz apiyvhziovg kvmon ja ojyvt'n Diozmizo.
- 15 AjynulpekJ wjz oayqnepu wna bqqzwiajpwh lwnpo kb pkzwu'o Ejpanjap.
- 16 Bkzovmqflk xka pbzrofqv xob crkaxjbkqxi mxoqp lc qlaxv'p Fkqbokbq.
- 17 Clapwnrgml ylb qcaspgrw ypc dslbykclryj nyprq md rmbyw'q Glrcplcr.
- 18 Dmbqxoshnm zmc rdbtqhsx zqd etmczldmszk ozqsr ne snczx'r Hmsdqmds.
- 19 Encryption and security are fundamental parts of today's Internet.
- 20 Fodszzqujpo boe tfdvsjuz bsf gvoebnfoubm qbsut pg upebz't Joufsofu.
- 21 Gpetarvkqp cpf ugewtkva ctg hwpfcogpvcn rctvu qh vqfca'u Kpvgtpgv.
- 22 Hqfubswlrq dqq vhfxulwb duh ixqgdphqwdo sduvv ri wrgdb'v Lqwhuqhw.
- 23 Irgvctxmsr erh wigyvmxc evi jyrheqirxep tevxw sj xshec'w Mrxivrix.
- 24 Jshwdyunts fsi xjhzwnyd fwj kzsifrjsyfq ufwyx tk ytifd'x Nsyjwsjy.
- 25 Ktixevzout gtj ykiaxoze gxk latjgsktzgr vgxzy ul zujge'y Otzkxtkz.



# Object Oriented Caesar Cipher ?

```
1 public class CaesarCipher {
2     public String encrypt(String input, int key) {
3         String alphabet = "ABCDEFGHIJKLMNOPQRSTUVWXYZ";
4         String shiftedAlphabet = alphabet.substring(key) +
5                         alphabet.substring(0, key);
6         StringBuilder sb = new StringBuilder(input);
7         for (int i = 0; i <sb.length(); i++) {
8             char c = sb.charAt(i);
9             int idx = alphabet.indexOf(c);
10            }
11        }
12    }
```