古典密码学：算法不公开，密钥不公开
现代密码学：算法公开，密钥不公开或公开一个密钥

3.1.4 Simple Cryptography with Strings and Character Arrays

One of the primary applications of arrays is the representation of strings of characters. That is, string objects are usually stored internally as an array of characters. Even if strings may be represented in some other way, there is a natural relationship between strings and character arrays—both use indices to refer to their characters. Because of this relationship, Java makes it easy for us to create string objects from character arrays and vice versa. Specifically, to create an object of class String from a character array A, we simply use the expression,

new String(A)

that is, one of the constructors for the String class takes a character array as its argument and returns a string having the same characters in the same order as the array. For example, the string we would construct from the array $A = [\texttt{a}, \texttt{c}, \texttt{a}, \texttt{t}]$ is `acat`. Likewise, given a string S, we can create a character array representation of S by using the expression,

S.toCharArray()

that is, the String class has a method, toCharArray, which returns an array (of type **char**[]) with the same characters as S. For example, if we call toCharArray on the string `adog`, we would get the array $B =$ [a, d, o, g].

The Caesar Cipher

凯撒密码

One area where being able to switch from string to character array and back again is useful is in ***cryptography***, the science of secret messages and their applications. This field studies ways of performing ***encryption***, which takes a message, called the ***plaintext***, and converts it into a scrambled message, called the ***ciphertext***. Likewise, cryptography also studies corresponding ways of performing ***decryption***, which takes a ciphertext and turns it back into its original plaintext.

Arguably the earliest encryption scheme is the ***Caesar cipher***, which is named after Julius Caesar, who used this scheme to protect important military messages. (All of Caesar's messages were written in Latin, of course, which already makes them unreadable for most of us!) The Caesar cipher is a simple way to obscure a message written in a language that forms words with an alphabet.

The Caesar cipher involves replacing each letter in a message with the letter that is three letters after it in the alphabet for that language. So, in an English message, we would replace each A with D, each B with E, each C with F, and so on. We continue this approach all the way up to W, which is replaced with Z. Then, we let the substitution pattern ***wrap around***, so that we replace X with A, Y with B, and Z with C.

Using Characters as Array Indices

If we were to number our letters like array indices, so that A is 0, B is 1, C is 2, and so on, then we can write the Caesar cipher as a simple formula:

Replace each letter i with the letter $(i+3)$ mod 26,

where mod is the ***modulus*** operator, which returns the remainder after performing an integer division. This operator is denoted % in Java, and it is exactly the operator we need to easily perform the wrap around at the end of the alphabet. For 26 mod 26 is 0, 27 mod 26 is 1, and 28 mod 26 is 2. The decryption algorithm for the Caesar cipher is just the opposite—we replace each letter with the one three places before it, with wrap around for A, B, and C.

We can capture this replacement rule using arrays for encryption and decryption. Since every character in Java is actually stored as a number—its Unicode value—we can use letters as array indices. For an uppercase character c, for example, we can use c as an array index by taking the Unicode value for c and subtracting A. Of course, this only works for uppercase letters, so we will require our secret messages to be uppercase. We can then use an array, encrypt, that represents the encryption replacement rule, so that encrypt$[i]$ is the letter that replaces letter number i (which is $c -$ A for an uppercase character c in Unicode). This usage is illustrated in Figure 3.6. Likewise, an array, decrypt, can represent the decryption replacement rule, so that decrypt$[i]$ is the letter that replaces letter number i.

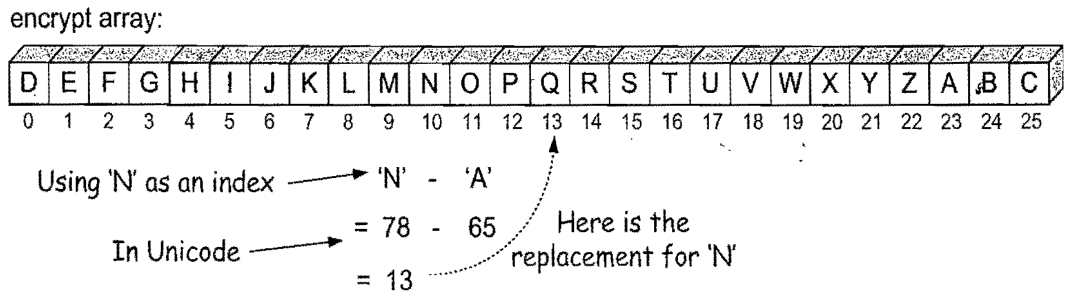

Figure 3.6: Illustrating the use of uppercase characters as array indices, in this case to perform the replacement rule for Caesar cipher encryption.

In Code Fragment 3.9, we give a simple, complete Java class for performing the Caesar cipher, which uses the approach above and also makes use of conversions between strings and character arrays. When we run this program (to perform a simple test), we get the following output:

```
Encryption order = DEFGHIJKLMNOPQRSTUVWXYZABC
Decryption order = XYZABCDEFGHIJKLMNOPQRSTUVW
WKH HDJOH LV LQ SODB; PHHW DW MRH'V.
THE EAGLE IS IN PLAY; MEET AT JOE'S.
```

```
/** Class for doing encryption and decryption using the Caesar Cipher. */
public class Caesar {
  public static final int ALPHASIZE = 26; // English alphabet (uppercase only)
  public static final char[] alpha = {'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'};
  protected char[] encrypt = new char[ALPHASIZE]; // Encryption array
  protected char[] decrypt = new char[ALPHASIZE]; // Decryption array
  /** Constructor that initializes the encryption and decryption arrays */
  public Caesar() {
    for (int i=0; i<ALPHASIZE; i++)
      encrypt[i] = alpha[(i + 3) % ALPHASIZE]; // rotate alphabet by 3 places
    for (int i=0; i<ALPHASIZE; i++)
      decrypt[encrypt[i] - 'A'] = alpha[i]; // decrypt is reverse of encrypt
  }
  /** Encryption method */
  public String encrypt(String secret) {
    char[] mess = secret.toCharArray();    // the message array
    for (int i=0; i<mess.length; i++)      // encryption loop
      if (Character.isUpperCase(mess[i]))  // we have a letter to change
        mess[i] = encrypt[mess[i] - 'A'];  // use letter as an index
    return new String(mess);
  }
  /** Decryption method */
  public String decrypt(String secret) {
    char[] mess = secret.toCharArray();    // the message array
    for (int i=0; i<mess.length; i++)      // decryption loop
      if (Character.isUpperCase(mess[i]))  // we have a letter to change
        mess[i] = decrypt[mess[i] - 'A'];  // use letter as an index
    return new String(mess);
  }
  /** Simple main method for testing the Caesar cipher */
  public static void main(String[] args) {
    Caesar cipher = new Caesar();          // Create a Caesar cipher object
    System.out.println("Encryption order = " + new String(cipher.encrypt));
    System.out.println("Decryption order = " + new String(cipher.decrypt));
    String secret = "THE EAGLE IS IN PLAY; MEET AT JOE'S.";
    secret = cipher.encrypt(secret);
    System.out.println(secret);            // the ciphertext
    secret = cipher.decrypt(secret);
    System.out.println(secret);            // should be plaintext again
  }
}
```

Code Fragment 3.9: A simple, complete Java class for the Caesar cipher.