Unicode

https://home.unicode.org/

Unicode is an international standard of character encoding which has the capability of representing a majority of written languages all over the globe. Unicode uses hexadecimal to represent a character. Unicode is a 16-bit character encoding system. The lowest value is \u0000 and the highest value is \uFFFF.

UTF-8 is a variable width character encoding. UTF-8 has the ability to be as condensed as ASCII but can also contain any Unicode characters with some increase in the size of the file. UTF stands for Unicode Transformation Format. The '8' signifies that it allocates 8-bit blocks to denote a character. The number of blocks needed to represent a character varies from 1 to 4.

UTF

The term UTF stands for " Unicode Transformation Format ". The Unicode Transformation Format is a structure to encode characters into Unicode. There exist multiple versions of the UTF, from which the " UTF-8 " version stands out in the most prolific way. The UTF-8 is a variable-length encoder and uses 8-bit code units while encoding. The UTF-8 version is designed for backwards compatibility with ASCII encoding. The number of blocks used to represent a character varies from one to four in the Unicode Transformation Format. The different UTF encodings that are used include,

- UTF-1: The first of the Unicode Transformation Formats. It is no longer a part of the Unicode standard.
- UTF-7: Uses 7-bits for the encoding process. It is the format which is primarily used in the mailing software "email".
- UTF-8: It is the most used format in the present times. The UTF-8 uses 8-bits to encode with variable width.
- UTF-16: Uses the 16-bit variable-width encoding format.
- UTF-32: Uses 32-bits for encoding, but the width is fixed, i.e. not variable width.
- UTF-EBCIDC: This format uses only 8-bits for encoding. It is designed to be compatible with the Extended Binary Coded Decimal Interchange Code (EBCDIC).

To convert a Unicode to UTF-8 in the Java Programming Language, we make use of a method called " getBytes() ". The getBytes() method will encode a string into a sequence of bytes to return a byte array as output.

Unicode is a computing industry standard designed to consistently and uniquely encode characters used in written languages throughout the world. The Unicode standard uses hexadecimal to express a character. For example, the value 0x0041 represents the Latin character A. The Unicode standard was initially designed using 16 bits to encode characters because the primary machines were 16-bit PCs.

When the specification for the Java language was created, the Unicode standard was accepted and the `char` primitive was defined as a 16-bit data type, with characters in the hexadecimal range from 0x0000 to 0xFFFF.

Because 16-bit encoding supports 2^{16} (65,536) characters, which is insufficient to define all characters in use throughout the world, the Unicode standard was extended to 0x10FFFF, which supports over one million characters. The definition of a character in the Java programming language could not be changed from 16 bits to 32 bits without causing millions of Java applications to no longer run properly. To correct the definition, a scheme was developed to handle characters that could not be encoded in 16 bits.

The characters with values that are outside of the 16-bit range, and within the range from 0x10000 to 0x10FFFF, are called *supplementary characters* and are defined as a pair of `char` values.

This lesson includes the following sections:

- Terminology – Code points and other terms are explained.
- Supplementary Characters as Surrogates – 16-bit surrogates are used to implement supplementary characters, which cannot be implemented as a single primitive `char` data type.
- Character and String API – A listing of related API for the `Character`, `String`, and related classes.
- Sample Usage – Several useful code snippets are provided.
- Design Considerations – Design considerations to keep in mind to ensure that your application will work with any language script.
- More Information – A list of further resources are provided.

```
String str1 = "\u0048\u0065\u006C\u006C\u006F";

String str2 = "Hello";

System.out.println(str1.equals(str2)); // true

System.out.println(str1.length());     // 5
```