9/1/20:

1. How memory is created in strings?
2. What is the usage of having matches() and intern()?
3. What is the purpose of replaceAll() method?
4. What is charset(), utf8(), utf16()?

Answers:

1. To create the string objects you need not to use ‘**new**‘ keyword. Where as to create other type of objects you have to use ‘new’ keyword. This attention is worth the while, because the strings are used almost everywhere while developing any kind of applications. While storing the string objects in the memory also, they are specially treated by the Java.

We all know that JVM divides the allocated memory to a Java program into two parts. one is **Stack** and another one is **heap**. Stack is used for execution purpose and heap is used for storage purpose. In that heap memory, JVM allocates some memory specially meant for string literals. This part of the heap memory is called **String Constant Pool**.

public class StringExamples

{

    public static void main(String[] args)

    {

        //Creating string objects using literals

        String s1 = "abc";

        String s2 = "abc";

        System.out.println(s1 == s2);        //Output : true

        //Creating string objects using new operator

        String s3 = new String("abc");

        String s4 = new String("abc");

        System.out.println(s3 == s4);        //Output : false

    }

}

3. We can replace all occurrence of a single character, or a substring of a given String in Java using the **replaceAll()** method of java.lang.String class. The replaceAll() method replaces each substring of this string (the String on which it is called) that matches the given regular expression with the given replacement. It internally uses classes like Pattern and Matcher from java.util.regex package for searching and replacing matching characters or substring.

Example:

String replaceSample = "String replace Example with regular expression";

String newString = replaceSample.replaceAll("^S","R");

**Output:** Rtring replace Example with regular expression

1. **Charset** this class defines methods for creating decoders and encoders and for retrieving the various names associated with a charset. Instances of this class are immutable.

This class also defines static methods for testing whether a particular charset is supported, for locating charset instances by name, and for constructing a map that contains every charset for which support is available in the current Java virtual machine.

**UTF**-**8** is a compromise character encoding that can be as compact as ASCII but can also contain any unicode characters. **UTF** stands for Unicode Transformation Format. The '**8**' means it **uses 8**-bit blocks to represent a character.

UTF stands for Unicode Transformation Format. The '8' means it uses 8-bit blocks to represent a character. The number of blocks needed to represent a character varies from 1 to 4.

**UTF-16** (16-[bit](https://en.wikipedia.org/wiki/Bit) [Unicode](https://en.wikipedia.org/wiki/Unicode) Transformation Format) is a [character encoding](https://en.wikipedia.org/wiki/Character_encoding) capable of encoding all 1,112,064 valid [code points](https://en.wikipedia.org/wiki/Code_point) of Unicode. The encoding is [variable-length](https://en.wikipedia.org/wiki/Variable-width_encoding), as code points are encoded with one or two 16-bit *code units*.

UTF-16 arose from an earlier fixed-width 16-bit encoding known as [UCS-2](https://en.wikipedia.org/wiki/UCS-2) once it became clear that more than 2 code points were needed.