Java String,StringBuffer and StringBuilder and Examples

-K.L.MADHAVI

Java Strings

- A **String** is an object that represents a sequence of characters.It acts the same as an array of characters in Java.The java.lang.String class is used to create a string object.
- Java provides a robust and flexible API for handling strings, allowing for various operations such as **concatenation**, **comparison**, and **manipulation**.
- **Example:**

```
String name = “Madhu";
```

- There are 2 ways to create a String:

1.By String Literal: Java String literal is created by using double quotes. For Example:

String s="welcome";

2.By new keyword: To make Java more memory efficient we use this.(because no new objects are created if it exists already in the string constant pool).

String s=**new** String("Welcome");//creates two objects and one refe rence variable

Java String Example

```
public class StringExample{
public static void main(String args[]){
String s1="java";//creating string by Java string literal
char ch[]={'s','t','r’,’i','n','g','s'};
String s2=new String(ch);//converting char array to string
String s3=new String("example");//creating Java string by new keyword
System.out.println(s1);
System.out.println(s2);
System.out.println(s3);
}}
```

Some disadvantages of String class

- String class objects are immutable.(can not be modified on same location).

  String Str= new String(“hello”);

  here “Str” cant be modified on same location.

- Memory is wasted(Requires more memory).

To overcome this problem we use StringBuffer Class.

StringBuffer class

Java StringBuffer class is used to create mutable (modifiable) String objects.It is the same as String class except it is mutable i.e. it can be changed.

Example:

```
StringBuffer Str=new StringBuffer(“welcome”)
System.out.println(Str);
```

Important Constructors of StringBuffer class

- **StringBuffer()**: creates an empty string buffer with an initial capacity of 16.
- **StringBuffer(String str)**: creates a string buffer with the specified string.
- **StringBuffer(int capacity)**: creates an empty string buffer with the specified capacity as length.

Continue…

There are several advantages of using StringBuffer over regular String objects in Java:

1. **Mutable:** StringBuffer objects are mutable, which means that you can modify the contents of the object after it has been created. In contrast, String objects are immutable, which means that you cannot change the contents of a String once it has been created.
2. **Efficient**: Because StringBuffer objects are mutable, they are more efficient than creating new String objects each time you need to modify a string. This is especially true if you need to modify a string multiple times, as each modification to a String object creates a new object and discards the old one.

Overall, if you need to perform multiple modifications to a string, using StringBuffer can be more efficient than regular String objects.

StringBuffer is a peer class of **String** that provides much of the functionality of strings. The string represents fixed-length, immutable character sequences while StringBuffer represents growable and writable character sequences.

Here is an example of using StringBuffer to append strings:

```
public class StringBufferExample
{
     public static void main(String[] args)
      {
             StringBuffer sb = new StringBuffer();
              sb.append("Hello");
              sb.append(" ");
              sb.append("world");
              System.out.println(sb); //Hello world
        }
}
```

StringBuilder class

StringBuilder in Java represents a mutable sequence of characters. Since the String Class in Java creates an immutable sequence of characters, the StringBuilder class provides an alternative to String Class, as it creates a mutable sequence of characters. The function of StringBuilder is very much similar to the StringBuffer class, as both of them provide an alternative to String Class by making a mutable sequence of characters.

However, the StringBuilder class differs from the StringBuffer class on the basis of synchronization. The StringBuilder class provides no guarantee of synchronization whereas the StringBuffer class does. String Builder is not thread-safe and high in performance compared to String buffer.

Constructors in Java StringBuilder Class

- **StringBuilder():** Constructs a string builder with no characters in it and an initial capacity of 16 characters.
- **StringBuilder(int capacity):** Constructs a string builder with no characters in it and an initial capacity specified by the capacity argument.
- **StringBuilder(CharSequence seq):** Constructs a string builder that contains the same characters as the specified CharSequence.
- **StringBuilder(String str):** Constructs a string builder initialized to the contents of the specified string.

Here is an example of using StringBuilder to append strings:

```
public class StringBuilderExample
{
    public static void main(String[] args)
     {
           StringBuilder sb = new StringBuilder();
           sb.append("Hello");
           sb.append(" ");
           sb.append("world");
           System.out.println(sb); //Hello world
      }
}
```