JAVA STRING

In general **string is a sequence of characters**. String is an object that represents a sequenceof characters. The **java.lang.String class is used to create string object**. In java, string is basically an object that represents sequence of char values. An array of characters works same asjava string. For example:

Java String class provides a lot of methods to perform operations on string such as com- pare(), concat(), equals(), split(), length(), replace(), compareTo(), intern(), substring() etc.

The java.lang.String class implements *Serializable*, *Comparable* and *CharSequence* in-terfaces. The CharSequence interface is used to represent sequence of characters. It is imple-mented by String, StringBuffer and StringBuilder classes. It means can create string in java by using these 3 classes.

The string objects can be created using two ways.

1. By String literal
2. By new Keyword

String Literal

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

1. String s=”welcome”;

Each time you create a string literal, the JVM checks the string constant pool first. If the string already exists in the pool, a reference to the pooled instance is returned. If string doesn’t exist in the pool, a new string instance is created and placed in the pool. For example:

String s1=”Welcome”;

String s2=”Welcome”;

In the above example only one object will be created. Firstly JVM will not find any string object with the value “Welcome” in string constant pool, so it will create a new object. Afterthat it will find the string with the value “Welcome” in the pool, it will not create new object but will return the reference to the same instance. To make Java more memory efficient (be-cause no new objects are created if it exists already in string constant pool).

2. By new keyword

String s=new String(“Welcome”);

In such case, JVM will create a new string object in normal (non pool) heap memory and the literal “Welcome” will be placed in the string constant pool. The variable s will refer to the object in heap (non pool).

The java String is immutable i.e. it cannot be changed. Whenever we change any string, a new instance is created. For mutable string, you can use StringBuffer and StringBuilder classes.

The following program explains the creation of strings

```
public class String_Example{
public static void main(String args[])
{
String s1="java";
char c[]={'s','t','r','i','n','g'};String s2=new String(c);
String s3=new String("example");
System.out.println(s1);
System.out.println(s2);
System.out.println(s3);
}}
```

Java String class methods

The java.lang.String class provides many useful methods to perform operations on sequence of char values.

Method	Description
char charAt(int index)	returns char value for the particular index
int length()	returns string length
static String format(String format,Object... args)	returns formatted string
static String format(Locale l, Stringformat, Object... args)	returns formatted string with given locale
String substring(int beginIndex)	returns substring for given begin index
String substring(int beginIndex, intendIndex)	returns substring for given begin index andend index
boolean contains(CharSequence s)	returns true or false after matching the sequence of char value
static String join(CharSequence delim- iter, CharSequence... elements)	returns a joined string
boolean equals(Object another)	checks the equality of string with object
boolean isEmpty()	checks if string is empty
String concat(String str)	concatinates specified string
String replace(char old, char new)	replaces all occurrences of specified char value

String replace(CharSequence old, Char- Sequence new)	replaces all occurrences of specified CharSe- quence
static String equalsIgnoreCase(String another)	compares another string. It doesn't check case.
String[] split(String regex)	returns splitted string matching regex
String[] split(String regex, int limit)	returns splitted string matching regex and limit
String intern()	returns interned string
int indexOf(int ch)	returns specified char value index
int indexOf(int ch, int fromIndex)	returns specified char value index starting with given index
int indexOf(String substring)	returns specified substring index
int indexOf(String substring, int fro- mIndex)	returns specified substring index starting with given index
String toLowerCase()	returns string in lowercase.
String toLowerCase(Locale l)	returns string in lowercase using specified locale.
String toUpperCase()	returns string in uppercase.
String toUpperCase(Locale l)	returns string in uppercase using specified locale.
String trim()	removes beginning and ending spaces of this string.
static String valueOf(int value)	converts given type into string. It is over-loaded

The following program is an example for String concat function:

```
class string_method{
 public static void main(String args[]){
   String s="Java";
   s=s.concat(" Programming");
   System.out.println(s);
 }
}
```

Output:

Java Programming