BCA – 401: Java Programming

Rahul Kumar Singh

In today's Class we have discussed on Strings in Java.

String in Java:-

Generally, String is a sequence of characters. But in Java, string is an object that represents a sequence of characters. The java.lang.String class is used to create a string object.

string object in Java:-

There are two ways to create String object:-

By string literal

By new keyword

1) String Literal

Java String literal is created by using double quotes.

For Example:

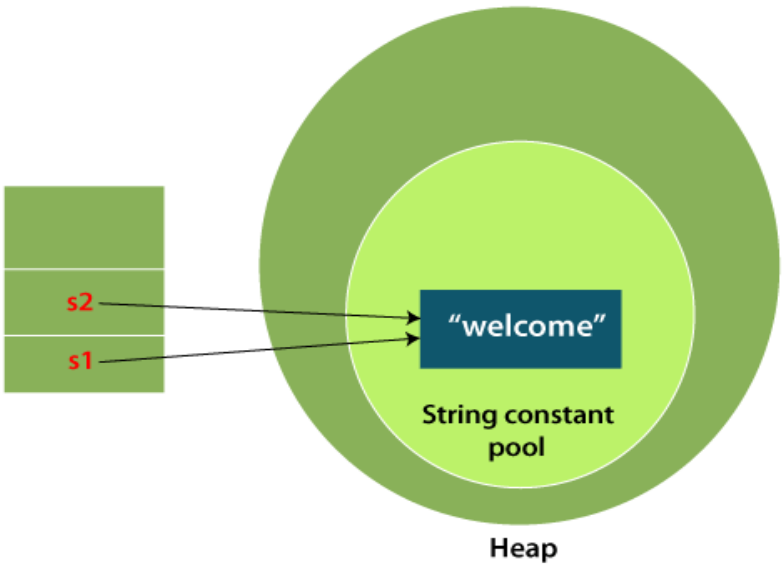
String s="welcome";

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 the 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"; //It doesn't create a new instance



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 that is why it will create a new object. After that it will find the string with the value "Welcome" in the pool, it will not create a new object but will return the reference to the same instance.

Why Java uses the concept of String literal?

To make Java more memory efficient (because no new objects are created if it exists already in the string constant pool).

2) By new keyword

String s1=new String("Welcome");

String s2=new String("Welcome");

//creates two objects and one reference variable

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 a heap (non-pool).

Java String Example:-

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

Output:-

java

strings

example

The above code, converts a char array into a String object. And displays the String objects s1, s2, and s3 on console using println() method.

Java String class methods:-

The java.lang.String class provides many useful methods to perform operations on sequence of char values. Which are as below:-

1. Java String class charAt() method- The Java String class charAt() method returns a char value at the given index number.

The index number starts from 0 and goes to n-1, where n is the length of the string. It returns StringIndexOutOfBoundsException, if the given index number is greater than or equal to this string length or a negative number.

Syntax:-

public char charAt(int index)

Example:-

public class CharAtExample

```
{
public static void main(String args[])
{
String name="hellojava";
char ch=name.charAt(4);
//returns the char value at the 4th index
System.out.println(ch);
}
}
```

Output:-

```
o
```

2. Java String class length() method-

The Java String class length() method finds the length of a string. The length of the Java string is the same as the Unicode code units of the string. It returns length of characters. In other words, the total number of characters present in the string.

Syntax:-

```
public int length()
```

Example:-

```
public class LengthExample
{
public static void main(String args[])
{
String s1="hellojava";
String s2="python";
System.out.println("string length is: "+s1.length());
//9 is the length of javatpoint string
System.out.println("string length is: "+s2.length());
//6 is the length of python string
}
}
```

3. java string format() method:-

The java string format() method returns the formatted string by given locale, format and arguments.

If you don't specify the locale in String.format() method, it uses default locale by calling Locale.getDefault() method.

The format() method of java language is like sprintf() function in c language and printf() method of java language.

There are two type of string format() method:

public static String format(String format, Object... args)

and,

public static String format(Locale locale, String format, Object... args)

Parameters:-

locale : specifies the locale to be applied on the format() method.

format : format of the string.

args : arguments for the format string. It may be zero or more.

Java String Format Specifiers:-

Here, we are providing a table of format specifiers supported by the Java String.

Format Specifier	Data Type	Output
%a	floating point (except BigDecimal)	Returns Hex output of floating point number.
%b	Any type	"true" if non-null, "false" if null
%c	character	Unicode character
%d	integer (incl. byte, short, int, long,	Decimal Integer

	bigint)	
%e	floating point n	decimal number in scientific notatio
%f	floating point	decimal number
%g	floating point	decimal number, possibly in scientific notation depending on the precision and value.
%h	any type	Hex String of value from hashCode() method.
%n	none	Platform-specific line separator.
%o	integer (incl. byte, short, int, long, bigint)	Octal number
%s	any type	String value
%t	Date/Time (incl. long, Calendar, Date and TemporalAccessor)	%t is the prefix for Date/Time conversions. More formatting flags are needed after

		this. See Date/Time conversion below.
%x	integer	(incl. byte, short, int, long, bigint) Hex string.

Example:-

```
public class FormatExample
{
    public static void main(String[] args)
     {
        // Integer value
        String str1 = String.format("%d", 101);
        // String value
        String str2 = String.format("%s", "Amar Singh");
        // Float value
        String str3 = String.format("%f", 101.00);
         // Hexadecimal value
        String str4 = String.format("%x", 101);
         // Char value
        String str5 = String.format("%c", 'c');
```

```
        System.out.println(str1);
        System.out.println(str2);
        System.out.println(str3);
        System.out.println(str4);
        System.out.println(str5);
    }
}
```

Output:-

```
101
Amar Singh
101.000000
65
c
```

4. Java String class substring() method- The Java String class substring() method returns a part of the string.

We pass beginIndex and endIndex number position in the Java substring method where beginIndex is inclusive, and endIndex is exclusive. In other words, the beginIndex starts from 0, whereas the endIndex starts from 1.

There are two types of substring methods in Java string.

```
public String substring(int startIndex)  // type - 1
```

and

public String substring(int startIndex, int endIndex) // type - 2

If we don't specify endIndex, the method will return all the characters from startIndex.

Parameters

startIndex : starting index is inclusive

endIndex : ending index is exclusive

Returns

specified string

Example:-

```
public class SubstringExample
{
public static void main(String args[])
{
String s1="JavaScript";
System.out.println(s1.substring(2,4));  //returns va
System.out.println(s1.substring(2));   //returns vaScript
}
}
```

Output:-

va

vaScript

5. Java String class contains() method:-

The Java String class contains() method searches the sequence of characters in this string. It returns true if the sequence of char values is found in this string otherwise returns false.

Syntax:-

public boolean contains(CharSequence sequence)

Parameter:-

sequence : specifies the sequence of characters to be searched.

Returns:-

It returns true if the sequence of char value exists, otherwise false.

Example:-

class ContainsExample

{

public static void main(String args[])

```
{
String name="what do you know about me";
System.out.println(name.contains("do you know"));
System.out.println(name.contains("about"));
System.out.println(name.contains("hello"));
}
}
```

Output:-

```
true
true
false
```

6. Java String class join() method:-

The Java String class join() method returns a string joined with a given delimiter. In the String join() method, the delimiter is copied for each element. The join() method is included in the Java string since JDK 1.8.

There are two types of join() methods in the Java String class.

Syntax:-

```
public static String join(CharSequence delimiter, CharSequence... elements)
```

and

public static String join(CharSequence delimiter, Iterable<? extends CharSequence> elements)

Parameters:-

delimiter : char value to be added with each element.

elements : char value to be attached with delimiter.

Returns:-

It returns joined string with delimiter.

Example:-

```
public class StringJoinExample
{
public static void main(String args[])
{
String joinString1=String.join("-","welcome","to","java");
System.out.println(joinString1);
}
}
```

Output:-

```
welcome-to-java
```

Example-2:-

```
public class StringJoinExample2
{
   public static void main(String[] args)
    {
      String date = String.join("/","25","06","2020");
      System.out.print(date);
      String time = String.join(":", "12","10","10");
      System.out.println(" "+time);
   }
}
```

Output:-

```
25/06/2020 12:10:10
```

7. Java String equals() method:-

The Java String class equals() method compares the two given strings based on the content of the string. If any character is not matched, it returns false. If all characters are matched, it returns true.

The String equals() method overrides the equals() method of the Object class.

Syntax:-

public boolean equals(Object anotherObject)

Parameter:-

anotherObject : another object, i.e., compared with this string.

Returns:-

It returns true if characters of both strings are equal otherwise false.

Example:-

```
public class EqualsExample
{
public static void main(String args[])
{
String s1="java";
String s2="java";
String s3="JavaScript";
String s4="python";
System.out.println(s1.equals(s2));
//true because content and case is same
System.out.println(s1.equals(s3));
//false because case is not same
System.out.println(s1.equals(s4));
```

```
//false because content is not same
}
}
```

Output:-

```
 true
false
false
```

Another example:-

```
public class EqualsExample2
{
    public static void main(String[] args)
    {
        String s1 = "java";
        String s2 = "java";
        String s3 = "Java";
        System.out.println(s1.equals(s2));
        // True because content is same
        if (s1.equals(s3))
        {
```

```
            System.out.println("both strings are equal");
        }
        else System.out.println("both strings are unequal");
    }
}
```

Output:-

true

both strings are unequal

8. Java String class isEmpty() method:-

The Java String class isEmpty() method checks if the input string is empty or not. Note that here empty means the number of characters contained in a string is zero.

Syntax:-

public boolean isEmpty()

Returns:-

It returns true if length is 0 otherwise false.

Example:-

```
public class IsEmptyExample
{
public static void main(String args[])
```

```
{
String s1="";
String s2="java";
System.out.println(s1.isEmpty());
System.out.println(s2.isEmpty());
}
}
```

Output:-

```
true
false
```

9. Java String class concat() method:-

The Java String class concat() method combines specified string at the end of this string. It returns a combined string. It is like appending another string.

Syntax:-

```
public String concat(String anotherString)
```

Parameter:-

anotherString : another string i.e., to be combined at the end of this string.

Returns:-

It returns combined string

Example:-

```
public class ConcatExample
{
public static void main(String args[])
{
String s1="java string";
// The string s1 does not get changed, even though it is invoking the method concat(), as it is immutable. Therefore, the explicit assignment is required here.
s1.concat("is immutable");
System.out.println(s1);
s1=s1.concat(" is immutable so assign it explicitly");
System.out.println(s1);
}
}
```

Output:-

```
java string
java string is immutable so assign it explicitly
```

10. Java String class replace() method:-

The Java String class replace() method returns a string replacing all the old char or CharSequence to new char or CharSequence.

Since JDK 1.5, a new replace() method is introduced that allows us to replace a sequence of char values.

There are two types of replace() methods in Java String class.

public String replace(char oldChar, char newChar)

and

public String replace(CharSequence target, CharSequence replacement) // this method is added since JDK 1.5.

Parameters:-

oldChar : old character

newChar : new character

target : target sequence of characters

replacement : replacement sequence of characters

Returns:-

It returns replaced string.

Example:-Java String replace(char old, char new) method.

public class ReplaceExample1

```
{
public static void main(String args[])
{
String s1="java is a very good language";
String replaceString=s1.replace('a','e');
//replaces all occurrences of 'a' to 'e'
System.out.println(replaceString);
}
}
```

Output:-

```
jeve is e very good lenguege
```

Example:- Java String replace(CharSequence target, CharSequence replacement) method

```
public class ReplaceExample2
{
public static void main(String args[])
{
String s1="my name is khan my name is java";
String replaceString=s1.replace("is","was");
//replaces all occurrences of "is" to "was"
```

```
System.out.println(replaceString);
}
}
```

Output:-

```
my name was khan my name was java
```

11. String class equalsIgnoreCase() method:-

The Java String class equalsIgnoreCase() method compares the two given strings on the basis of the content of the string irrespective of the case (lower and upper) of the string. It is just like the equals() method but doesn't check the case sensitivity. If any character is not matched, it returns false, else returns true.

Syntax:-

```
publicboolean equalsIgnoreCase(String str)
```

Parameter:-

str : another string i.e., compared with this string.

Returns:-

It returns true if characters of both strings are equal, ignoring case otherwise false.

Example:-

```
public class EqualsIgnoreCaseExample
```

```
{
public static void main(String args[])
{
String s1="java";
String s2="java";
String s3="JAVA";
String s4="python";
System.out.println(s1.equalsIgnoreCase(s2));
//true because content and case both are same
System.out.println(s1.equalsIgnoreCase(s3));
//true because case is ignored
System.out.println(s1.equalsIgnoreCase(s4));
//false because content is not same
}
}
```

Output:-

```
true
true
false
```

12. java string split() method:-

The java string split() method splits this string against given regular expression and returns a char array.

There are two types of split() method in java string.

public String split(String regex)

and,

public String split(String regex, int limit)

Parameter:-

regex : regular expression to be applied on string.

limit : limit for the number of strings in array. If it is zero, it will returns all the strings matching regex.

Returns:-

It returns array of strings.

Example:- Java String split() method

```
public class SplitExample
{
public static void main(String args[])
{
String s1="java string split method";
//splits the string based on whitespace
```

```
String[] words=s1.split("\\s");
//using java foreach loop to print elements of string array
for(String w:words)
{
System.out.println(w);
}
}
}
```

Output:-

```
java
string
split
method
```

Example:- Java String split() method with regex and length

```
public class SplitExample2
{
public static void main(String args[])
{
String s1="welcome to split world";
```

```
System.out.println("returning words:");
for(String w:s1.split("\\s",0))
{
System.out.println(w);
}
System.out.println("returning words:");
for(String w:s1.split("\\s",1))
{
System.out.println(w);
}
System.out.println("returning words:");
for(String w:s1.split("\\s",2))
{
System.out.println(w);
}
}
}
```

Output:-

```
returning words:
welcome
```

to

split

world

returning words:

welcome to split world

returning words:

welcome

to split world

Following String class methods are also used in Java:-

1) **char charAt(int index) -** It returns char value for the particular index.
2) **int length() -** It returns string length.
3) **static String format(String format, Object... args) -** It returns a formatted string.
4) **static String format(Locale l, String format, Object... args)-** It returns formatted string with given locale.
5) **String substring(int beginIndex)-** It returns substring for given begin index.
6) **String substring(int beginIndex, int endIndex)-** It returns substring for given begin index and end index.
7) **boolean contains(CharSequence s)** - It returns true or

false after matching the sequence of char value.

8) **static String join(CharSequence delimiter, CharSequence... elements)-** It returns a joined string.

9) **static String join(CharSequence delimiter, Iterable<? extends CharSequence> elements)-** It returns a joined string.

10) **boolean equals(Object another) -** It checks the equality of string with the given object.

11) **boolean isEmpty()-** It checks if string is empty.

12) **String concat(String str)-** It concatenates the specified string.

13) **String replace(char old, char new)-** It replaces all occurrences of the specified char value.

14) **String replace(CharSequence old, CharSequence new)-** It replaces all occurrences of the specified CharSequence.

15) **static String equalsIgnoreCase(String another)-** It compares another string. It doesn't check case.

16) **String[] split(String regex)-** It returns a split string matching regex.

17) **String[] split(String regex, int limit)-** It returns a split string matching regex and limit.

18) **String intern()-** It returns an interned string.

19) **int indexOf(int ch)-** It returns the specified char

value index.

20) **int indexOf(int ch, int fromIndex)-** It returns the specified char value index starting with given index.

21) **int indexOf(String substring)-** It returns the specified substring index.

22) **int indexOf(String substring, int fromIndex)-** It returns the specified substring index starting with given index.

23) String toLowerCase()- It returns a string in lowercase.

24) **String toLowerCase(Locale l)-** It returns a string in lowercase using specified locale.

25) String toUpperCase()- It returns a string in uppercase.

26) String toUpperCase(Locale l)- It returns a string in uppercase using specified locale.

27) **String trim()-** It removes beginning and ending spaces of this string.

28) **static String valueOf(int value)-** It converts given type into string. It is an overloaded method.