9. STRING HANDLING

Strings, which are widely used in Java programming, are a sequence of characters.The Java platform provides the String class to create and manipulate strings.

1. String Property :

String objects are fixed. Every time we create a string, a new object is created. Working on immutable string is efficient than that of on mutable strings**.**

Mutable classes: StringBuffer and **StringBuilder**

2. Creating Strings :

Two approaches to create the strings in java.

Approach 1: using new keyword

Approach 2: using literals

1. Creating String objects using new keywords:

As with any other object, you can create String objects by using the new keyword and a constructor. These type of String objects will get created dynamically and loaded into heap memory.

There are several ways to create strings using first approach

1. Creating empty string: We can use following contructor of String class
   String()
   Example: String s=new String();
2. Converting character array to string: We can use following constructor
   String(char c[])
   Example: char charray[]={'C' , 'o' , 'd' , 'e' , 'r'};
   String s=new String(chararray);
3. Creating string by passing the character sequence
   Example: String str = new String(“Hello World”);
4. Creating String by passing another string: We can use following contructor
   String(String s)
   Example: String s1=new String(str);

2. Creating String objects using string literals:

We can create the string object by writing the character sequence in double quotes (“ ”). Whenever compiler encounters a string literal in your code, it creates a String object with its value in constant pool. This constant pool is also known as literal pool. Thus literals will statically get loaded into literal pool.

The way to create literals is:

String str = "Hello world";

The difference between both approaches is that

1. Literals will get loaded statically in literal pool, but objects with new keyword will get loaded dynamically in heap memory.
2. If two objects with new keyword contains same character sequence, there will be two

different independant objects having different memory location. But if two literals contains same character sequence then there will one copy of literal in literal pool but two references will be there. Thus they are pointing to same memory location.

Example:
If String s=new String(“Hello”);
String s1=new String(“Hello”);
then s==s1 will be false.
But if String s2=“Coder”;
String s3=“Coder”;
then s2==s3 will be true
If we are comparing two objects using ==, it is known as reference comparison. It will compare the memory locations of both objects.

3. String Methods:

Here is the list of methods supported by String class:

SN	Methods with Description
1	char charAt(int index) Returns the character at the specified index.
2	int compareTo(Object o) Compares this String to another Object.
3	int compareTo(String anotherString) Compares two strings lexicographically.
4	int compareToIgnoreCase(String str) Compares two strings lexicographically, ignoring case differences.
5	String concat(String str) Concatenates the specified string to the end of this string.
6	boolean contentEquals(StringBuffer sb) Returns true if and only if this String represents the same sequence of characters as the specified StringBuffer.
7	static String copyValueOf(char[] data) Returns a String that represents the character sequence in the array specified.
8	static String copyValueOf(char[] data, int offset, int count) Returns a String that represents the character sequence in the array specified.
9	boolean endsWith(String suffix) Tests if this string ends with the specified suffix.

10	boolean equals(Object anObject) Compares this string to the specified object.
11	boolean equalsIgnoreCase(String anotherString) Compares this String to another String, ignoring case considerations.
13	byte[] getBytes(String charsetName) Encodes this String into a sequence of bytes using the named charset, storing the result into a new byte array.
14	void getChars(int srcBegin, int srcEnd, char[] dst, int dstBegin) Copies characters from this string into the destination character array.
15	int hashCode() Returns a hash code for this string.
16	int indexOf(int ch) Returns the index within this string of the first occurrence of the specified character.
17	int indexOf(int ch, int fromIndex) Returns the index within this string of the first occurrence of the specified character, starting the search at the specified index.
18	int indexOf(String str) Returns the index within this string of the first occurrence of the specified substring.
19	int indexOf(String str, int fromIndex) Returns the index within this string of the first occurrence of the specified substring, starting at the specified index
20	String intern() Returns a canonical representation for the string object.
21	int lastIndexOf(int ch) Returns the index within this string of the last occurrence of the specified character.
22	int lastIndexOf(int ch, int fromIndex) Returns the index within this string of the last occurrence of the specified character, searching backward starting at the specified index.
23	int lastIndexOf(String str) Returns the index within this string of the rightmost occurrence of the specified substring.
24	int lastIndexOf(String str, int fromIndex) Returns the index within this string of the last occurrence of the specified substring, searching

	backward starting at the specified index.
25	int length() Returns the length of this string.
26	boolean matches(String regex) Tells whether or not this string matches the given regular expression.
27	boolean regionMatches(boolean ignoreCase, int toffset, String other, int ooffset, int len) Tests if two string regions are equal.
28	boolean regionMatches(int toffset, String other, int ooffset, int len) Tests if two string regions are equal
29	String replace(char oldChar, char newChar) Returns a new string resulting from replacing all occurrences of oldChar in this string with newChar.
30	String replaceAll(String regex, String replacement Replaces each substring of this string that matches the given regular expression with the given replacement.
31	String replaceFirst(String regex, String replacement) Replaces the first substring of this string that matches the given regular expression with the given replacement.
32	String[] split(String regex) Splits this string around matches of the given regular expression.
33	String[] split(String regex, int limit) Splits this string around matches of the given regular expression.
34	boolean startsWith(String prefix) Tests if this string starts with the specified prefix.
35	boolean startsWith(String prefix, int toffset) Tests if this string starts with the specified prefix beginning a specified index.
36	CharSequence subSequence(int beginIndex, int endIndex) Returns a new character sequence that is a subsequence of this sequence.
37	String substring(int beginIndex) Returns a new string that is a substring of this string.
38	String substring(int beginIndex, int endIndex) Returns a new string that is a substring of this string.
39	char[] toCharArray()

	Converts this string to a new character array.
40	String toLowerCase() Converts all of the characters in this String to lower case using the rules of the default locale.
41	String toLowerCase(Locale locale) Converts all of the characters in this String to lower case using the rules of the given Locale.
42	String toString() This object (which is already a string!) is itself returned.
43	String toUpperCase() Converts all of the characters in this String to upper case using the rules of the default locale.
44	String toUpperCase(Locale locale) Converts all of the characters in this String to upper case using the rules of the given Locale.
45	String trim() Returns a copy of the string, with leading and trailing whitespace omitted.
46	static String valueOf(primitive data type x) Returns the string representation of the passed data type argument.

String Length:

Methods used to obtain information about an object are known as accessor methods. One accessor method that you can use with strings is the length() method, which returns the number of characters contained in the string object.

Example:

```
public class StringDemo {
  public static void main(String args[]) {
    String palindrome = "We saw it was Tod";
    int len = palindrome.length();
    System.out.println( "String Length is : " + len );
  }
}
```

Output:

String Length is : 17

Concatenating Strings:

The String class includes a method for concatenating two strings:
String str=string1.concat(string2);
Also we can use + operator to conact the strings.

e.g. String s1 = “I saw ”+string1;

Example:

```
public class StringDemo {
        public static void main(String args[]) {
          String string1 = "Technologies ";
              String concat1  = "Coder ".concat(string1);
              String concatstring = concat1.concat("Vashi");
          System.out.println(concatstring);
        }
      }
```

Output:

Coder Technologies Vashi

Comparing Strings:

The two string contents can be compared by using following methods

- **boolean** equals(String s)
- **boolean** equalsIgnoreCase(String s)
- **int** compareTo(String s)
- **int** compareToIgnoreCase(String s)

First two methods returns boolean and next two methods returns integer. compareTo and compareToIgnoreCase methods are used when you want to sort the string in ascending or descending order. The methods equalsIgnoreCase and compareToIgnoreCase ignore the case while comparing two strings. The other two methods are case-sensitive.

Example:

```
public class StringComparison {

      public static void main(String[] args) {

            String s="Coder";
            String s1="Coder";
            String s2="coder";
            String s3="Coper";

            /*---------- equals and equalsIgnoreCase methods--------*/

            if(s.equals(s1))
                  System.out.println("s and s1 are equal");
            else
                  System.out.println("s and s1 are not equal");
```

```
                if(s.equalsIgnoreCase(s2))
                        System.out.println("s and s2 are equal");
                else
                        System.out.println("s and s2 are not equal");

                if(s.equals(s3))
                        System.out.println("s and s3 are equal");
                else
                        System.out.println("s and s3 are not equal");

                /*---------- compareTo and compareToIgnoreCase methods--------*/

                if(s.compareTo(s3)==0)
                        System.out.println("s and s3 are equal");
                else if(s.compareTo(s3)<0)
                        System.out.println("s is smaller than s3");
                else if(s.compareTo(s3)>0)
                        System.out.println("s is greater than s3");
        }
}
```

Output:

s and s1 are equal
s and s2 are equal
s and s3 are not equal
s is smaller than s3

The above methods will compare the contents between two strings not the memory location of string objects.

4. StringBuffer class:

StringBuffer is growable and writeable character sequence. It is mutable.

4.1 StringBuffer Constructors:

- StringBuffer( )
- StringBuffer(int size)
- StringBuffer(String str)
- StringBuffer(CharSequence chars)

4.2 Methods:

It has several methods that can be used to operated on character sequence. Some of the methods are

- length(), capacity(),
- ensureCapasity(), setLength()
- chartAt(), setCharAt(),
- getChars(),
- append(),
- insert(),
- reverse(),
- delete(),
- deleteCharAt(),
- replace(),
- substring()

Example:

```
public class StringBufferDemo {

        public static void main(String[] args) {
                StringBuffer stringBuffer = new StringBuffer("Coder Technologies, Vashi");
                System.out.println("Length: "+stringBuffer.length());
                System.out.println("Capacity: "+stringBuffer.capacity());

            String str = new StringBuffer().append("Coder").append("Technologies").toString();
            System.out.println("String str: "+str);

            stringBuffer.replace(0, 5, "Squad");
            System.out.println("After replacing : "+ stringBuffer);

                                StringBuffer    delstr=stringBuffer.delete(stringBuffer.indexOf("Vashi"),
stringBuffer.length());
            System.out.println("After Deleting: "+delstr);
            System.out.println("Reverse: "+stringBuffer.reverse());
        }
}
```

Output:

```
Length: 25
Capacity: 41
String str: CoderTechnologies
After replacing : Squad Technologies, Vashi
After Deleting: Squad Technologies,
Reverse:  ,seigolonhceT dauqS
```

5. StringBuilder class:

It is identical to StringBuffer but difference is that it is not synchronized. That is it is not thread safe.
Advantages: faster performance

6. Use of intern Method :

intern method returns a canonical representation for the string object. This method is supposed to return the String from the String pool if the String is found in String pool, otherwise a new string object will be added in String pool and the reference of this String is returned.

Syntax:
public String intern()
s.intern() == t.intern() is true if and only if **s.equals(t)** is true.
All literal strings and string-valued constant expressions are interned.

Example:
String str1=”CoderTechnologies”;
String str2=new String(“CoderTechnologies”);
String str3=str2.intern();

(str1==str2) : will be false
(str1==str3) : will be true

More examples on strings are covered in **BasicPrograms** section.

ASSIGNMENTS

1. How many objects will get created in following code?
   String s1=“Welcome”
   String s1=“Welcome”
   String s1=“Welcome”

2. How many objects will get created in following code?
   String s1=new String(“Welcome”)
   String s2=new String(“Welcome”)

3. Reverse the following String Without using String API?
   String s=”Coder Technologies”

4. Sort the following String characters without using String API?
   String s=”Coder Technologies”

5. Sort the String with using String API?
   String s=”Coder Technologies”

6. Check whether the given string is palindrome or not?
   String s="DAD";

7. Reverse the words of following string using String API.
   String s="Coder Technologies vashi"
   Answer should be s1="vashi Technologies Coder"

8. Reverse the words of following string without using String API.
   String s="Coder Technologies vashi"
   Answer should be s1="vashi Technologies Coder"

9. Split the following string into number of words.
   String s="Squad Infotech Nerul"

10. Find out the vowels in following string. Do not repeat the same vowel.
    String s="India is my country. I love my country"
    Answer should be: I a o u e

11. Find out the rhyme words from following string
    String s="I got the job Then I saw the pot in window He love me a lot. Forget me not"
    Answer should be: got pot lot not

12. Replace the string of number by digit in following string.
    String s="There were two balls in bucket. I picked up one ball and then I put three balls"
    Answer should be: "There were 2 balls in bucket. I picked up 1 ball and then I put 3 balls"

13. Check whether the given string is female name or male name (Assume general trend of having male and female names).

14. String str1="CoderTechnologies";
    String str2=new String("CoderTechnologies");
    String str3=str2.intern();
    What will be the answer of following comparison?
    1. str1==str2
    2. str1==str3