String: Generally, String is a sequence of characters. But in Java, String is an object that represents a sequence of characters.

OR

An array of characters works same as Java String.

Example:

char[] ch = {‘d’,’h’,’r’,’u’,’v’};

String s = new String(ch);

Same as:

String s = “dhruv”;

Example1

```java
package com.dhruv;

public class StringEx {
    public static void main(String[] args) {
        char[] ch = {'d','h','r','u','v'};
        String s = new String(ch);
        /*
        is same as:
        String s = "dhruv";
         */
        System.out.println(s);
    }
}
```

Example2

```java
package com.dhruv;

public class StringEx2 {
    public static void main(String[] args)
        String s = "dhruv";
        System.out.println(s);
        System.out.println(s.charAt(0));
        System.out.println(s.charAt(1));
        System.out.println(s.charAt(2));
        System.out.println(s.charAt(3));
        System.out.println(s.charAt(4));
    }
}
```

Note:

1. String is a collection/group of characters.
2. String are also based on indexing which start from 0 and ends with size - 1.
3. In Java, String is a class which is a part of java.lang.String package.
4. String is an immutable(can't be change their value) class.

There are 2 ways to create an object of String class:

1. **By String Literal:**

String a = “dhruv”

String b = “dhruv”

2. **By New Keyword:**

String a = new String(“dhruvlenka”);

String b = new String(“dhruvlenka”);

Methods of String Class:

1. char charAt()
2. int length()
3. String concat()
4. int indexOf()
5. int lastIndexOf()
6. int compareTo()
7. String replace()
8. String substring()
9. String[] split()
10. String toUpperCase()
11. String toLowerCase()
12. String trim()
13. boolean equals()
14. boolean equalsIgnoreCase()
15. String valueOf()

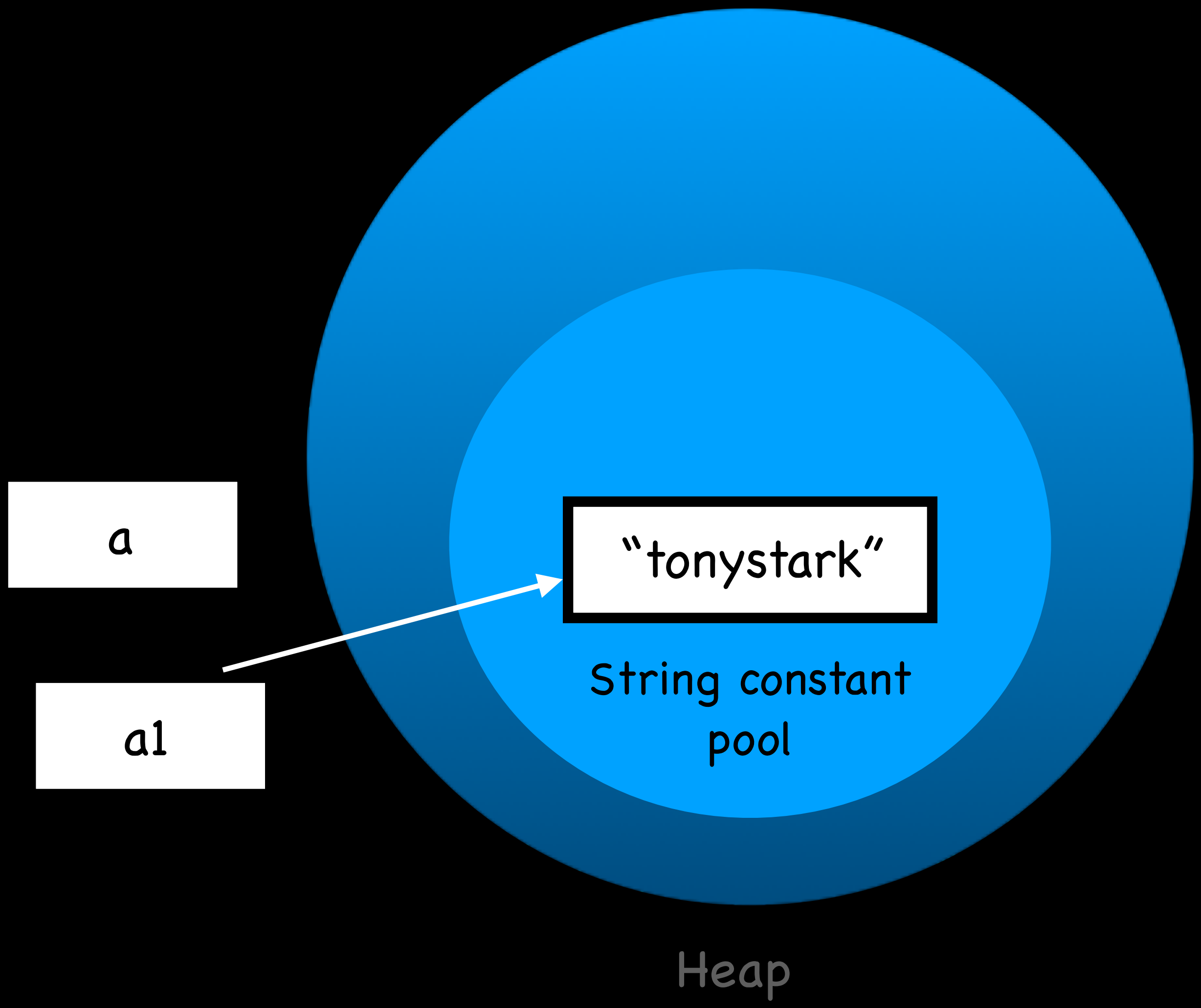
CharSequence Interface: The CharSequence interface is used to represent the sequence of characters. String, StringBuffer and StringBuilder classes implement it. It means, we can create strings in Java by using (String, StringBuffer and StringBuilder) these three classes.

→ The Java String is immutable which means it cannot be changed. Whenever we change any string, a new instance is created. For mutable strings, we can use StringBuffer and StringBuilder classes.

Note on String Literal: String literal is created by using double quotes (String s = "dhruvlenka"); Each time we create a string literal, the JVM checks the "string constant pool" first. If the string is already exist or matched 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.

Example:

```
String a = “tonystark”;
String a1 = “tonystark”;
```



In the above example, only one object will be created. Firstly, JVM will not find any string object with the value “tonystark: in string constant pool that is why it will create a new object. After that it will find the string with the value “tonystark” 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 object are created if it exists already in the string constant pool).

Program:

```java
package com.dhruv;

import java.net.SocketTimeoutException;
public class StringEx3 {
    public static void main(String[] args) {
        String name = "Jack Harlow";
        char ch[] = {'J','a','c','k',' ','R','y','a','n'};
        String name2 = new String(ch);
        String name3 = new String("Jack Sparrow");

        System.out.println(name);
        System.out.println(name2);
        System.out.println(name3);
    }
}
```

METHODS OF STRING:

1. **char charAt():** this method returns a char value at the given index number.

* The index number starts from 0 and goes n-1, where n is the length of the string. If the given index number is greater than or equal to this string length it will return StringIndexOutOfBoundsException.
* The method accepts index as a parameter.

Program:

```java
public class charAt {
    public static void main(String[] args) {
        String d = "dhruv";
        System.out.println(d.charAt(4));
    }
}
```

2. **int length():** this methods finds the length of a string.

Program:

```java
package com.dhruv;

public class Length {
    public static void main(String[] args) {
        String counTry1 = "India";
        String country2 = new String("Unites States of America");
        String country3 = "";

        System.out.println("String Length is: " + counTry1.length());
        System.out.println("String Length is: " +country2.length());
        System.out.println(country3.length());
    }
}
```

3. String Concat: It return a combined string.

Program:

```java
package com.dhruv;

public class Concat {
    public static void main(String[] args) {
        String s = "The Dhruv";
        s = s.concat(" Lenka");
        System.out.println(s);
    }
}
```

4. IndexOf: method returns the position of the first occurrence specified character or string in a specified string.

There are four overloaded indexOf() method.

1. int indexOf(int ch) - It returns the index position for the given char value.
2. int indexOf(int ch, int fromIndex) - It returns the index position for the given char value and from index.
3. int indexOf(String substring) - It return the index position for the given substring.
4. int indexOf(String substring, int fromIndex) - It returns the index position for the given substring and from index.
5. LastIndexOf: methods returns the last index of the given character value or substring. If it is not found it will return -1.

Same like IndexOf, It has four methods:

1. int lastIndexOf(int ch) - It returns last index position for the given char value.
2. int lastIndexOf(int ch, int fromIndex) - It returns last index position for the given char value and from index.

3. int lastIndexOf(String substring) - It returns last index position for the given substring.
4. int lastIndexOf(String substring, int fromIndex)- It returns last index position for the given substring and from index.

Program:

```java
package com.dhruv;
public class IndexLastIndex {
    public static void main(String[] args) {
        String quote = "Prove them wrong";
        /*
        Working of
        IndexOf: L -> R
        LastIndexOf: L <--  R
         */
        System.out.println(quote.indexOf("them"));
        System.out.println(quote.lastIndexOf("them"));
    }
}
```

6. CompareTo: method compares the given string with the current string. It returns a positive number, negative number or zero.

* It compares strings on the basic of Unicode value of each character in the strings.
* If the first string is greater than the second string (a > b) it returns a positive number (difference of character value). If the first string is less than the second string (a < b), it returns a negative number, and if the first string is equal to the second string (a==b), it returns 0.

```java
public class CompareTo {
    public static void main(String[] args) {
        String word = "Hii";
        String word2 = "hii";
        String word3 = "Bye";
        String word4 = "Boohoo";
        String word5 = "Hii";
        System.out.println(word.compareTo(word2));
        System.out.println(word.compareTo(word3));
        System.out.println(word.compareTo(word4));
        System.out.println(word.compareTo(word5));
    }
}
```

7. String replace: method returns a string replacing all the old char or CharSequence to new char or CharSequence.

Program:

```java
package com.dhruv;

public class Replace {
    public static void main(String[] args) {
        String s = "apple is the best tech company in the world";
        String replaceString = s.replace('a','b');
        System.out.println(replaceString);
    }
}
```

8. String 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.

```java
package com.dhruv;

public class Substring {
    public static void main(String[] args) {
        String s = "dhruvlenka";
        System.out.println(s.substring(1,4));
        System.out.println(s.substring(2));
    }
}
```

9. String[] split(): method splits this string against given regular expression and returns a char array.

```java
package com.dhruv;

public class Split {
    public static void main(String[] args) {
        String intro = "Hyy, I am Dhruv Lenka";
        String[] words = intro.split("\\s");
                for (String w:words ) {
                    System.out.println(w);
                }
    }
}
```

10. String toUpperCase: method converts lowercase words to uppercase.

11. String toLowerCase: method converts uppercase words to lowercase.

Program:

```java
public class UpperLowerCase {
    public static void main(String[] args) {
        String s = "GITHUB";
        System.out.println(s.toLowerCase());
        String social = "instagram";
        System.out.println(social.toUpperCase());
    }
}
```

12. String trim(): method remove whitespaces from both ends of a string.

Program:

```java
public class Trim {
    public static void main(String[] args) {
     String charActer = " Lucifer ";
     System.out.println(charActer+ "MorningStar"); //Without trim
     System.out.println(charActer.trim()+ "MorningStar"); //With trim
    }

```

13. boolean equals(): method compares the two given strings based on the content of the string. If any character is not matched, it return false. If all characters are matched, it return true.

Program:

```
public class Equals {
    public static void main(String[] args) {
        String a = "Apple";
        String b = "Ball";
        String c = "Cat";
        String d = "Dog";
        String e = "Apple";
        System.out.println(a.equals(b));
        System.out.println(a.equals(c));
        System.out.println(a.equals(d));
        System.out.println(a.equals(e));
 }
}
```

14. Equals IgnoreCase: method compares the two given string 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 case sensitivity. If any character is not matched, it returns false, if matched then it will return true.

Program:

```
public class EqualsIgnoreCase {
    public static void main(String[] args) {
        String a = "Apple";
        String b = "Bitcoin";
        String c = "duck duck go";
        String d = "Duck Duck Go";
        String e = "bitcoin";
        System.out.println(a.equalsIgnoreCase(b));
        System.out.println(b.equalsIgnoreCase(c));
        System.out.println(c.equalsIgnoreCase(d));
        System.out.println(b.equalsIgnoreCase(e));
    }
}
```

15. valueOf(): method converts different types of dataTypes(int, char, long, float, double) into string, and also converts object to string and char array to string.

* valueOf() method does not change the original string.

```java
public class ValueOf {
    public static void main(String[] args) {
        int a = 100;
        char c = 'D';
        float f = 68.50F;
        long l = 4500000l;
        double d = 4.55555555;

        String dataTy1 = String.valueOf(a);
        String dataTy2 = String.valueOf(c);
        String dataTy3 = String.valueOf(f);
        String dataTy4 = String.valueOf(l);
        String dataTy5 = String.valueOf(d);
        System.out.println(dataTy1);
        System.out.println(dataTy2);
        System.out.println(dataTy3);
        System.out.println(dataTy4);
        System.out.println(dataTy5);
    }
}
```

Extra from JavaTpoint

16. Contains(): method searches the sequence of characters in the string. It will return true if the sequence of character values is found in the string and if not found it will return false.

Program:

```java
public class Contains {
    public static void main(String[] args) {
        String str = "This program is for String contains()";
        System.out.println(str.contains("This")); //True
        System.out.println(str.contains("that")); //False
        System.out.println(str.contains("this")); //False
    }
}
```

17. endsWith(): method checks if the string ends with a given suffix. It returns true if not then returns false.

Program:

```java
public class endsWith {
    public static void main(String[] args) {
        String str = "thedhruvlenka";
        System.out.println(str.endsWith("lenka"));
        System.out.println(str.endsWith("the"));

    }
}
```

18. getBytes(): method encodes the string into a sequence of bytes and stores it in a byte array.
* When you write this code you will get ASCII values.

Program:

```java
public class GetBytes {
    public static void main(String[] args) {
        String s = "ABCDabcd";
        byte[] br = s.getBytes();
        for (byte b : br) {
            System.out.println(b);
        }
    }
}
```