Java String Function

String is a sequence of characters. In java, objects of String are immutable which means a constant and cannot be changed once created.

- in java, strings are treated as objects.

There are two ways to create string in Java:

- String literal

String s = “rohitShukla”;

- Using new keyword

String s = new String (“rohitShukla”);

Post: https://www.interviewbit.com/java-string-interview-questions/

Post: https://www.programiz.com/java-programming/library/string

String Methods

split()

The split() method divides the string at the specified regex and returns an array of substrings. We can pass the parameter in it. Example:

string.split(String regex, int limit)

The string split() method can take two parameters:

1. regex - the string is divided at this regex (can be strings)
2. limit (optional) - controls the number of resulting substrings
3. If the limit parameter is not passed, split() returns all possible substrings.

Note: If the regular expression passed to split() is invalid, the split() method raises *PatternSyntaxExpression* exception.

```
public static void main(String[] args) {
    String txt = "Hello My name is Rohit shukla";
    String[] m = txt.split(" ",3);
    for (String k : m) {
        System.out.println(k+", ");
    }
}
```

More: https://www.programiz.com/java-programming/library/string/split

compareTo() / compareToIgnoreCase

The compareTo() method compares two strings lexicographically (in the dictionary order). The comparison is based on the Unicode value of each character in the strings.

compareTo() Return Value

1. returns 0 if the strings are equal
2. returns a negative integer if the string comes before the str argument in the dictionary order
3. returns a positive integer if the string comes after the str argument in the dictionary order

```
class Main {
  public static void main(String[] args) {
    String str1 = "Learn Java";
    String str2 = "Learn Java";
    String str3 = "Learn Kolin";
    int result;

    // comparing str1 with str2
    result = str1.compareTo(str2);
    System.out.println(result);  // 0

    // comparing str1 with str3
    result = str1.compareTo(str3);
    System.out.println(result);  // -1

    // comparing str3 with str1
    result = str3.compareTo(str1);
    System.out.println(result);  // 1
  }
}
```

compareTo() takes the letter case into consideration. It means **learn** and **Learn** will be treat differently. But if you want to ignore this case, mean you want to treat **learn** and **Learn** same so you can use compareToIgnoreCase() and except this things as same as compareTo()

```
Sytax: string.compareTo(String str)
```

```
class Main {
  public static void main(String[] args) {
    String str1 = "java is fun";

    // extract substring from index 0 to 3
    System.out.println(str1.substring(0, 4));
  }
}

// Output: java
```

More: https://www.programiz.com/java-programming/library/string/compareto

More: https://www.programiz.com/java-programming/library/string/comparetoignorecase

length()

The length() method returns the length of the string. this method doesn't take any parameters.

```
Syntax: string.length()
```

More: https://www.programiz.com/java-programming/library/string/length

replace() / replaceAll()

The replace() method replaces each matching occurrence of the old character/text in the string with the new character/text.

```
Syntax: string.replace(char oldChar, char newChar)
```

it can replace single character and substring both.

More: https://www.programiz.com/java-programming/library/string/replace

Replace() didn't support regular expression (regex) but replaceAll() supports. Syntax is same of both.

```
string.replaceAll(String regex, String replacement)
```

```
class Main {
  public static void main(String[] args) {
    String str1 = "bat ball";

    // replace b with c
    System.out.println(str1.replace('b', 'c'));
  }
}

// Output: cat call
```

\\d use this regex to remove of numeric value from string.

\\s use this regex to remove of white space from string.

```
class Main {
  public static void main(String[] args) {
    String str1 = "Learn\nJava \n\n   ";
    String result;

    // replace all whitespace characters with empty string
    result = str1.replaceAll("\\s", "");
    System.out.println(result);    // LearnJava
  }
}
```

replace() method is a bit faster than replaceAll() due to regex.

substring()

The substring() method return the substring on that correct index number. And we can also pass start Index & end index then substring then output will be substring between both indexes. We can also paas one index value. Java treats it like a start index.

```
string.substring(int startIndex, int endIndex)
```

```
class Main {
  public static void main(String[] args) {
    String str1 = "program";

    // from the first character to the end
    System.out.println(str1.substring(0));  // program

    // from the 4th character to the end
    System.out.println(str1.substring(3));  // gram
  }
}
```

Example

--- and ---

More: https://www.programiz.com/java-programming/library/string/substring

equals()

The equals() method returns true if two strings are equal. If not, it returns false.

it returns true if the strings are equal and return false if strings are not equal and the str argument is null. It treats **learn** and **Learn** different. If you want to ignore this case then use **equalsIgnoreCase()** in place of **equals()** .

trim()

The trim() method removes any leading (starting) and trailing (ending) whitespaces from the specified string. It doesn't remove whitespace that appears in the middle. it doesn't take any parameters.

If you need to remove all whitespace characters from a string, you can use the String replaceAll() method with proper regex. Like this →

```
class Main {
  public static void main(String[] args) {
    String str1 = "Learn Java";
    Boolean result;

    // check if str1 contains "Java"
    result = str1.contains("Java");
    System.out.println(result);  // true

    // check if str1 contains "Python"
    result = str1.contains("Python");
    System.out.println(result);  // false

    // check if str1 contains ""
    result = str1.contains("");
    System.out.println(result);  // true
  }
}
```

contains()

The contains() method checks whether the specified string (sequence of characters) is present in the string or not. If string is present then it return “true” and string is not present then it return “False”. It treats **learn** and **Learn** different.

indexOf()

The indexOf() method returns the index of the specified character or substring which is presented in the string. It return output in numeric.

We can pass two parameter in this. Character/substring and index number. Index number is optional but if we pass then searching will start from that index only. Character/substring is mandatory whose we want to search. It treats **learn** and **Learn** different. it return -1 when character/subString is not found. substring() is just opposite of it.

```
String s = ”Learn Share Learn”;
int output = s.indexOf(“Share”); // returns 6


String s = ”Learn Share Learn”;
int output = s.indexOf("ea",3);// returns 13
```

lastIndexOf()

we it work from last of the string and return the specific index of the substring/character.

charAt()

The charAt() method returns the character at the specified index. Here we pass the index number and it return the character which is located on that index in String.

```java
class Main {
  public static void main(String[] args) {
    String str1 = "Learn Java";
    String str2 = "Learn\nJava";

    // first character
    System.out.println(str1.charAt(0));  // 'L'

    // seventh character
    System.out.println(str1.charAt(6));  // 'J'

    // sixth character
    System.out.println(str2.charAt(5));  // '\n'
  }
}
```

toLowerCase()

The toLowerCase() method converts all characters in the string to lowercase characters.

```java
class Main {
  public static void main(String[] args) {
    String str1 = "Learn Java";
    String str2 = "Java123";

    // convert to lowercase letters
    System.out.println(str1.toLowerCase());  // "learn java"
    System.out.println(str2.toLowerCase());  // "java123"
  }
}
```

concat()

The concat() method concatenates (joins) two strings and returns it.

The concat() method takes a single parameter. In Java, you can also use the + operator to concatenate two strings.

```java
class Main {
  public static void main(String[] args) {
    String str1 = "Java";
    String str2 = "Programming";

    // concatenate str1 and str2
    System.out.println(str1.concat(str2));

  }
}
```

```
// Output: JavaProgramming
```

concat() Vs the + Operator for Concatenation

concat()	the + Operator
Suppose, **str1** is `null` and **str2** is `"Java"`. Then, `str1.concat(str2)` throws **NullPointerException**.	Suppose, **str1** is `null` and **str2** is `"Java"`. Then, `str1 + str2` gives **"nullJava"**.
You can only pass a String to the `concat()` method.	If one of the operands is a string and another is a non-string value. The non-string value is internally converted to a string before concatenation. For example, `"Java" + 5` gives `"Java5"`.

valueOf()

The valueOf() method returns the string representation of the argument passed. The valueOf() method takes a single parameter. we can convert char and char array to String by to value() method. we have to just pass array name in parameter.

```
class Main {
  public static void main(String[] args) {
    int a = 5;
    long l = -2343834L;
    float f = 23.4f;
    double d = 923.234d;

    // convert numbers to strings
    System.out.println(String.valueOf(a));  // "5"
    System.out.println(String.valueOf(l));  // "-2343834"
    System.out.println(String.valueOf(f));  // "23.4"
    System.out.println(String.valueOf(d));  // "923.234"
  }
}
```

we can also pass array, offset, length in parameter. syntax should be like this -->

syntax: valueOf**(**char[] data, int offset, int length**)**

offerset: it will left this number of elemnt from starting and continue to work after that.

length: it mean output/ string length

```
class Main {
  public static void main(String[] args) {
    char ch[] = {'p', 'r', 'o', 'g', 'r', 'a', 'm'};
    int offset = 2;
```

```
    int length = 4;
    String result;

    // subarray {'o', 'g', 'r', 'm'} is converted to string
    result = String.valueOf(ch, offset, length);

    System.out.println(result);  // "ogrm"
  }
}
```

matches()

The matches() method checks whether the string matches the given regular expression or not. and give the output in true /false

More: https://www.programiz.com/java-programming/library/string/matches

startsWith()

As it's name shows the startsWith() method checks whether the string begins with the specified string or not. and give the output in true and false.

we can also pass the offset along with Stringt.

syntax: string.startsWith**(**String str, int offset**)**

offset mean as same here as I earlier told.

More: https://www.programiz.com/java-programming/library/string/startswith

endsWith()

The Java String endsWith() method checks whether the string ends with the specified string or not.

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

    String str = "Java Programming";

    System.out.println(str.endsWith("mming")); // true
    System.out.println(str.endsWith("a Programming")); // true
    System.out.println(str.endsWith("Java")); // false
  }
}
```

isEmpty()

The Java String isEmpty() method checks whether the string is empty or not. The isEmpty() method does not take any parameters. it give output in true and false.

intern()

Suppose we are intializing 2 variable and declaring same string in both of them. now they will capture the memory diffently or only one time bcoz value of both variable is same. obviously they will captue the memory diffrently bcoz pool of both value is diffrent but this looks weird bcoz the value of both varibale is same to same then we should they are not covering onr space?

str1 and str2 have the same content. However, they are not equal because they don't share the same memory. you can manually use the intern() method so that the same memory is used for strings having the same content. Now they will use same string pool.

hashCode()

The Java String hashCode() method returns a hash code for the string. A hashcode is a number (object's memory address) generated from any object, not just strings. This number is used to store/retrieve objects quickly in a hashtable.

join()

The join() method returns a new string with the given elements joined with the specified delimiter.

Syntax: String.join**(**CharSequence delimiter, CharSequence... elements**)**

Here, ... signifies there can be one or more CharSequence.

```
class Main {
  public static void main(String[] args) {
    String str1 = "I";
    String str2 = "love";
    String str3 = "Java";

    // join strings with space between them
    String joinedStr = String.join(" ", str1, str2, str3);

    System.out.println(joinedStr);
  }
}

// Output: I love Java
```

//++++++++++++++++++++

```
import java.util.ArrayList;

class Main {
  public static void main(String[] args) {
    ArrayList<String> text = new ArrayList<>();

    // adding elements to the arraylist
    text.add("Java");
    text.add("is");
    text.add("fun");

    String result;

    result = String.join("-", text);

    System.out.println(result);  // Java-is-fun
```

```
  }
}
```

replaceFirst()

The Java String replaceFirst() method replaces the first substring that matches the regex of the string with the specified text.

Syntax: string.replaceFirst**(**String regex, String replacement**)**

subSequence()

The Java String subSequence() method returns a character sequence (a subsequence) from the string.

Syntax: string.subSequence**(**int startIndex, int endIndex**)**

```
class Main {
  public static void main(String[] args) {
    String str = "Java Programming";

    System.out.println(str.subSequence(3, 8)); // a Pro
  }
}
```

toCharArray()

The Java String toCharArray() method converts the string to a char array and returns it. this method doesn't take any parameters.

Syntax: string.toCharArray()

format()

The format() method returns a formatted string based on the argument passed.

More: https://www.programiz.com/java-programming/library/string/format

Important Points about String in java

Strings are derived data types in java. Derived data types are those that are made by using any other data type for example, arrays.

Java treats Strings as objects, not arrays. String objects are created using the java.lang.String class. String objects in Java are immutable; you cannot modify their contents. This means whenever we manipulate a String object, the new String is created rather than the original string being modified.

Strings are immutable in Java. Immutable objects mean they can't be changed or altered once they've been created. However, we can only modify the reference to the string object. The String is immutable in Java because of many reasons like security, caching, synchronization and concurrency, and class loading.

we can use a string in the switch case in java.

In Java, every immutable object is thread-safe, which means String is also thread-safe.

String pool in Java.

String Pool, also known as SCP (String Constant Pool), is a special storage space in Java heap memory that is used to store unique string objects. Whenever a string object is created, it first checks whether the String object with the same string value is already present in the String pool or not, and if it is available, then the reference to the string object from the string pool is returned. Otherwise, the new string object is added to the string pool, and the respective reference will be returned.

intern() method do in Java?

If you apply the intern() method to a few strings, you will ensure that all strings having the same content share the same memory in string pool

Difference between StringBuffer and StringBuilder in Java.

StringBuffer and StringBuilder are two Java classes for manipulating strings. These are mutable objects, i.e., they can be modified, and provide various methods such as insert(), substring(), delete(), and append(), for String manipulation.

StringBuffer: The StringBuffer class was created by the Java Team when they realized the need for an editable string object. Nevertheless, StringBuffer has all methods synchronized, meaning they are thread-safe. Therefore, StringBuffer allows only one thread to access a method at once, so it is not possible to call StringBuffer methods from two threads simultaneously, which means it takes more time to access. The StringBuffer class has synchronized methods, making it thread-safe, slower, and less efficient than StringBuilder. The StringBuffer class was introduced in Java 1.0.

Syntax:

StringBuffer var = new StringBuffer(str);

StringBuilder: It was at that point that the Java Team realized that making all methods of StringBuffer synchronized wasn't the best idea, which led them to introduce StringBuilder. The StringBuilder class has no synchronized methods. Unlike StringBuffer, StringBuilder does not offer synchronized methods, which makes it less thread-safe, faster, and more efficient. StringBuilder was introduced in Java 1.5 in response to StringBuffer's shortcomings.

Syntax:

StringBuilder var = new StringBuilder(str);

Difference between str1 == str2 and str1.equals(str2)?

Java offers both the equals() method and the "==" operator for comparing objects. However, here are some differences between the two:

1) Essentially, equals() is a method, while == is an operator.
2) The == operator can be used for comparing references (addresses) and the .equals() method can be used to compare content. To put it simply, == checks if the objects point to the same memory location, whereas .equals() compares the values of the objects.

```
public class StringComparison
{
   public static void main(String[] args)
   {
       String str1=new String("Scaler");
       String str2=new String("Scaler");
       System.out.println(str1 == str2);
       System.out.println(str1.equals(str2));
   }
}
```

Output:

```
false
true
```

How can a Java string be converted into a byte array?

The getBytes() method allows you to convert a string to a byte array

How do you convert a string to an integer and vice versa?

There is an Integer class in the Java lang package that provides different methods for converting strings to integers and vice versa.

```
public class StringtoInteger {
  public static void main(String args[])
  {
    String str1 = "1296";
    int  i= Integer.parseInt(str1);
    System.out.println(i);
    String str2 = Integer.toString(i);
    System.out.println(str2);
  }
}
```

How can we convert string to StringBuilder?

The append() method can be used to convert String to StringBuilder, and the toString() method can be used to convert StringBuilder to String.

```
public class StringToStringBuilder {
  public static void main(String args[]) {
    String strs[] = {"Scaler", "by", "InterviewBit!"};
    StringBuilder sb = new StringBuilder();
    sb.append(strs[0]);
    sb.append(" "+strs[1]);
    sb.append(" "+strs[2]);
    System.out.println(sb.toString());
```

```
    }
}
```

How do you check whether a String is empty in Java?

java provide isEmpty() method to check whether a string is empty. if output is true it means string is empty else false.

String Pool

String Pool is a pool of Strings stored in Java heap memory. We know that String is a special class in Java and we can create String object using new operator as well as providing values in double quotes.