# Java String

## Java String Manipulation

Java provides the String class that contains a lot of useful methods to operate on strings. For example, you can use Java string operations methods to find the length of a string, get a character in a specified index of a string, combine two strings, change cases of a string to lower case or upper case, find the position of a character or a substring in another string, convert a string to an array of character, get a substring from a string, compare two strings, test a string whether it starts with or ends with a substring, and replace an existing substring with a new substring in a string. The Java string operation methods that can handle those such things can be length(), charAt(), concat(), toLowerCase, toUpperCase, indexOf(), toCharArray(), substring(), compareTo(), equals(), startsWith(), endsWith(), and replace() methods.

### The length() method

The length() method can be used to count the number of characters in a string object. It is much like that you count the number of elements in an array object. In the example below we use the length() method to test whether the user entered the correct length of password field that is greater than 8 characters.

Example:

```
import java.io.*;

public class JavaExercises
{
public static void main(String[] args)
{
strmanip();
}

static void strmanip(){
String name="";
String password="";
try{
BufferedReader br=new BufferedReader(new InputStreamReader(System.in));
System.out.print("Enter Username:");
name=br.readLine();
System.out.print("Enter Password:");
password= br.readLine();
}catch(IOException e){}

if (password.length() > 8)
  System.out.println("Welcome!");
else System.out.println("The password must be greater than 8 characters.");
```

```
  }

}

}
```

**The charAt(int index) method**

To get a character in a specified position from a string, you can use the charAt(int index) method. This method returns a character that is specified by the index in a string. In the example below, the charAt(int index) method is used to test whether a sentence input by the use contains a dot at the end or not. The last index of a string is equal to the length of the string subtracted by 1.

Example:

```
import java.io.*;

public class JavaString
{
public static void main(String[] args)
{
strmanip();
}

static void strmanip(){
String se="";

try{
BufferedReader br=new BufferedReader(new InputStreamReader(System.in));
System.out.print("Enter a sentence:");
se= br.readLine();
}catch(IOException e){}

if (se.charAt(se.length()-1)!='.')
System.out.println("A sentence must end with a dot.");
else System.out.println("A good sentence!");



}

}
```

**The concat(String) method**

In fact, you can concatenate two strings by using the + operator. In this page, you will learn another way to concatenate two strings. The latter is using the concat(String) method.

Example:

```
public class JavaExercises
{
public static void main(String[] args)
{
  strmanip();
}

static void strmanip(){
String st1="Learn Java";
String st2=" programming";
String st3=st1.concat(st2);
System.out.println("st1:"+st1);
System.out.println("st2:"+st2);
System.out.println("The result from concatenating:"+st3);

   }

}
```

**The toLowerCase() and toUppercase() methods**

You can convert all letters of a string to lowercase by using toLowerCase() method. In contrast, to convert all letters of a string to uppercase, you need to use toUppercase() method.

Example:

```
import java.io.*;

public class JavaExercises
{
public static void main(String[] args)
{
strmanip();
}

static void strmanip(){
String st1="Learn Java programming";
System.out.println("Lower case:"+st1.toLowerCase());
System.out.println("Upper case:"+st1.toUpperCase());

 }

}
```

**The indexOf() method**

The indexOf() method can be used to find the index of a character or a substring in a string. This method has four different prototypes:

-indexOf(char c) finds the position of character c argument by searching from the beginning in a string.

-indexOf(char c, int start_position) finds the position of character c argument by searching from the start_position argument in a string.

-indexOf(String st) finds the position of st substring argument by searching from the beginning in a string.

-indexOf(String st, int start_position) finds the position of st substring argument by searching from the start_position argument in a string.

This method stops searching process when it first meets the target character or substring in a string. It continues searching the target until it reaches the end of the string.

In the example below, we use the indexOf() method to get the position of the @ character in an e-mail address input by the user. If the @ character resides in the middle of email address string, we say that it is a valid e-mail address.

Example:

```
import java.io.*;

public class JavaExercises
{
public static void main(String[] args)
{
strmanip();
}

static void strmanip(){
String email="";


try{
BufferedReader br=new BufferedReader(new InputStreamReader(System.in));
System.out.print("Enter an e-mail address:");
email= br.readLine();
}catch(IOException e){}

if (email.indexOf('@')>0 && email.indexOf('@')<email.length()-1)
System.out.println("It is a valid e-mail address.");
else System.out.println("It is an invalid e-mail address.");

 }

}
```

**The toCharArray() method**

The toCharArray() method can be used to convert a string to an array of characters. In example below, we try to validate an e-mail address input by the user. The e-mail is said to be valid if it fulfill the following requirements:

-It contains the @ character at the middle of the e-mail address string.

-The dot character that resides after the @ character can occur only one time.

In the previous example , we already used the indexOf() method to meet the first requirement. To fulfill the second requirement, we use toCharArray() method to convert an e-mail string to an array of characters. Then dot characters that stay after the @ character are counted.

Example:

```
import java.io.*;

public class JavaExercises
{
public static void main(String[] args)
{
strmanip();
}

static void strmanip(){
String email="";
int i,count_dot=0;
char[] st=null;



try{
BufferedReader br=new BufferedReader(new InputStreamReader(System.in));
System.out.print("Enter an e-mail address:");
email= br.readLine();
st=email.toCharArray();
}catch(IOException e){}


for(i=email.indexOf('@');i<email.length();i++)
  if(st[i]=='.') count_dot++;

if (email.indexOf('@')>0 && email.indexOf('@')<email.length()-1 && count_dot==1)
  System.out.println("It is a valid e-mail address.");
else System.out.println("It is an invalid e-mail address.");

 }
```

```
}
```

**The substring() method**

The substring(int start, [int length]) method can be used to to get a substring from a string. The substring to be taken out starts from the start argument and the number of characters to be taken out is specified by the length argument. If the length argument is not specified, the substring to be taken out starts from the start argument until the end of the string. In the example below, we want to take word "Java" from the string "Java programming". Therefore the start index is 0 and the length index is 4.

Example:

```
public class JavaExercises
{
public static void main(String[] args)
{
        strmanip();
}

static void strmanip(){
        String st="Java programming";
        String subst=st.substring(0,4);
        System.out.println(subst);

 }

}
```

For another example, we write a Java program to ask the user to input his/her full name then the substring() method is used to separate the first name from the last name.

```
import java.io.*;

public class JavaExercises
{
public static void main(String[] args)
{
strmanip();
}

static void strmanip(){

        String name="";
        String fname="";
        String lname="";
   try{
        BufferedReader br=new BufferedReader(new InputStreamReader(System.in));
        System.out.print("Enter your full name:");
```

```
        name= br.readLine();
}catch(IOException e){}

        fname=name.substring(0,name.indexOf(' '));
        lname=name.substring(name.indexOf(' '));

        System.out.println("Your first name is:"+fname);
        System.out.println("Your last name is:"+lname);


  }

}
```

**The compareTo() and equals() methods**

You can use the CompareTo(String) method to compare the invoking string with the string specified by the string argument. This function returns 0 if both strings are equal. It returns 1 if the invoking string is greater than the string specified by the string argument, otherwise it returns -1. Another method to compare strings in Java is equals(String) method. This method return true if both strings are equals, otherwise it return false.

Example:

```
import java.io.*;

public class JavaExercises
{
public static void main(String[] args)
{
        selectChoice();
}

static void selectChoice(){

String choice;
String con="y";
try{
        BufferedReader br=new BufferedReader(new InputStreamReader(System.in));
        System.out.println("What is the command keyword to exit a loop in Java?");
        System.out.println("a.quit");
        System.out.println("b.continue");
        System.out.println("c.break");
        System.out.println("d.exit");


while (con.compareTo("y")==0)
{
```

```
        System.out.print("Enter your choice:");
  choice =br.readLine();

if (choice.compareTo("c")==0)
{
  System.out.println("Congratulation!");
}
else if (choice.compareTo("q")==0 || choice.compareTo("e")==0) {
System.out.println("Exiting...!"); break; }
else System.out.println("Incorrect!");

System.out.print("Again? press y to continue:");
con =br.readLine();
}

}catch(IOException e){}


  }

}
```

The example code above can be rewritten as shown below:

```
import java.io.*;

public class JavaExercises
{
public static void main(String[] args)
{
        selectChoice();
}

static void selectChoice(){

String choice;
String con="y";
try{
        BufferedReader br=new BufferedReader(new InputStreamReader(System.in));
        System.out.println("What is the command keyword to exit a loop in Java?");
        System.out.println("a.quit");
        System.out.println("b.continue");
        System.out.println("c.break");
        System.out.println("d.exit");


while (con.equals("y"))
{
```

```
        System.out.print("Enter your choice:");
        choice =br.readLine();

if (choice.equals("c"))
{
        System.out.println("Congratulation!");
}
else if (choice.equals("q") || choice.equals("e")) { System.out.println("Exiting...!"); break; }
else System.out.println("Incorrect!");

System.out.print("Again? press y to continue:");
con =br.readLine();
}

}catch(IOException e){}


  }

}
```

**The startsWith() and endsWith methods**

The startsWith(String) can be used to test whether a string contains a substring at the beginning. In contrast, you can use endsWith() method to test whether a string contains a substring at the end.

Example:

```
import java.io.*;

public class JavaExercises
{
public static void main(String[] args)
{
        strmanip();
}

static void strmanip(){

String filename="";

try{
        BufferedReader br=new BufferedReader(new InputStreamReader(System.in));
        System.out.print("Enter the file name:");
        filename= br.readLine();
}catch(IOException e){}
```

```
if(filename.endsWith(".txt"))
 System.out.println("Valid file name!");
else
 System.out.println("invalid file name!");


}

}
```

**The replace() method**

You can use the replace(Old_String, New_String) method to replace an old substring with a new substring in a string.

**Example:**

```
import java.io.*;

public class JavaExercises
{


public static void main(String[] args)
{
        strmanip();
}

static void strmanip(){

        String st="C#, C, C++, Java, Python tutorial";
        String newst = st.replace(",", " tutorial");
        System.out.println(newst);


    }

}
```