String Operations

The most commonly-used String methods are:

- length(): return the length of the string.
- charAt(int index): return the char at the index position (index begins at 0 to length()-1).
- equals(): for comparing the contents of two strings. Take note that you cannot use "==" to compare two strings.

For examples,

```
String str = "Java is cool!";

System.out.println(str.length());       // return int 13
System.out.println(str.charAt(2));      // return char 'v'
System.out.println(str.charAt(5));      // return char 'i'

// Comparing two Strings
String anotherStr = "Java is COOL!";
System.out.println(str.equals(anotherStr));           // return boolean false
System.out.println(str.equalsIgnoreCase(anotherStr)); // return boolean true
System.out.println(anotherStr.equals(str));           // return boolean false
System.out.println(anotherStr.equalsIgnoreCase(str)); // return boolean true
// (str == anotherStr) to compare two Strings is WRONG!!!
```

To check all the available methods for String, open JDK API documentation ⇒ select *package* "java.lang" ⇒ select *class* "String" ⇒ choose *method*. For examples,

```
String str = "Java is cool!";

System.out.println(str.length());           // return int 13
System.out.println(str.charAt(2));          // return char 'v'
System.out.println(str.substring(0, 3));    // return String "Jav"
System.out.println(str.indexOf('a'));       // return int 1
System.out.println(str.lastIndexOf('a'));   // return int 3
System.out.println(str.endsWith("cool!"));  // return boolean true
System.out.println(str.toUpperCase());      // return a new String "JAVA IS COOL!"
System.out.println(str.toLowerCase());      // return a new String "java is cool!"
```

Exercise 1: Given a string, return a new string made of 3 copies of the last 2 chars of the original string. The string length will be at least 2.

extraEnd("Hello") → "lololo"
extraEnd("ab") → "ababab"
extraEnd("Hi") → "HiHiHi"

```
public static String extraEnd(String str) {

}
```

Hint:

```
public class Ex01{
	public static void main(String [] arg){
		String str=”Lac Hong”;
		System.out.print(“Output: “ + extraEnd(str));
	}
	public static String extraEnd(String str){
		//Write your code here
	}
}
```

Exercise 2: Given two strings, a and b, return the result of putting them together in the order abba, e.g. "Hi" and "Bye" returns "HiByeByeHi".

makeAbba("Hi", "Bye") → "HiByeByeHi"
makeAbba("Yo", "Alice") → "YoAliceAliceYo"
makeAbba("What", "Up") → "WhatUpUpWhat"

```
public static String makeAbba(String a, String b) {



}
```

Exercise 3: Given a string, if the string begins with "red" or "blue" return that color string, otherwise return the empty string.

seeColor("redxx") → "red"
seeColor("xxred") → ""
seeColor("blueTimes") → "blue"

```
public static String seeColor(String str) {



}
```

Exercise 4: Given a string of odd length, return the string length 3 from its middle, so "Candy" yields "and". The string length will be at least 3.

middleThree("Candy") → "and"
middleThree("and") → "and"
middleThree("solving") → "lvi"

```
public static String middleThree(String str) {



}
```

Exercise 5: Given 2 strings, a and b, return a new string made of the first char of a and the last char of b, so "yo" and "java" yields "ya". If either string is length 0, use '@' for its missing char.

lastChars("last", "chars") → "ls"
lastChars("yo", "java") → "ya"
lastChars("hi", "") → "h@"

```
public static String lastChars(String a, String b) {



}
```

Exercise 6: Given a string, return true if it ends in "ly".

endsLy("oddly") → true
endsLy("y") → false
endsLy("oddy") → false

```
public static boolean endsLy(String str) {


}
```

Exercise 7: Given a string of even length, return the first half. So the string "WooHoo" yields "Woo".

firstHalf("WooHoo") → "Woo"
firstHalf("HelloThere") → "Hello"
firstHalf("abcdef") → "abc"

```
public static String firstHalf(String str) {


}
```

Exercise 8: Given 2 strings, return their concatenation, except omit the first char of each. The strings will be at least length 1.

nonStart("Hello", "There") → "ellohere"
nonStart("java", "code") → "avaode"
nonStart("shill", "java") → "hillava"

```
public static String nonStart(String a, String b) {


}
```

Exercise 9: Given an "out" string length 4, such as "<<>>", and a word, return a new string where the word is in the middle of the out string, e.g. "<<word>>".

makeOutWord("<<>>", "Yay") → "<<Yay>>"
makeOutWord("<<>>", "WooHoo") → "<<WooHoo>>"
makeOutWord("[[]]", "word") → "[[word]]"

```
public static String makeOutWord(String out, String word) {


}
```

Exercise 10: Given a string, return a string length 1 from its front, unless **front** is false, in which case return a string length 1 from its back. The string will be non-empty.

theEnd("Hello", true) → "H"
theEnd("Hello", false) → "o"
theEnd("oh", true) → "o"

```
public static String theEnd(String str, boolean front) {


}
```

Exercise 11: Given a string, return a version without the first and last char, so "Hello" yields "ell". The string length will be at least 2.

withoutEnd("Hello") → "ell"
withoutEnd("java") → "av"
withoutEnd("coding") → "odin"

```
public static String withoutEnd(String str) {


}
```

Exercise 12: Given a string, return a "rotated left 2" version where the first 2 chars are moved to the end. The string length will be at least 2.

left2("Hello") → "lloHe"
left2("java") → "vaja"
left2("Hi") → "Hi"

```
public static String left2(String str) {


}
```

Exercise 13: Given a string, return a version without both the first and last char of the string. The string may be any length, including 0.

withouEnd2("Hello") → "ell"
withouEnd2("abc") → "b"
withouEnd2("ab") → ""

```
public static String withouEnd2(String str) {


}
```

Exercise 14: Given a string and an int n, return a string made of the first and last n chars from the string. The string length will be at least n.

nTwice("Hello", 2) → "Helo"
nTwice("Chocolate", 3) → "Choate"
nTwice("Chocolate", 1) → "Ce"

```
public static String nTwice(String str, int n) {


}
```

Exercise 15: Given two strings, append them together (known as "concatenation") and return the result. However, if the concatenation creates a double-char, then omit one of the chars, so "abc" and "cat" yields "abcat".

conCat("abc", "cat") → "abcat"
conCat("dog", "cat") → "dogcat"
conCat("abc", "") → "abc"

```
public static String conCat(String a, String b) {


}
```

Exercise 16: The web is built with HTML strings like "<i>Yay</i>" which draws Yay as italic text. In this example, the "i" tag makes <i> and </i> which surround the word "Yay". Given tag and word strings, create the HTML string with tags around the word, e.g. "<i>Yay</i>".

makeTags("i", "Yay") → "<i>Yay</i>"
makeTags("i", "Hello") → "<i>Hello</i>"
makeTags("cite", "Yay") → "<cite>Yay</cite>"

```
public static String makeTags(String tag, String word) {


}
```