**Advanced Placement Computer Science**

[**Shenendehowa HS**](http://www.shenet.org/shen-high-school/)[**mr Hanley**](http://hanley.co.nr)

**Unit 5: Strings/Short Circuit EvaluationFILLED\_OUT**

**Lesson: java.lang.String**

***Last Updated:*** *4/27/2017*

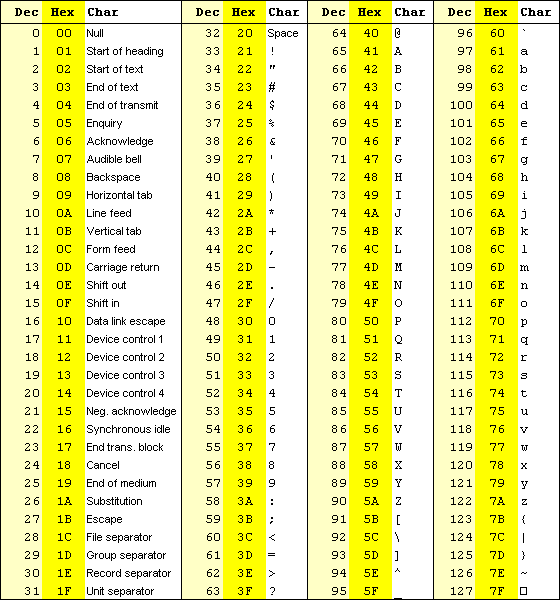
Lesson: Parameter Passing Mechanisms

*Last Updated: 100/11001/1100*

The String class is a special class in Java.

What is a char?Code that represents a single character

String manages an array of Unicode characters  
  
What is Unicode? Internationally friendly set of codes for language  
  
What is ASCII? American Standard Code for Information Interchange (created for teletype machines)

The String class has 3077 lines of code and comments!!!!!!!!!!!!!!!!!  


Some of the characteristics

1. References  
     
   References point to objects, they are NOT objects.
2. No need for new  
     
   Only class which doesn’t require new  
   String name = “Andrew”;
3. Immutability  
     
   Strings cannot be changed once created.
4. Operator  
     
   Only class that uses cool operators

String name = “Bailey” + “ “ + “Moore”;

+concatenation  
+=  
name+=” the genius”;

What is the AP String subset? Although String has many methods, only these are required for the AP Exam  
class java.lang.String implements java.lang.Comparable

* int compareTo(Object other)  
  // specified by java.lang.Comparable
* boolean equals(Object other)
* int length()
* String substring(int from, int to)   
  // returns the substring beginning at from  
  // and ending at to-1
* String substring(int from)  
  // returns substring(from, length())
* int indexOf(String s)  
  // returns the index of the first occurrence of s;  
  // returns -1 if not found

The following methods are explained in the context of these two strings

String s1 = “Blue bird,black bird!”, s2 = “bird”;

**length() :** returns number of characters in String(incl spaces)What is the length of s1\_21 s2\_4  
  
equals returns true if Strings are identical including case

if(s1.equals(s2) == true) { //this is how you compare 2 strings!  
NOTE: Sometimes s1 == s2 will work but NOT guaranteed!!!!

**indexOf** searches for a target string, returns -1 if not found

1st version

“ben Hogan”.indexOf(“n”) = 2

2nd version : takes in a place to start looking

“ben Hogan”.indexOf(“n”,5) = 8

**compareTo**

character by character ASCII subtraction.

“Amy”.compareTo(“Arun”) = negative number which means that Arun comes after Amy

String name1 = “IBM”, name2 = “Apple”, name3= “Oracle”, name4 = “BEA Systems”, name5 = “Hewlett Packard” name6 = “IBM”;

When using name1.compareTo all of the other strings, which comparisons yield a negative result?  
  
Which yield a 0?

“aa”.compareTo(“ab”)=-1  
“ab”.compareTo(“aa”)=1  
“aa”.compareTo(“aa”)=0  
“aaa”.compareTo(“aa”)=97

**substring**

takes a specified subset of the String

1st version

“Ben Gile”.substring(1,5) (first you want, first no want) = “en G”

2nd version

Has only one parameter: start here and take rest  
“Ben Gile”.substring(3) = “ Gile”

1. “subway”.substring(3,6)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| s | u | b | w | a | y |
| 0 | 1 | 2 | 3 | 4 | 5 |

“way”

1. “subway”.substring(0,3);

“sub”

1. “subway”.substring(1,3);

“ubw”

1. “subway”.substring(5);

“y”

1. “subway”.substring(0);

“subway”

1. “subway”.substring(-1,3);

index of out bounds exception

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| s | u | b | w | a | y | ‘ ‘ | s | t | a | t | i | o | n |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 |

1. “subway station”.substring(3,”subway station”.length());

“way station”

1. “subway”.indexOf(“w”);

3

1. “subway”.indexOf(“s”);

0 (only finds first!)

1. “subway station”.indexOf(“ ”);

6

1. “subway station”.indexOf(“S”);

-1

1. “subway station”.indexOf(“sta”);

7

1. “subway station”.indexOf(“a”);

4

//indexOf is overloaded

1. “subway station”.indexOf(“a”,5);

9

A string is really stored as an array of characters

String name = “Joseph”;

Convert this string to a character array

char[]myArray = name.toCharArray();

We can also examine individual characters with charAt

String city = “Constantinople”;

Use the charAt command to pull out all of the even index letters and print them to the screen

for(int i=0; i< city.length(); i+=2){  
 System.out.println(city.charAt(i);

}

Check to see if the first letter of the city is upper case using Character.isUpper

if(Character.isUpper(city.charAt(0)))

System.out.println(“this city starts with an upper case!”);

The **fall of Constantinople**, also known as the **conquest of Constantinople**, was the capture of [the capital](https://en.wikipedia.org/wiki/Constantinople) of the [Byzantine Empire](https://en.wikipedia.org/wiki/Byzantine_Empire) by the [Ottoman Empire](https://en.wikipedia.org/wiki/Ottoman_Empire). The city was captured on 29 May 1453 as part of the culmination of a 55-day [siege](https://en.wikipedia.org/wiki/Siege) which had begun on 6 April.

The attacking [Ottoman Army](https://en.wikipedia.org/wiki/Army_of_the_classical_Ottoman_Empire), which significantly outnumbered Constantinople's defenders, was commanded by the 21-year-old [Sultan](https://en.wikipedia.org/wiki/List_of_sultans_of_the_Ottoman_Empire) [Mehmed II](https://en.wikipedia.org/wiki/Mehmed_the_Conqueror) (later nicknamed "the Conqueror"), while the [Byzantine army](https://en.wikipedia.org/wiki/Byzantine_army_(Palaiologan_era)) was led by [Emperor](https://en.wikipedia.org/wiki/List_of_Byzantine_emperors) [Constantine XI Palaiologos](https://en.wikipedia.org/wiki/Constantine_XI_Palaiologos). After conquering the city, Mehmed II made Constantinople the new Ottoman capital, replacing [Adrianople](https://en.wikipedia.org/wiki/Edirne).

String x = some first name, then a space then a last name, for example “Marissa Mayer”;  
  
Create a first name String and a last name String and parse out the first and last names from the String x. You may find it helpful to declare an integer to find where the space is located (Make this work for any first and last name combinations)  
int space = x.indexOf(“ “);

String first = x.substring(0,space);

String last = x.substring(space+1);

1. Use length and substring to loop through a string read in from the keyboard to variable string1 and count how many times the character ‘X’ appears

String string1 = input.nextLine();

int count = 0;

int i = 0;

while(i < string1.length()){

if(string1.substring(i, i+1).equals(“x”))  
 count++;

i++;

}

Stole the following from <https://beginnersbook.com/2013/12/java-strings/>

1. [**char charAt(int index)**](https://beginnersbook.com/2013/12/java-string-charat-method-example/): It returns the character at the specified index. Specified index value should be between 0 to length() -1 both inclusive. It throws IndexOutOfBoundsException if index<0||>= length of String.
2. [**boolean equals(Object obj)**](https://beginnersbook.com/2013/12/java-string-equals-and-equalsignorecase-methods-example/)**:** Compares the string with the specified string and returns true if both matches else false.
3. [**boolean equalsIgnoreCase(String string)**](https://beginnersbook.com/2013/12/java-string-equals-and-equalsignorecase-methods-example/): It works same as equals method but it doesn’t consider the case while comparing strings. It does a case insensitive comparison.
4. [**int compareTo(String string)**](https://beginnersbook.com/2013/12/java-string-compareto-method-example/)**:** This method compares the two strings based on the Unicode value of each character in the strings.
5. [**int compareToIgnoreCase(String string)**](https://beginnersbook.com/2013/12/java-string-comparetoignorecase-method-example/)**:** Same as CompareTo method however it ignores the case during comparison.
6. [**boolean startsWith(String prefix, int offset)**](https://beginnersbook.com/2013/12/java-string-startswith-method-example/)**:** It checks whether the substring (starting from the specified offset index) is having the specified prefix or not.
7. [**boolean startsWith(String prefix)**](https://beginnersbook.com/2013/12/java-string-startswith-method-example/)**:** It tests whether the string is having specified prefix, if yes then it returns true else false.
8. [**boolean endsWith(String suffix)**](https://beginnersbook.com/2013/12/java-string-endswith-method-example/)**:** Checks whether the string ends with the specified suffix.
9. [**int hashCode()**](https://beginnersbook.com/2013/12/java-string-trim-and-hashcode-methods/)**:** It returns the hash code of the string.
10. [**int indexOf(int ch)**](https://beginnersbook.com/2013/12/java-string-indexof-method-example/)**:** Returns the index of first occurrence of the specified character ch in the string.
11. [**int indexOf(int ch, int fromIndex)**](https://beginnersbook.com/2013/12/java-string-indexof-method-example/)**:** Same as indexOf method however it starts searching in the string from the specified fromIndex.
12. [**int lastIndexOf(int ch)**](https://beginnersbook.com/2013/12/java-string-lastindexof-method-example/)**:** It returns the last occurrence of the character ch in the string.
13. [**int lastIndexOf(int ch, int fromIndex)**](https://beginnersbook.com/2013/12/java-string-lastindexof-method-example/)**:** Same as lastIndexOf(int ch) method, it starts search from fromIndex.
14. [**int indexOf(String str)**](https://beginnersbook.com/2013/12/java-string-indexof-method-example/)**:** This method returns the index of first occurrence of specified substring str.
15. [**int lastindexOf(String str)**](https://beginnersbook.com/2013/12/java-string-lastindexof-method-example/): Returns the index of last occurrence of string str.
16. [**String substring(int beginIndex)**](https://beginnersbook.com/2013/12/java-string-substring-method-example/)**:** It returns the substring of the string. The substring starts with the character at the specified index.
17. [**String substring(int beginIndex, int endIndex)**](https://beginnersbook.com/2013/12/java-string-substring-method-example/)**:** Returns the substring. The substring starts with character at beginIndex and ends with the character at endIndex.
18. [**String concat(String str)**](https://beginnersbook.com/2013/12/java-string-concat-method-example/)**:** Concatenates the specified string “str” at the end of the string.
19. [**String replace(char oldChar, char newChar)**](https://beginnersbook.com/2013/12/java-string-replace-replacefirst-replaceall-method-examples/)**:** It returns the new updated string after changing all the occurrences of oldChar with the newChar.
20. [**boolean contains(CharSequence s)**](https://beginnersbook.com/2017/10/java-string-contains-method/)**:** It checks whether the string contains the specified sequence of char values. If yes then it returns true else false. It throws NullPointerException of ‘s’ is null.
21. [**String toUpperCase(Locale locale)**](https://beginnersbook.com/2013/12/java-string-tolowercase-method-example/)**:** Converts the string to upper case string using the rules defined by specified locale.
22. [**String toUpperCase()**](https://beginnersbook.com/2013/12/java-string-tolowercase-method-example/)**: Equivalent** to toUpperCase(Locale.getDefault()).
23. [**public String intern()**](https://beginnersbook.com/2017/10/java-string-intern-method/)**:** This method searches the specified string in the memory pool and if it is found then it returns the reference of it, else it allocates the memory space to the specified string and assign the reference to it.
24. [**public boolean isEmpty()**](https://beginnersbook.com/2017/10/java-string-isempty-method-with-example/)**:** This method returns true if the given string has 0 length. If the length of the specified Java String is non-zero then it returns false.
25. [**public static String join()**](https://beginnersbook.com/2017/10/java-string-join-method/)**:** This method joins the given strings using the specified delimiter and returns the concatenated Java String
26. [**String replaceFirst(String regex, String replacement)**](https://beginnersbook.com/2013/12/java-string-replace-replacefirst-replaceall-method-examples/)**:** It replaces the first occurrence of substring that fits the given regular expression “regex” with the specified replacement string.
27. [**String replaceAll(String regex, String replacement)**](https://beginnersbook.com/2013/12/java-string-replace-replacefirst-replaceall-method-examples/)**:** It replaces all the occurrences of substrings that fits the [**regular expression regex**](https://beginnersbook.com/2014/08/java-regex-tutorial/) with the replacement string.
28. [**String[ ] split(String regex, int limit)**](https://beginnersbook.com/2013/12/java-string-split-method-example/)**:** It splits the string and returns the array of substrings that matches the given regular expression. limit is a result threshold here.
29. [**String[ ] split(String regex)**](https://beginnersbook.com/2013/12/java-string-split-method-example/)**:** Same as split(String regex, int limit) method however it does not have any threshold limit.
30. [**String toLowerCase(Locale locale)**](https://beginnersbook.com/2013/12/java-string-tolowercase-method-example/)**:** It converts the string to lower case string using the rules defined by given locale.
31. [**public static String format()**](https://beginnersbook.com/2017/10/java-string-format-method/)**:** This method returns a formatted java String
32. [**String toLowerCase()**](https://beginnersbook.com/2013/12/java-string-tolowercase-method-example/)**:** Equivalent to toLowerCase(Locale. getDefault()).
33. [**String trim()**](https://beginnersbook.com/2013/12/java-string-trim-and-hashcode-methods/)**:** Returns the substring after omitting leading and trailing white spaces from the original string.
34. [**char[ ] toCharArray()**](https://beginnersbook.com/2013/12/java-string-tochararray-method-example/)**:** Converts the string to a character array.
35. [**static String copyValueOf(char[ ] data)**](https://beginnersbook.com/2013/12/java-string-copyvalueof-method-example/)**:** It returns a string that contains the characters of the specified character array.
36. [**static String copyValueOf(char[ ] data, int offset, int count)**](https://beginnersbook.com/2013/12/java-string-copyvalueof-method-example/)**:** Same as above method with two extra arguments – initial offset of subarray and length of subarray.
37. [**void getChars(int srcBegin, int srcEnd, char[ ] dest, int destBegin)**](https://beginnersbook.com/2013/12/java-string-getchars-method-example/)**:** It copies the characters of **src**array to the **dest** array. Only the specified range is being copied(srcBegin to srcEnd) to the dest subarray(starting fromdestBegin).
38. [**static String valueOf()**](https://beginnersbook.com/2017/10/java-string-valueof-method/)**:** This method returns a string representation of passed arguments such as int, long, float, double, char and char array.
39. [**boolean contentEquals(StringBuffer sb)**](https://beginnersbook.com/2013/12/java-string-contentequals-method-example/)**:** It compares the string to the specified string buffer.
40. [**boolean regionMatches(int srcoffset, String dest, int destoffset, int len)**](https://beginnersbook.com/2013/12/java-string-regionmatches-method-example/)**:** It compares the substring of input to the substring of specified string.
41. [**boolean regionMatches(boolean ignoreCase, int srcoffset, String dest, int destoffset, int len)**](https://beginnersbook.com/2013/12/java-string-regionmatches-method-example/)**:** Another variation of regionMatches method with the extra boolean argument to specify whether the comparison is case sensitive or case insensitive.
42. [**byte[ ] getBytes(String charsetName)**](https://beginnersbook.com/2013/12/java-string-getbytes-method-example/)**:** It converts the String into sequence of bytes using the specified charset encoding and returns the array of resulted bytes.
43. [**byte[ ] getBytes()**](https://beginnersbook.com/2013/12/java-string-getbytes-method-example/)**:** This method is similar to the above method it just uses the default charset encoding for converting the string into sequence of bytes.
44. [**int length()**](https://beginnersbook.com/2013/12/java-string-length-method-example/)**:** It returns the length of a String.
45. [**boolean matches(String regex)**](https://beginnersbook.com/2013/12/java-string-matches-method-example/)**:** It checks whether the String is matching with the specified [**regular expression**](http://docs.oracle.com/javase/7/docs/api/java/util/regex/Pattern.html#sum) regex.
46. **int codePointAt(int index):**It is similar to the charAt method however it returns the Unicode code point value of specified index rather than the character itself.

HyperText Markup Language: Created by Tim Berners-Lee  
  
This is a file format consisting of tags, which surround different types of information.  
Here are some common tags and an example .html file

Jake’s Magical Web Sheet

<!DOCTYPE html>

<html>

<body>

<title> This is just a title </title>

<style type="text/css">

h1 {color:red;}

p {color:green;}

</style><!-- css code -->

<h1>This is the heading you can have from 1 to 6</h1>

<p style="text-align:center">

<strike><strong><i>This text has a line through it is italicized and strong!</i></strong></strike>

<br>This is just normal text

</p>

<img src="http://cdn.animals-zone.com/wp-content/uploads/2012/08/seal.jpg" alt="baby seal">

</br>

<a href="http://www.w3schools.com"> This is a link that goes to the best website ever w3schools.com</a>

<table border="2">

<tr>

<th>These are </th>

<th>headings</th>

</tr>

<tr>

<td>These are </td>

<td>data places</td>

</tr>

</table>

</body>

</html>

