<https://beginnersbook.com/2013/12/java-strings/>

https://beginnersbook.com/2013/12/java-string-indexof-method-example/

**Java – String Class and its methods explained with examples**

BY CHAITANYA SINGH | FILED UNDER: [STRING HANDLING](https://beginnersbook.com/category/string-handling/)

**String** is a sequence of characters, for e.g. “Hello” is a string of 5 characters. In java, string is an immutable object which means it is constant and can cannot be changed once it has been created. In this tutorial we will learn about String class and String methods in detail along with many other Java String tutorials.

**Creating a String**

There are two ways to create a String in Java

1. String literal
2. Using new keyword

**String literal**

In java, Strings can be created like this: Assigning a String literal to a String instance:

String str1 = "Welcome";

String str2 = "Welcome";

**The problem with this approach**: As I stated in the beginning that String is an object in Java. However we have not created any string object using new keyword above. The compiler does that task for us it creates a string object having the string literal (that we have provided , in this case it is “Welcome”) and assigns it to the provided string instances.

**But** if the object already exist in the memory it does not create a new Object rather it assigns the same old object to the new instance, that means even though we have two string instances above(str1 and str2) compiler only created on string object (having the value “Welcome”) and assigned the same to both the instances. For example there are 10 string instances that have same value, it means that in memory there is only one object having the value and all the 10 string instances would be pointing to the same object.

What if we want to have two different object with the same string? For that we would need to create strings using **new keyword**.

**Using New Keyword**

As we saw above that when we tried to assign the same string object to two different literals, compiler only created one object and made both of the literals to point the same object. To overcome that approach we can create strings like this:

String str1 = new String("Welcome");

String str2 = new String("Welcome");

In this case compiler would create two different object in memory having the same string.

**A Simple Java String Example**

public class Example{

public static void main(String args[]){

//creating a string by java string literal

String str = "Beginnersbook";

char arrch[]={'h','e','l','l','o'};

//converting char array arrch[] to string str2

String str2 = new String(arrch);

//creating another java string str3 by using new keyword

String str3 = new String("Java String Example");

//Displaying all the three strings

System.out.println(str);

System.out.println(str2);

System.out.println(str3);

}

}

Output: 6003819535 9518215121

Beginnersbook

hello

Java String Example

**Java String Methods**

Here are the list of the methods available in the Java String class. These methods are explained in the separate tutorials with the help of examples. Links to the tutorials are provided below:

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.

# Java String charAt() Method example

By Chaitanya Singh | Filed Under: [String handling](https://beginnersbook.com/category/string-handling/)

The **Java String charAt(int index)** method returns the character at the specified index in a string. The index value that we pass in this method should be between 0 and (length of string-1). For example: s.charAt(0) would return the first character of the string represented by instance s. Java String charAt method throws [IndexOutOfBoundsException](https://docs.oracle.com/javase/7/docs/api/java/lang/IndexOutOfBoundsException.html), if the index value passed in the charAt() method is less than zero or greater than or equal to the length of the string (index<0|| index>=length()).

## Java String charAt() Method example

Lets take an example to understand the use of charAt() method. In this example we have a string and we are printing the 1st, 6th, 12th and 21st character of the string using charAt() method.

public class CharAtExample {

public static void main(String args[]) {

String str = "Welcome to string handling tutorial";

//This will return the first char of the string

char ch1 = str.charAt(0);

//This will return the 6th char of the string

char ch2 = str.charAt(5);

//This will return the 12th char of the string

char ch3 = str.charAt(11);

//This will return the 21st char of the string

char ch4 = str.charAt(20);

System.out.println("Character at 0 index is: "+ch1);

System.out.println("Character at 5th index is: "+ch2);

System.out.println("Character at 11th index is: "+ch3);

System.out.println("Character at 20th index is: "+ch4);

}

}

**Output:**

Character at 0 index is: W

Character at 5th index is: m

Character at 11th index is: s

Character at 20th index is: n

## IndexOutOfBoundsException while using charAt() method

When we pass negative index or the index which is greater than length()-1 then the charAt() method throws IndexOutOfBoundsException. In the following example we are passing negative index in the charAt() method, lets see what we get in the output.

public class JavaExample {

public static void main(String args[]) {

String str = "BeginnersBook";

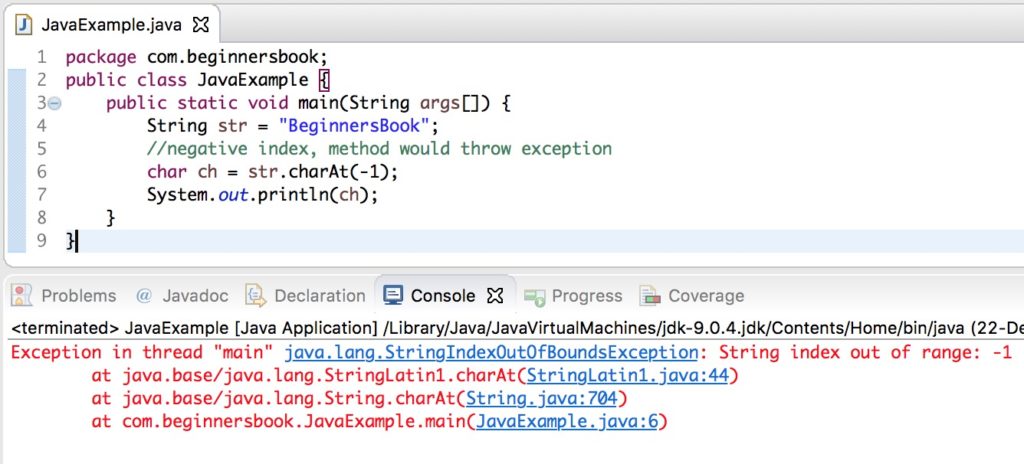
//negative index, method would throw exception

char ch = str.charAt(-1);

System.out.println(ch);

}

}

**Output:**  


## Java String charAt() example to print all characters of string

To print all the characters of a string, we are running a [for loop](https://beginnersbook.com/2015/03/for-loop-in-java-with-example/) from 0 to length of string – 1 and displaying the character at each iteration of the loop using the charAt() method.

public class JavaExample {

public static void main(String args[]) {

String str = "BeginnersBook";

for(int i=0; i<=str.length()-1; i++) {

System.out.println(str.charAt(i));

}

}

}

Output:

B

e

g

i

n

n

e

r

s

B

o

o

k

## Java String charAt() example to count the occurrence of a character

In this example, we will use the charAt() method to count the occurrence of a particular character in the given string. Here we have a string and we are counting the occurrence of character ‘B’ in the string.

public class JavaExample {

public static void main(String[] args) {

String str = "BeginnersBook";

//initialized the counter to 0

int counter = 0;

for (int i=0; i<=str.length()-1; i++) {

if(str.charAt(i) == 'B') {

//increasing the counter value at each occurrence of 'B'

counter++;

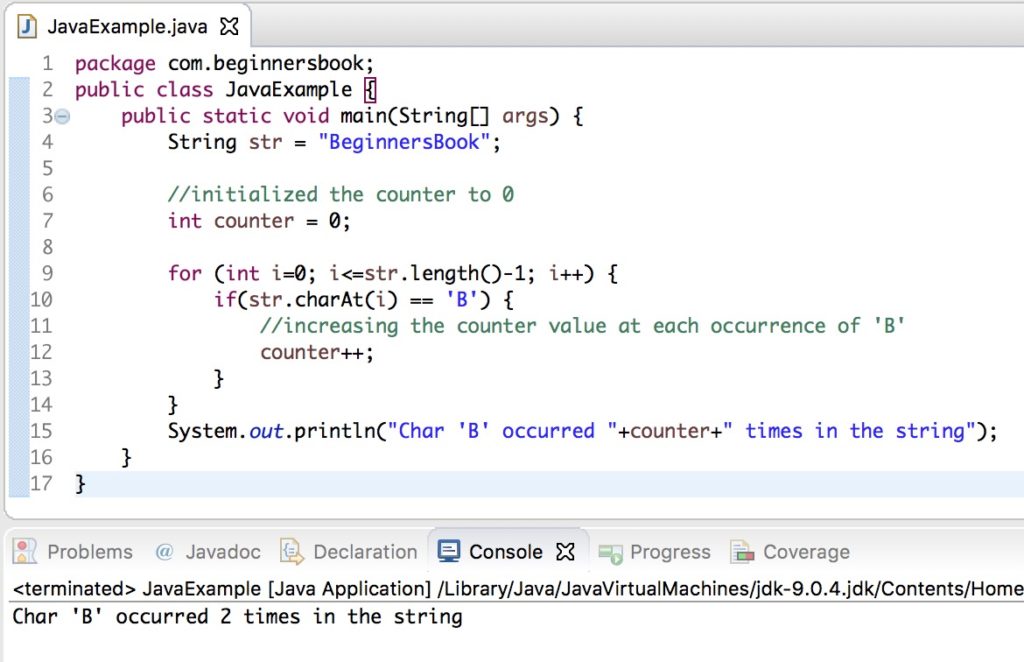
}

}

System.out.println("Char 'B' occurred "+counter+" times in the string");

}

}

**Output:**  


1. [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.

# Java String equals() and equalsIgnoreCase() Methods example

By Chaitanya Singh | Filed Under: [String handling](https://beginnersbook.com/category/string-handling/)

In this tutorial we will discuss equals() and equalsIgnoreCase() methods. Both of these methods are used for comparing two strings. The only difference between them is that the equals() methods considers the case while equalsIgnoreCase() methods ignores the case during comparison. For e.g. The equals() method would return false if we compare the strings “TEXT” and “text” however equalsIgnoreCase() would return true.

boolean equals(String str): Case sensitive comparison  
boolean equalsIgnoreCase(String str): Case in-sensitive comparison

## Java String equals() method example

In this example we will see how equals() method works in different scenarios. We can compare two String instances (str1, str2, str3) using equals() method like we did in the following example or we can also compare string instances with the hardcoded strings passed as an argument to the equals() method as shown in the example below.

As you can observe in the output that when we compared the String str1 (value “Hello”) with the string “hello”, the equals() method returned false because this method case sensitive and considers the case while comparing strings. On the other hand the equalsIgnoreCase() method compares strings while ignoring their cases, which we will see in the next section.

public class JavaExample{

public static void main(String args[]){

String str1= new String("Hello");

String str2= new String("Hi");

String str3= new String("Hello");

System.out.println("str1 equals to str2:"+str1.equals(str2));

System.out.println("str1 equals to str3:"+str1.equals(str3));

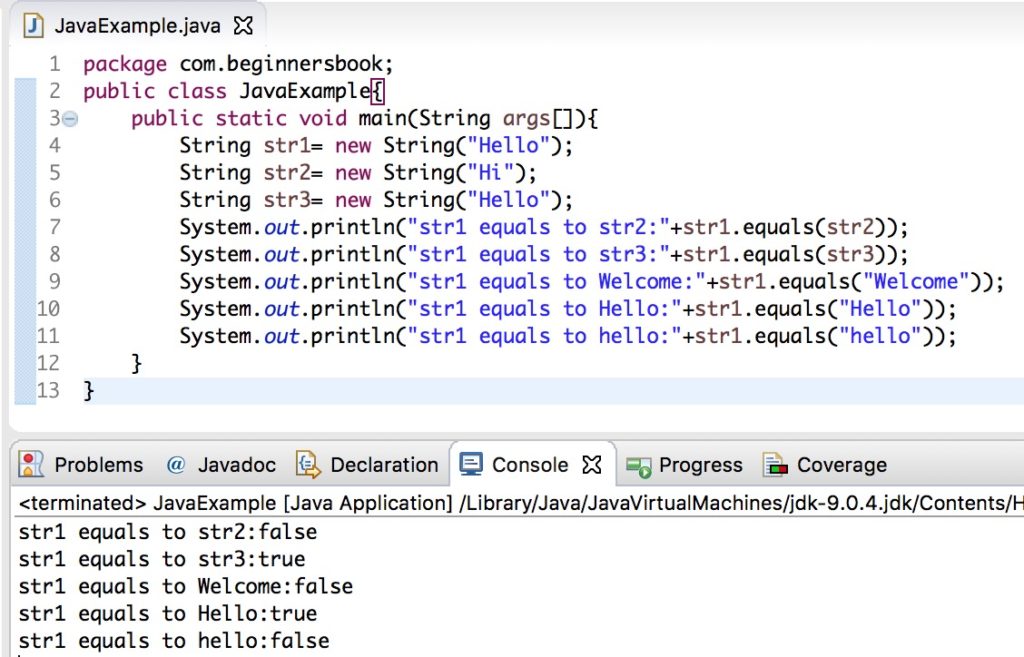
System.out.println("str1 equals to Welcome:"+str1.equals("Welcome"));

System.out.println("str1 equals to Hello:"+str1.equals("Hello"));

System.out.println("str1 equals to hello:"+str1.equals("hello"));

}

}

**Output:**  


## Java String equalsIgnoreCase() method example

The method equalsIgnoreCase() ignores the case while comparing two strings. In the following example we compared the string “Apple” with the string “APPLE” and it returned true.

public class JavaExample{

public static void main(String args[]){

String str1= new String("Apple");

String str2= new String("MANGO");

String str3= new String("APPLE");

System.out.println("str1 equals to str2:"+str1.equalsIgnoreCase(str2));

System.out.println("str1 equals to str3:"+str1.equalsIgnoreCase(str3));

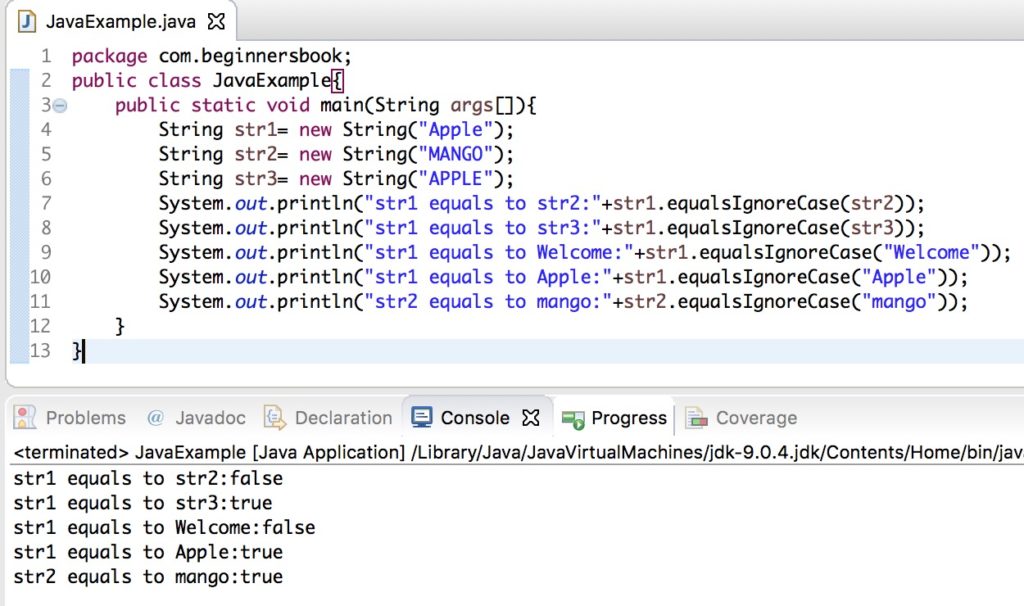
System.out.println("str1 equals to Welcome:"+str1.equalsIgnoreCase("Welcome"));

System.out.println("str1 equals to Apple:"+str1.equalsIgnoreCase("Apple"));

System.out.println("str2 equals to mango:"+str2.equalsIgnoreCase("mango"));

}

}

**Output:**  


1. [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.

# Java String equals() and equalsIgnoreCase() Methods example

By Chaitanya Singh | Filed Under: [String handling](https://beginnersbook.com/category/string-handling/)

In this tutorial we will discuss equals() and equalsIgnoreCase() methods. Both of these methods are used for comparing two strings. The only difference between them is that the equals() methods considers the case while equalsIgnoreCase() methods ignores the case during comparison. For e.g. The equals() method would return false if we compare the strings “TEXT” and “text” however equalsIgnoreCase() would return true.

boolean equals(String str): Case sensitive comparison  
boolean equalsIgnoreCase(String str): Case in-sensitive comparison

## Java String equals() method example

In this example we will see how equals() method works in different scenarios. We can compare two String instances (str1, str2, str3) using equals() method like we did in the following example or we can also compare string instances with the hardcoded strings passed as an argument to the equals() method as shown in the example below.

As you can observe in the output that when we compared the String str1 (value “Hello”) with the string “hello”, the equals() method returned false because this method case sensitive and considers the case while comparing strings. On the other hand the equalsIgnoreCase() method compares strings while ignoring their cases, which we will see in the next section.

public class JavaExample{

public static void main(String args[]){

String str1= new String("Hello");

String str2= new String("Hi");

String str3= new String("Hello");

System.out.println("str1 equals to str2:"+str1.equals(str2));

System.out.println("str1 equals to str3:"+str1.equals(str3));

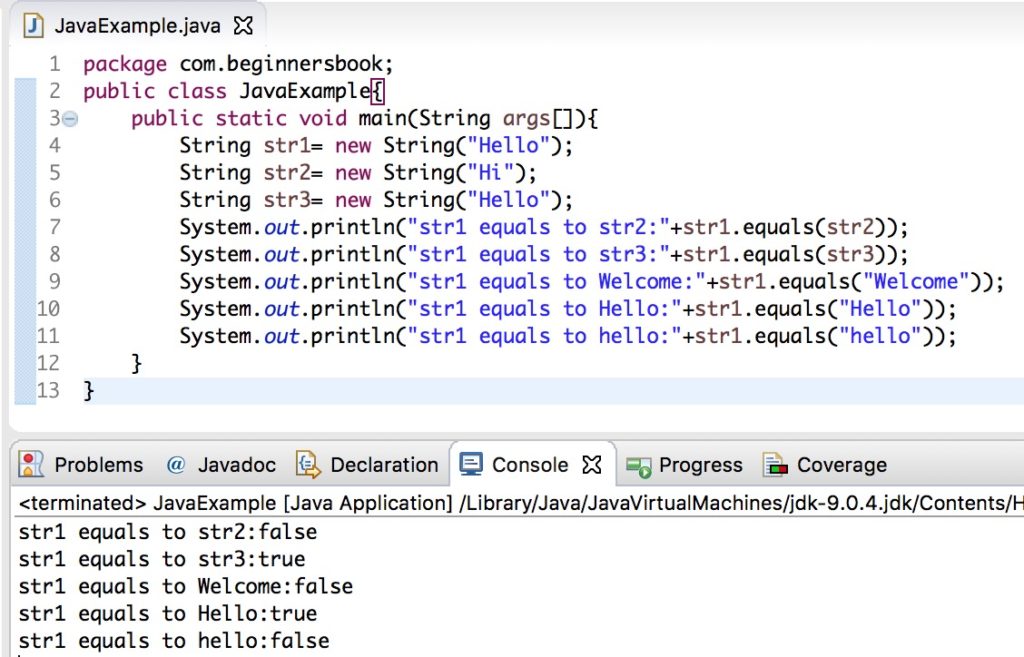
System.out.println("str1 equals to Welcome:"+str1.equals("Welcome"));

System.out.println("str1 equals to Hello:"+str1.equals("Hello"));

System.out.println("str1 equals to hello:"+str1.equals("hello"));

}

}

**Output:**  


## Java String equalsIgnoreCase() method example

The method equalsIgnoreCase() ignores the case while comparing two strings. In the following example we compared the string “Apple” with the string “APPLE” and it returned true.

public class JavaExample{

public static void main(String args[]){

String str1= new String("Apple");

String str2= new String("MANGO");

String str3= new String("APPLE");

System.out.println("str1 equals to str2:"+str1.equalsIgnoreCase(str2));

System.out.println("str1 equals to str3:"+str1.equalsIgnoreCase(str3));

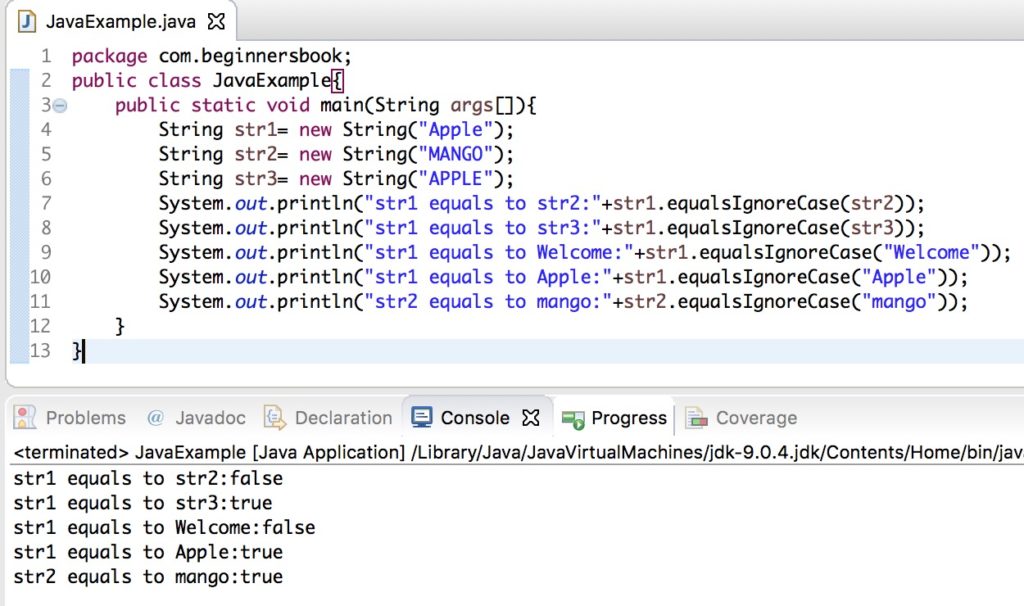
System.out.println("str1 equals to Welcome:"+str1.equalsIgnoreCase("Welcome"));

System.out.println("str1 equals to Apple:"+str1.equalsIgnoreCase("Apple"));

System.out.println("str2 equals to mango:"+str2.equalsIgnoreCase("mango"));

}

}

**Output:**  


# Java String indexOf() Method with example

By Chaitanya Singh | Filed Under: [String handling](https://beginnersbook.com/category/string-handling/)

**Java String indexOf() method** is used to find the index of a specified character or a substring in a given String. There are 4 variations of this method in [String class](https://beginnersbook.com/2013/12/java-strings/):

## The indexOf() method signature

int indexOf(int ch): It returns the index of the first occurrence of character ch in a given String.

int indexOf(int ch, int fromIndex): It returns the index of first occurrence of character ch in the given string after the specified index “fromIndex”. For example, if the indexOf() method is called like this str.indexOf(‘A’, 20) then it would start looking for the character ‘A’ in string str after the index 20.

int indexOf(String str): Returns the index of string str in a particular String.

int indexOf(String str, int fromIndex): Returns the index of string str in the given string after the specified index “fromIndex”.

All the above variations **returns -1** if the specified char/substring is not found in the particular String.

## Java String indexOf() Method example

public class IndexOfExample{

public static void main(String args[]) {

String str1 = new String("This is a BeginnersBook tutorial");

String str2 = new String("Beginners");

String str3 = new String("Book");

String str4 = new String("Books");

System.out.println("Index of B in str1: "+str1.indexOf('B'));

System.out.println("Index of B in str1 after 15th char:"+str1.indexOf('B', 15));

System.out.println("Index of B in str1 after 30th char:"+str1.indexOf('B', 30));

System.out.println("Index of string str2 in str1:"+str1.indexOf(str2));

System.out.println("Index of str2 after 15th char"+str1.indexOf(str2, 15));

System.out.println("Index of string str3:"+str1.indexOf(str3));

System.out.println("Index of string str4"+str1.indexOf(str4));

System.out.println("Index of hardcoded string:"+str1.indexOf("is"));

System.out.println("Index of hardcoded string after 4th char:"+str1.indexOf("is", 4));

}

}

**Output:**

Index of B in str1: 10

Index of B in str1 after 15th char:19

Index of B in str1 after 30th char:-1

Index of string str2 in str1:10

Index of str2 after 15th char-1

Index of string str3:19

Index of string str4-1

Index of hardcoded string:2

Index of hardcoded string after 4th char:5

## Another example of indexOf() method

Lets take a simple example with a short string where we are finding the indexes of given chars and [substring](https://beginnersbook.com/2013/12/java-string-substring-method-example/) using the indexOf() method.

public class JavaExample {

public static void main(String[] args) {

String str = "Java String";

char ch = 'J';

char ch2 = 'S';

String subStr = "tri";

int posOfJ = str.indexOf(ch);

int posOfS = str.indexOf(ch2);

int posOfSubstr = str.indexOf(subStr);

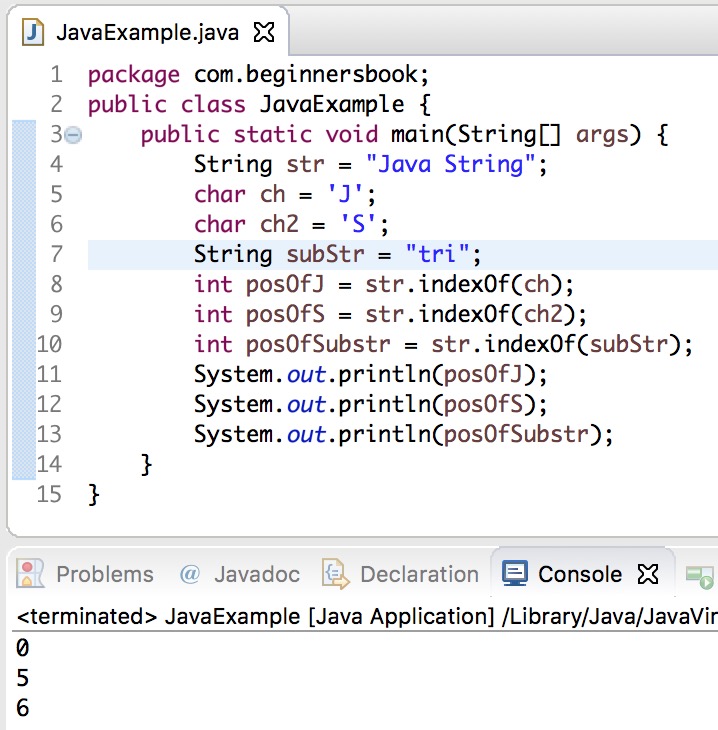
System.out.println(posOfJ);

System.out.println(posOfS);

System.out.println(posOfSubstr);

}

}

**Output:**  


============================================

1. [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.
2. [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.
3. [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.
4. [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.
5. [boolean endsWith(String suffix)](https://beginnersbook.com/2013/12/java-string-endswith-method-example/): Checks whether the string ends with the specified suffix.
6. [int hashCode()](https://beginnersbook.com/2013/12/java-string-trim-and-hashcode-methods/): It returns the hash code of the string.
7. [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.
8. [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.
9. [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.
10. [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.
11. [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.
12. [int lastindexOf(String str)](https://beginnersbook.com/2013/12/java-string-lastindexof-method-example/): Returns the index of last occurrence of string str.
13. [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.
14. [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.
15. [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.
16. [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.
17. [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.
18. [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.
19. [String toUpperCase()](https://beginnersbook.com/2013/12/java-string-tolowercase-method-example/): Equivalent to toUpperCase(Locale.getDefault()).
20. [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.
21. [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.
22. [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
23. [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.
24. [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.
25. [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.
26. [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.
27. [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.
28. [public static String format()](https://beginnersbook.com/2017/10/java-string-format-method/): This method returns a formatted java String
29. [String toLowerCase()](https://beginnersbook.com/2013/12/java-string-tolowercase-method-example/): Equivalent to toLowerCase(Locale. getDefault()).
30. [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.
31. [char[] toCharArray()](https://beginnersbook.com/2013/12/java-string-tochararray-method-example/): Converts the string to a character array.
32. [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.
33. [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.
34. [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).
35. [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.
36. [boolean contentEquals(StringBuffer sb)](https://beginnersbook.com/2013/12/java-string-contentequals-method-example/): It compares the string to the specified string buffer.
37. [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.
38. [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.
39. [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.
40. [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.
41. [int length()](https://beginnersbook.com/2013/12/java-string-length-method-example/): It returns the length of a String.
42. [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](https://docs.oracle.com/javase/7/docs/api/java/util/regex/Pattern.html#sum) regex.
43. 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.
44. A Java both int and Integer are used to store integer type data the major difference between both is type of int is primitive while Integer is of class type.This difference become significant when concept of OOPs comes in picture during development as int follows the principle of primitive data type while Integer behave as a wrapper class.
45. Following are the important differences between int and Integer.

| **Sr. No.** | **Key** | **int** | **Integer** |
| --- | --- | --- | --- |
| 1 | Type | A int is a data type that stores 32 bit signed two's compliment integer. | On other hand Integer is a wrapper class which wraps a primitive type int into an object. |
| 2 | Purpose | int helps in storing integer value into memory. | Integer helps in converting int into object and to convert an object into int as per requirement. |
| 3 | Flexibility | int provides less flexibility as compare to Integer as it only allows binary value of an integer in it. | Integer on other hand is more flexible in storing and manipulating an int data.Since Wrapper classes inherit Object class, they can be used in collections with Object reference or generics. |
| 4 | Memory allocation | As already mentioned int is a primitive data type and takes 32 bits(4 bytes) to store. | On other hand Integer is an object which takes 128 bits (16 bytes) to store its int value. |
| 5 | Casting | In java one canâTMt assign a string value (containing an integer only) to an int variable directly or even by casting. | In case of Integer we can assign string to an object of Integer type using the Integer(String) constructor or by even use parseInt(String) to convert a String literal to an int value. |
| 6 | Direct Conversion to Other base. | In case of int we can't convert its integer value to other base. | However in Integer we can directly convert its integer value to other bases such as Binary, Octal or Hexadecimal format using toBinaryString(), toOctalString() or toHexString() respectively. |
| 7 | Allowed operations | int do not allowed any of inbuilt functions to change its value or syntax. | However in Integer we can reverse number or rotate it left or right using reverse(), rotateLeft() and rotateRight() respectively. |

## Example of int vs Integer

1. **JavaTester.java**

## Example

1. [Live Demo](http://tpcg.io/FibrBR9X)
2. public class JavaTester {
3. public static void main(String args[]){
4. Integer a = new Integer("456");
5. // Casting not possible
6. // int a = (int)"456";
7. // Casting not possible
8. // int c="456";
9. // Casting possible using methods
10. // from Integer Wrapper class
11. int b = Integer.parseInt("456");
12. System.out.print(b);
13. }
14. }

## Output

1. 456