Introduction to java

COMP1030 Lecture #10

Housekeeping

- Our goal - 100% pass rate in this course.
- Review the lecture slides ahead of time.
- Review the lecture slides after class with a study group.
- Repeat the lab at home 1-2 times.
- Take notes for each lecture.
- Attend live stream tutorials every Monday night at 7pm via webex.

The four secrets of success

Housekeeping

- Assignment #1 Answer key is posted under the assignments link
- Midterm-Exam question paper and answer key are posted under course information link
- Labs will no longer be checked as homework
- This is the last lecture of the course, next week you will be writing assignment #2.

Review

- toString
- Big “O” Object
- System.out.println
- String class methods

Characters

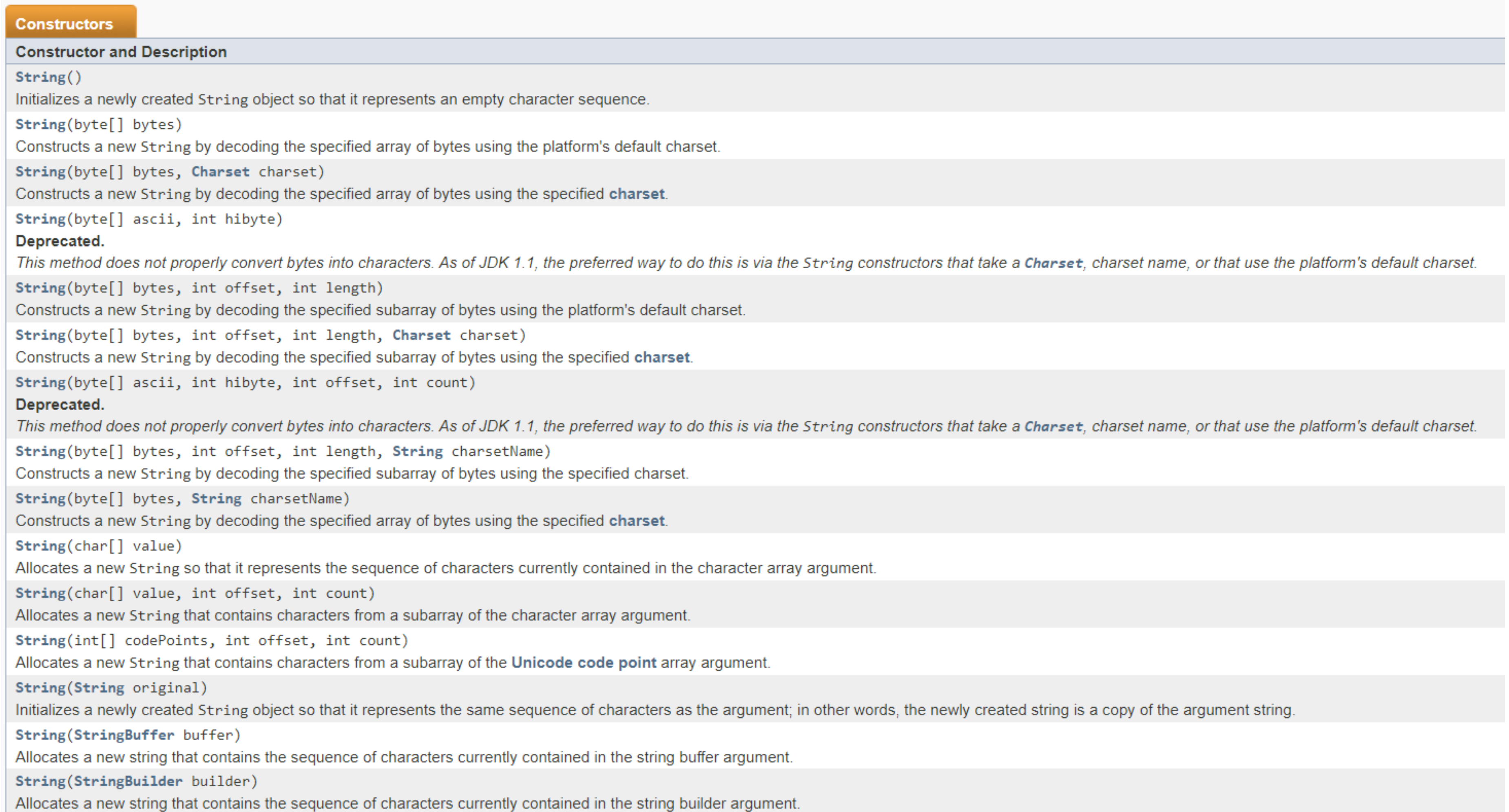
- A character literal is an integer value represented as a character in single quotes. The value of the character literal is the integer value of the character in the Unicode character set.

String class

- A String is a sequence of characters.
- A String can include letters, digits and various special characters.

String class Constructors (partial list)

- The String class has several constructors:



Constructors

Constructor and Description
`String()` Initializes a newly created `String` object so that it represents an empty character sequence.
`String(byte[] bytes)` Constructs a new `String` by decoding the specified array of bytes using the platform's default charset.
`String(byte[] bytes, Charset charset)` Constructs a new `String` by decoding the specified array of bytes using the specified **charset**.
`String(byte[] ascii, int hibyte)` **Deprecated.** *This method does not properly convert bytes into characters. As of JDK 1.1, the preferred way to do this is via the `String` constructors that take a **`Charset`**, charset name, or that use the platform's default charset.*
`String(byte[] bytes, int offset, int length)` Constructs a new `String` by decoding the specified subarray of bytes using the platform's default charset.
`String(byte[] bytes, int offset, int length, Charset charset)` Constructs a new `String` by decoding the specified subarray of bytes using the specified **charset**.
`String(byte[] ascii, int hibyte, int offset, int count)` **Deprecated.** *This method does not properly convert bytes into characters. As of JDK 1.1, the preferred way to do this is via the `String` constructors that take a **`Charset`**, charset name, or that use the platform's default charset.*
`String(byte[] bytes, int offset, int length, String charsetName)` Constructs a new `String` by decoding the specified subarray of bytes using the specified charset.
`String(byte[] bytes, String charsetName)` Constructs a new `String` by decoding the specified array of bytes using the specified **charset**.
`String(char[] value)` Allocates a new `String` so that it represents the sequence of characters currently contained in the character array argument.
`String(char[] value, int offset, int count)` Allocates a new `String` that contains characters from a subarray of the character array argument.
`String(int[] codePoints, int offset, int count)` Allocates a new `String` that contains characters from a subarray of the **Unicode code point** array argument.
`String(String original)` Initializes a newly created `String` object so that it represents the same sequence of characters as the argument; in other words, the newly created string is a copy of the argument string.
`String(StringBuffer buffer)` Allocates a new string that contains the sequence of characters currently contained in the string buffer argument.
`String(StringBuilder builder)` Allocates a new string that contains the sequence of characters currently contained in the string builder argument.

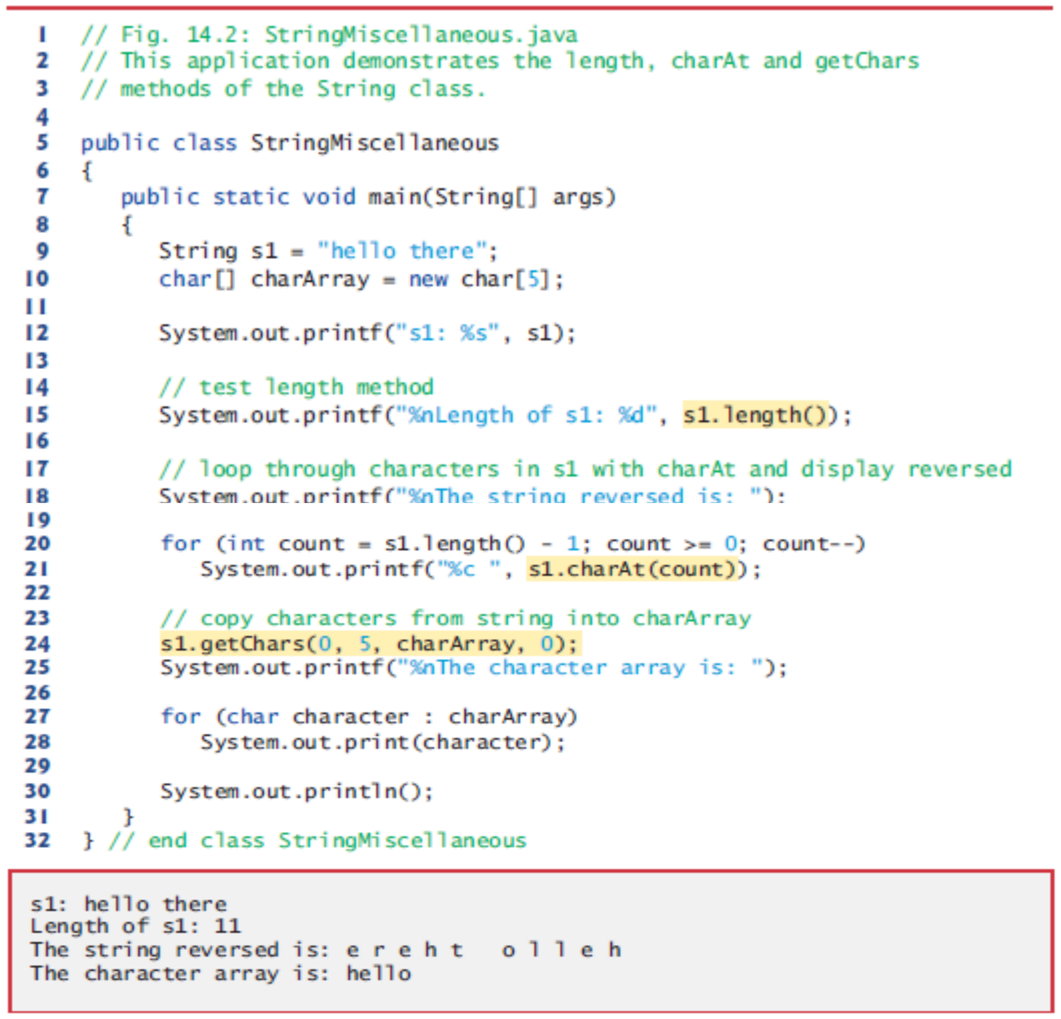
String class Constructors

```java
// Fig. 14.1: StringConstructors.java
// String class constructors.

public class StringConstructors
{
   public static void main(String[] args)
   {
      char[] charArray = {'b', 'i', 'r', 't', 'h', ' ', 'd', 'a', 'y'};
      String s = new String("hello");

      // use String constructors
      String s1 = new String();
      String s2 = new String(s);
      String s3 = new String(charArray);
      String s4 = new String(charArray, 6, 3);

      System.out.printf(
         "s1 = %s%ns2 = %s%ns3 = %s%ns4 = %s%n", s1, s2, s3, s4);
   }
} // end class StringConstructors
```

```
s1 =
s2 = hello
s3 = birth day
s4 = day
```

Fig. 14.1 | String class constructors.

String class methods

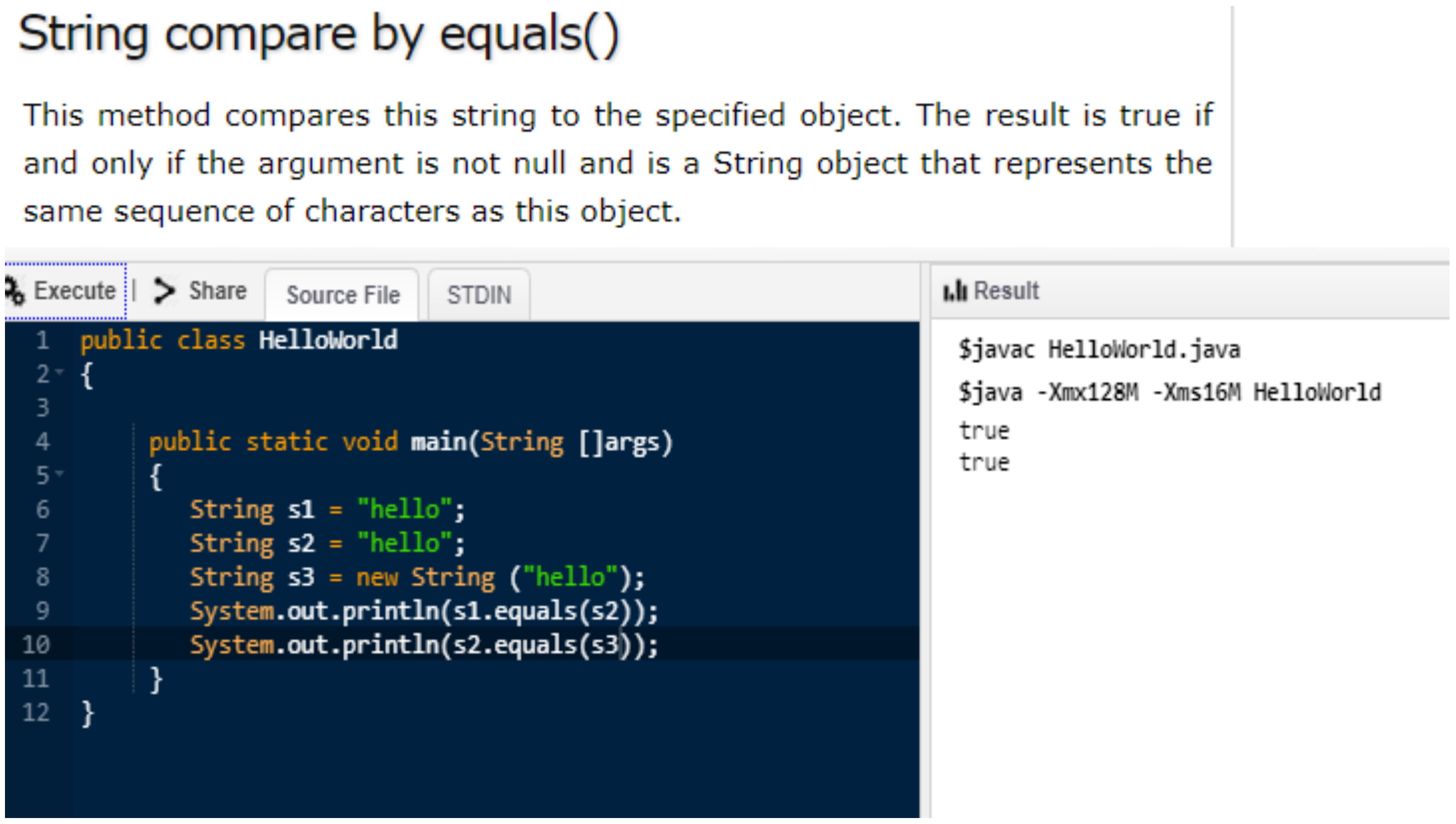


```java
// Fig. 14.2: StringMiscellaneous.java
// This application demonstrates the length, charAt and getChars
// methods of the String class.

public class StringMiscellaneous
{
   public static void main(String[] args)
   {
      String s1 = "hello there";
      char[] charArray = new char[5];

      System.out.printf("s1: %s", s1);

      // test length method
      System.out.printf("%nLength of s1: %d", s1.length());

      // loop through characters in s1 with charAt and display reversed
      System.out.printf("%nThe string reversed is: ");

      for (int count = s1.length() - 1; count >= 0; count--)
         System.out.printf("%c ", s1.charAt(count));

      // copy characters from string into charArray
      s1.getChars(0, 5, charArray, 0);
      System.out.printf("%nThe character array is: ");

      for (char character : charArray)
         System.out.print(character);

      System.out.println();
   }
} // end class StringMiscellaneous
```

```
s1: hello there
Length of s1: 11
The string reversed is: e r e h t  o l l e h
The character array is: hello
```

Fig. 14.2 | String methods length, charAt and getChars. (Part 2 of 2.)

String class – comparing Strings



String compare by equals()

This method compares this string to the specified object. The result is true if and only if the argument is not null and is a String object that represents the same sequence of characters as this object.

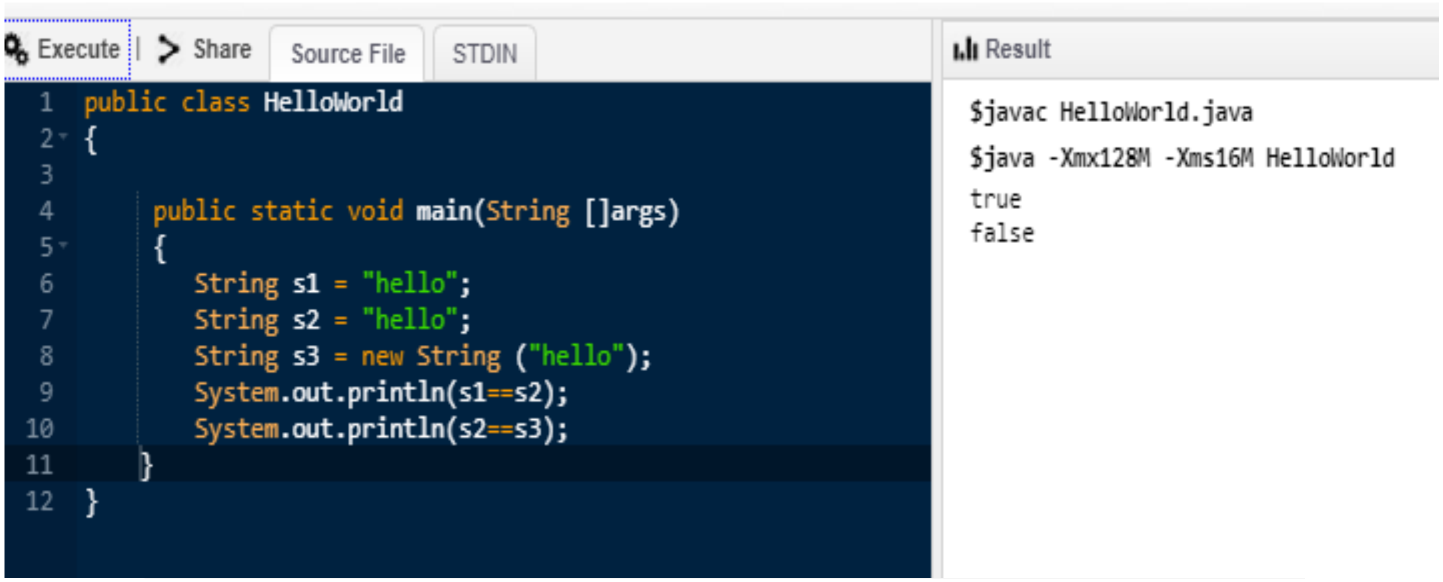
```java
public class HelloWorld
{

    public static void main(String []args)
    {
        String s1 = "hello";
        String s2 = "hello";
        String s3 = new String ("hello");
        System.out.println(s1.equals(s2));
        System.out.println(s2.equals(s3));
    }
}
```

```
$javac HelloWorld.java
$java -Xmx128M -Xms16M HelloWorld
true
true
```

Retrieved from: https://www.tutorialspoint.com/javaexamples/string_compare.htm

String class – comparing Strings

String compare by == operator

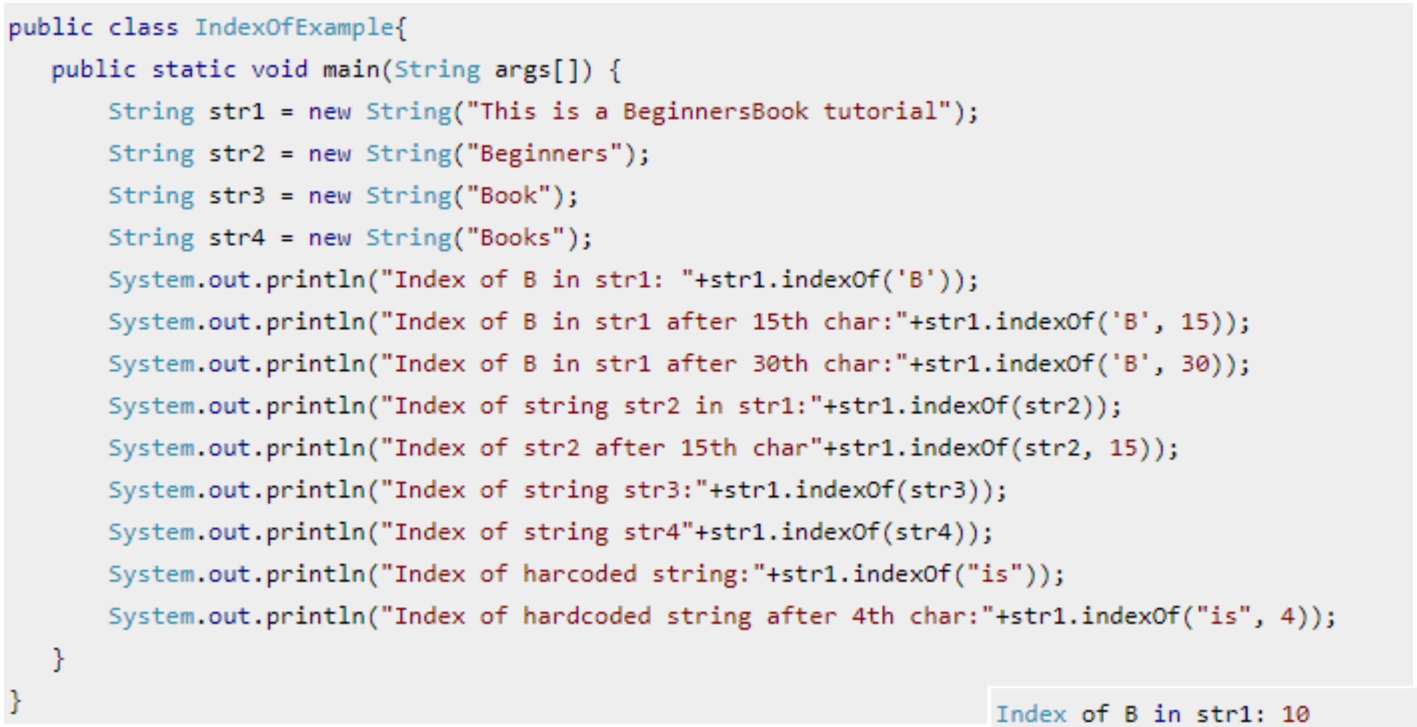


```java
public class HelloWorld
{

    public static void main(String []args)
    {
      String s1 = "hello";
      String s2 = "hello";
      String s3 = new String ("hello");
      System.out.println(s1==s2);
      System.out.println(s2==s3);
    }
}
```

```
$javac HelloWorld.java
$java -Xmx128M -Xms16M HelloWorld
true
false
```

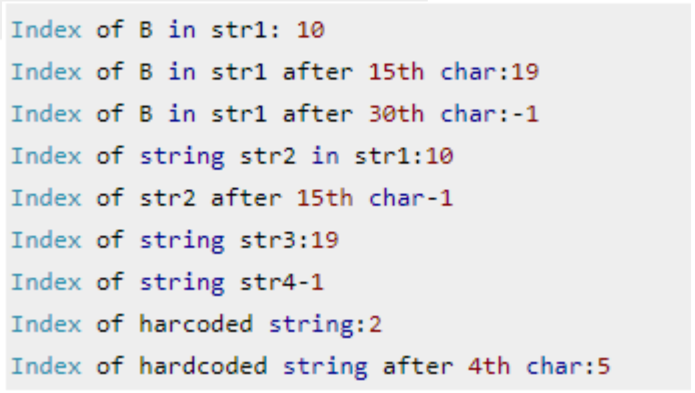
Retrieved from: https://www.tutorialspoint.com/javaexamples/string_compare.htm

String class – indexOf



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

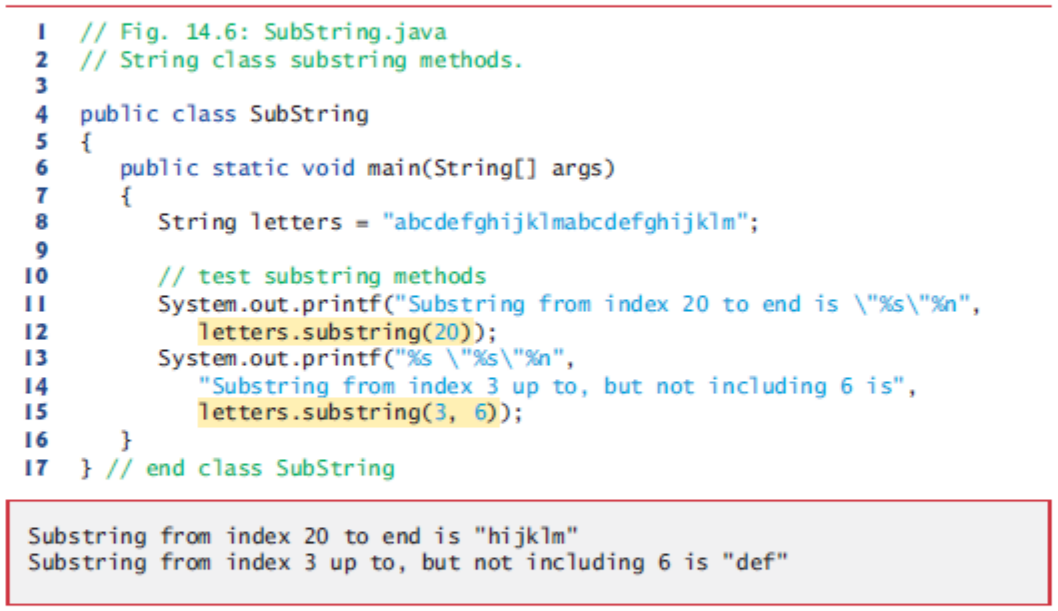
Output



```
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 harcoded string:2
Index of hardcoded string after 4th char:5
```

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

String class – substring



```java
// Fig. 14.6: SubString.java
// String class substring methods.

public class SubString
{
   public static void main(String[] args)
   {
      String letters = "abcdefghijklmabcdefghijklm";

      // test substring methods
      System.out.printf("Substring from index 20 to end is \"%s\"%n",
         letters.substring(20));
      System.out.printf("%s \"%s\"%n",
         "Substring from index 3 up to, but not including 6 is",
         letters.substring(3, 6));
   }
} // end class SubString
```

```
Substring from index 20 to end is "hijklm"
Substring from index 3 up to, but not including 6 is "def"
```

Fig. 14.6 | String class substring methods.

String class – concatenating strings

```java
// Fig. 14.7: StringConcatenation.java
// String method concat.

public class StringConcatenation
{
   public static void main(String[] args)
   {
      String s1 = "Happy ";
      String s2 = "Birthday";

      System.out.printf("s1 = %s%ns2 = %s%n%n",s1, s2);
      System.out.printf(
         "Result of s1.concat(s2) = %s%n", s1.concat(s2));
      System.out.printf("s1 after concatenation = %s%n", s1);
   }
} // end class StringConcatenation
```

```
s1 = Happy
s2 = Birthday

Result of s1.concat(s2) = Happy Birthday
s1 after concatenation = Happy
```

Fig. 14.7 | String method concat.

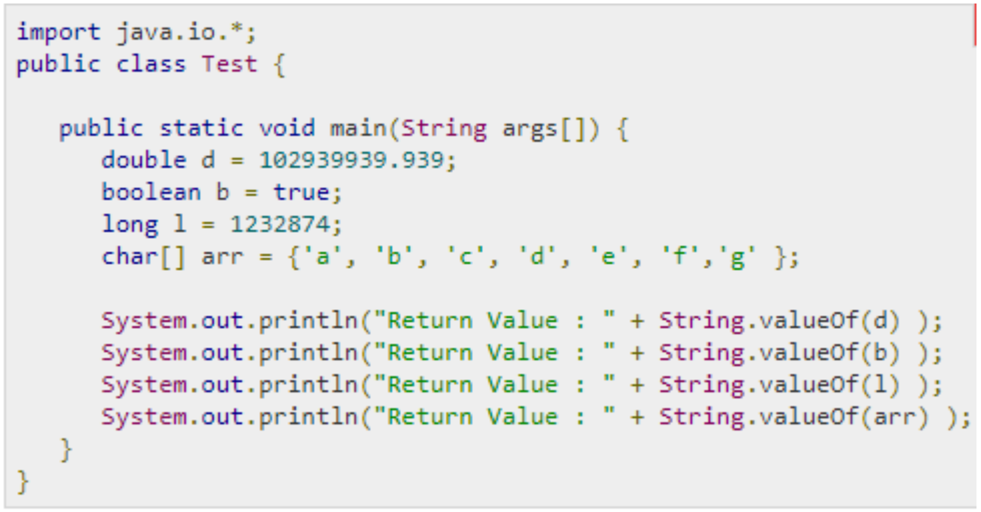
String class – misc. methods

```java
// Fig. 14.8: StringMiscellaneous2.java
// String methods replace, toLowerCase, toUpperCase, trim and toCharArray.

public class StringMiscellaneous2
{
   public static void main(String[] args)
   {
      String s1 = "hello";
      String s2 = "GOODBYE";
      String s3 = "   spaces    ";

      System.out.printf("s1 = %s%ns2 = %s%ns3 = %s%n%n", s1, s2, s3);

      // test method replace
      System.out.printf(
         "Replace 'l' with 'L' in s1: %s%n%n", s1.replace('l', 'L'));

      // test toLowerCase and toUpperCase
      System.out.printf("s1.toUpperCase() = %s%n", s1.toUpperCase());
      System.out.printf("s2.toLowerCase() = %s%n%n", s2.toLowerCase());

      // test trim method
      System.out.printf("s3 after trim = \"%s\"%n%n", s3.trim());

      // test toCharArray method
      char[] charArray = s1.toCharArray();
      System.out.print("s1 as a character array = ");

      for (char character : charArray)
         System.out.print(character);

      System.out.println();
   }
} // end class StringMiscellaneous2
```

Output

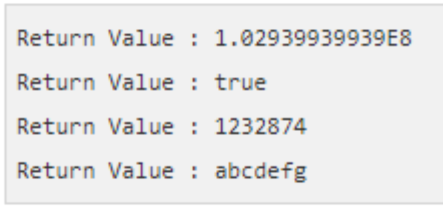
```
s1 = hello
s2 = GOODBYE
s3 =    spaces

Replace 'l' with 'L' in s1: heLLo

s1.toUpperCase() = HELLO
s2.toLowerCase() = goodbye

s3 after trim = "spaces"

s1 as a character array = hello
```

String class – valueOf method



```java
import java.io.*;
public class Test {

   public static void main(String args[]) {
      double d = 102939939.939;
      boolean b = true;
      long l = 1232874;
      char[] arr = {'a', 'b', 'c', 'd', 'e', 'f','g' };

      System.out.println("Return Value : " + String.valueOf(d) );
      System.out.println("Return Value : " + String.valueOf(b) );
      System.out.println("Return Value : " + String.valueOf(l) );
      System.out.println("Return Value : " + String.valueOf(arr) );
   }
}
```

Output



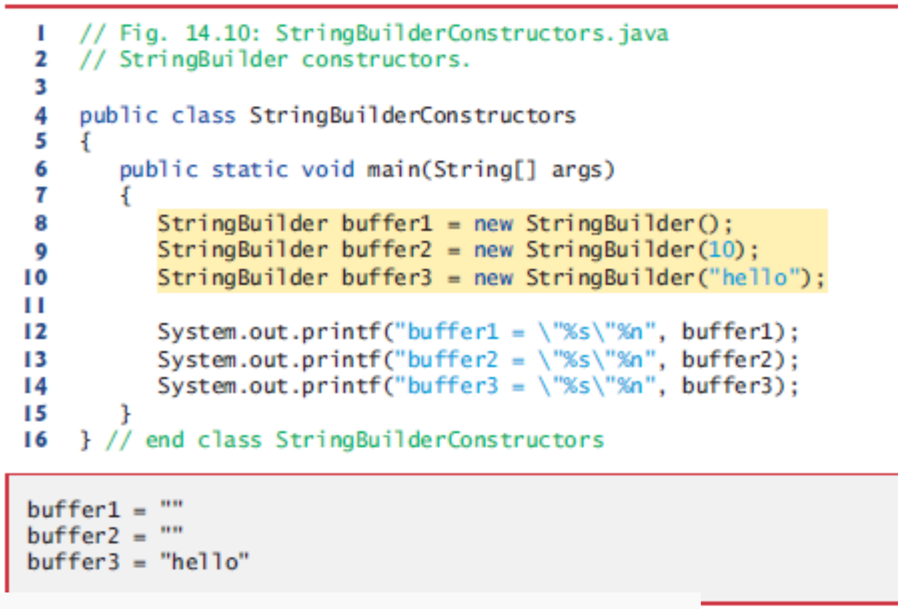
```
Return Value : 1.02939939939E8
Return Value : true
Return Value : 1232874
Return Value : abcdefg
```

Retreived from: https://www.tutorialspoint.com/java/java_string_valueof.htm

StringBuilder Class

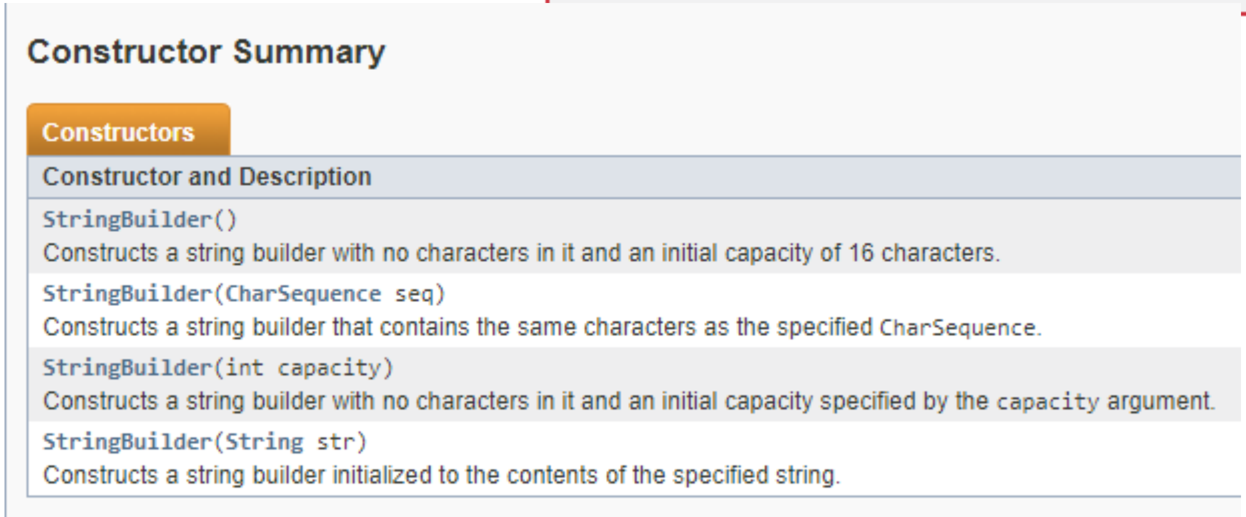
- Recall – Strings are immutable.
- Strings built from the StringBuilder class can change.

StringBuilder Class - Constructors



```java
// Fig. 14.10: StringBuilderConstructors.java
// StringBuilder constructors.

public class StringBuilderConstructors
{
   public static void main(String[] args)
   {
      StringBuilder buffer1 = new StringBuilder();
      StringBuilder buffer2 = new StringBuilder(10);
      StringBuilder buffer3 = new StringBuilder("hello");

      System.out.printf("buffer1 = \"%s\"%n", buffer1);
      System.out.printf("buffer2 = \"%s\"%n", buffer2);
      System.out.printf("buffer3 = \"%s\"%n", buffer3);
   }
} // end class StringBuilderConstructors
```

```
buffer1 = ""
buffer2 = ""
buffer3 = "hello"
```

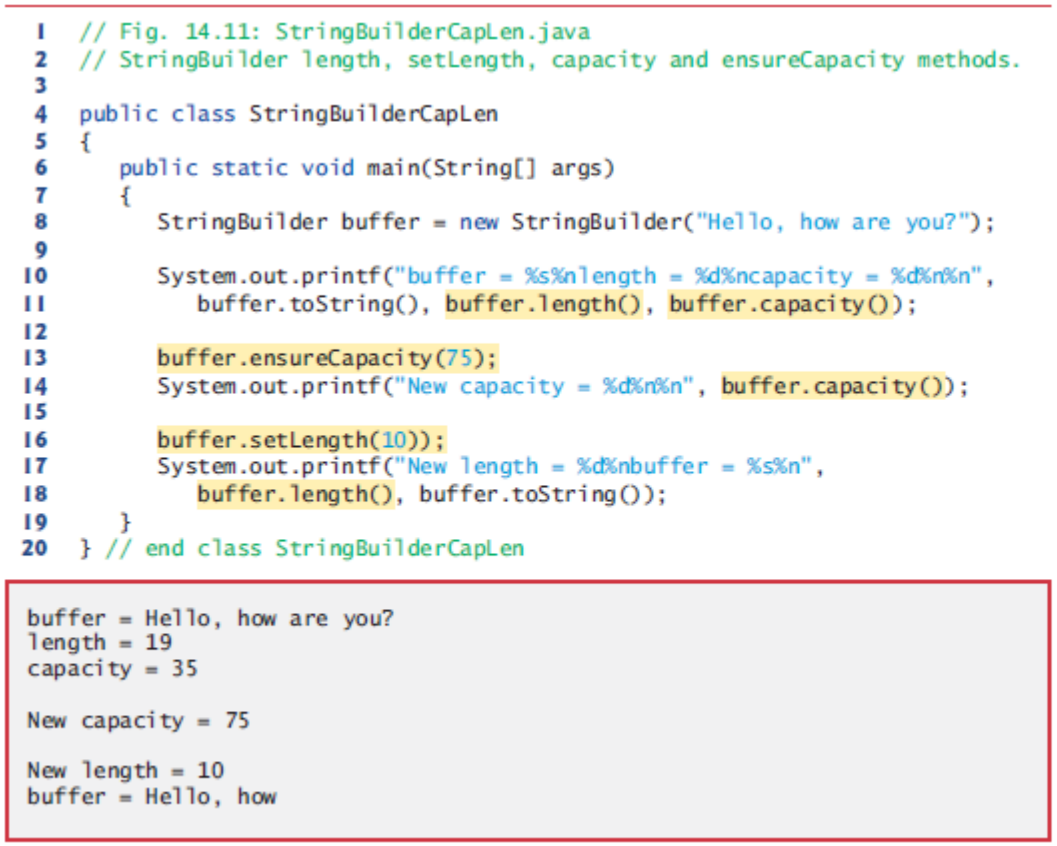


Constructor Summary

Constructors

Constructor and Description
`StringBuilder()` Constructs a string builder with no characters in it and an initial capacity of 16 characters.
`StringBuilder(CharSequence seq)` Constructs a string builder that contains the same characters as the specified `CharSequence`.
`StringBuilder(int capacity)` Constructs a string builder with no characters in it and an initial capacity specified by the `capacity` argument.
`StringBuilder(String str)` Constructs a string builder initialized to the contents of the specified string.

StringBuilder Class - methods

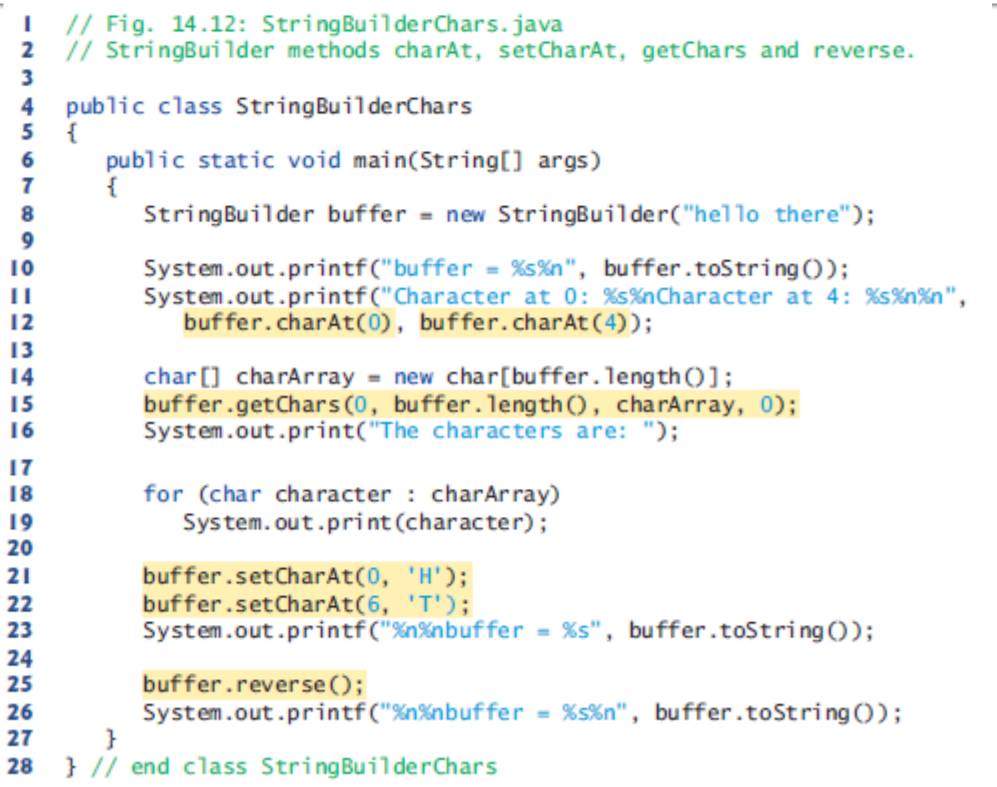


```java
// Fig. 14.11: StringBuilderCapLen.java
// StringBuilder length, setLength, capacity and ensureCapacity methods.

public class StringBuilderCapLen
{
   public static void main(String[] args)
   {
      StringBuilder buffer = new StringBuilder("Hello, how are you?");

      System.out.printf("buffer = %s%nlength = %d%ncapacity = %d%n%n",
         buffer.toString(), buffer.length(), buffer.capacity());

      buffer.ensureCapacity(75);
      System.out.printf("New capacity = %d%n%n", buffer.capacity());

      buffer.setLength(10));
      System.out.printf("New length = %d%nbuffer = %s%n",
         buffer.length(), buffer.toString());
   }
} // end class StringBuilderCapLen
```

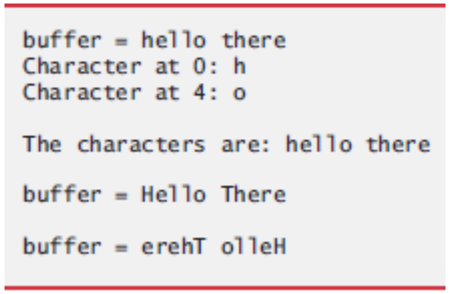
```
buffer = Hello, how are you?
length = 19
capacity = 35

New capacity = 75

New length = 10
buffer = Hello, how
```

Fig. 14.11 | `StringBuilder` `length`, `setLength`, `capacity` and `ensureCapacity` methods.

StringBuilder Class - methods

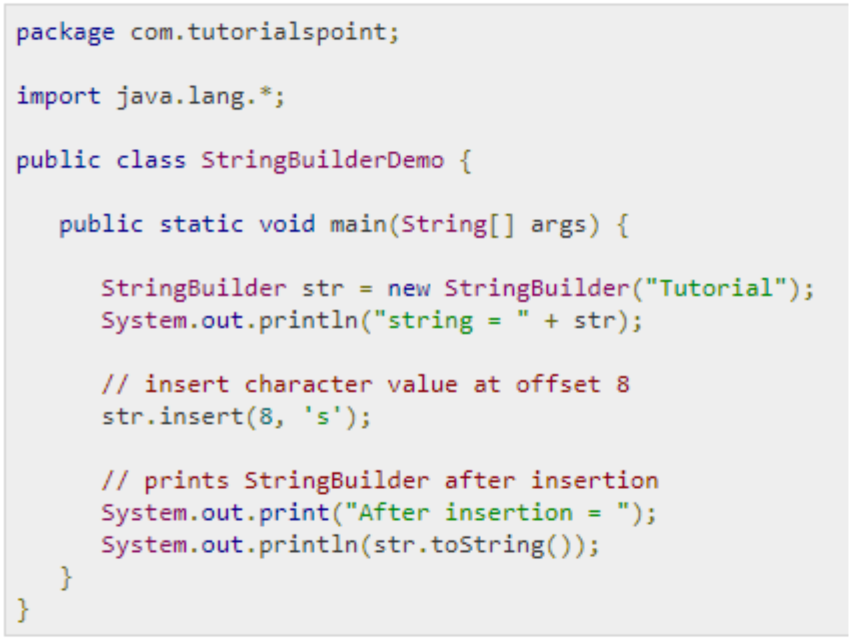


```java
// Fig. 14.12: StringBuilderChars.java
// StringBuilder methods charAt, setCharAt, getChars and reverse.

public class StringBuilderChars
{
   public static void main(String[] args)
   {
      StringBuilder buffer = new StringBuilder("hello there");

      System.out.printf("buffer = %s%n", buffer.toString());
      System.out.printf("Character at 0: %s%nCharacter at 4: %s%n%n",
         buffer.charAt(0), buffer.charAt(4));

      char[] charArray = new char[buffer.length()];
      buffer.getChars(0, buffer.length(), charArray, 0);
      System.out.print("The characters are: ");

      for (char character : charArray)
         System.out.print(character);

      buffer.setCharAt(0, 'H');
      buffer.setCharAt(6, 'T');
      System.out.printf("%n%nbuffer = %s", buffer.toString());

      buffer.reverse();
      System.out.printf("%n%nbuffer = %s%n", buffer.toString());
   }
} // end class StringBuilderChars
```

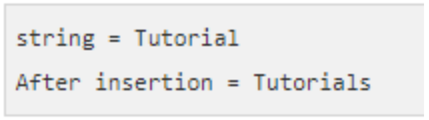


```
buffer = hello there
Character at 0: h
Character at 4: o

The characters are: hello there

buffer = Hello There

buffer = erehT olleH
```

StringBuilder Class - insert



```
package com.tutorialspoint;

import java.lang.*;

public class StringBuilderDemo {

   public static void main(String[] args) {

      StringBuilder str = new StringBuilder("Tutorial");
      System.out.println("string = " + str);

      // insert character value at offset 8
      str.insert(8, 's');

      // prints StringBuilder after insertion
      System.out.print("After insertion = ");
      System.out.println(str.toString());
   }
}
```



```
string = Tutorial
After insertion = Tutorials
```

Retrieved from: https://www.tutorialspoint.com/java/lang/stringbuilder_insert_char.htm

StringBuilder Class - delete

```java
package com.tutorialspoint;

import java.lang.*;

public class StringBuilderDemo {

   public static void main(String[] args) {

      StringBuilder str = new StringBuilder("Java lang package");
      System.out.println("string = " + str);

      // deleting characters from index 4 to index 9
      str.delete(4, 9);
      System.out.println("After deletion = " + str);
   }
}
```

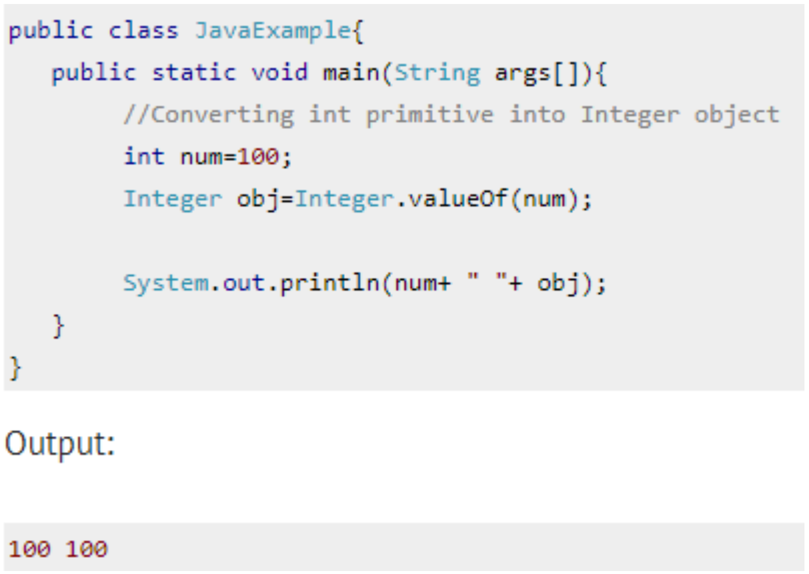
```
string = Java lang package
After deletion = Java package
```

Retrieved from: https://www.tutorialspoint.com/java/lang/stringbuilder_insert_char.htm

Wrapper classes

Java Primitive Data Type	Wrapper Class
int	Integer
double	Double
boolean	Boolean
byte	Byte
char	Character
float	Float
long	Long
short	Short

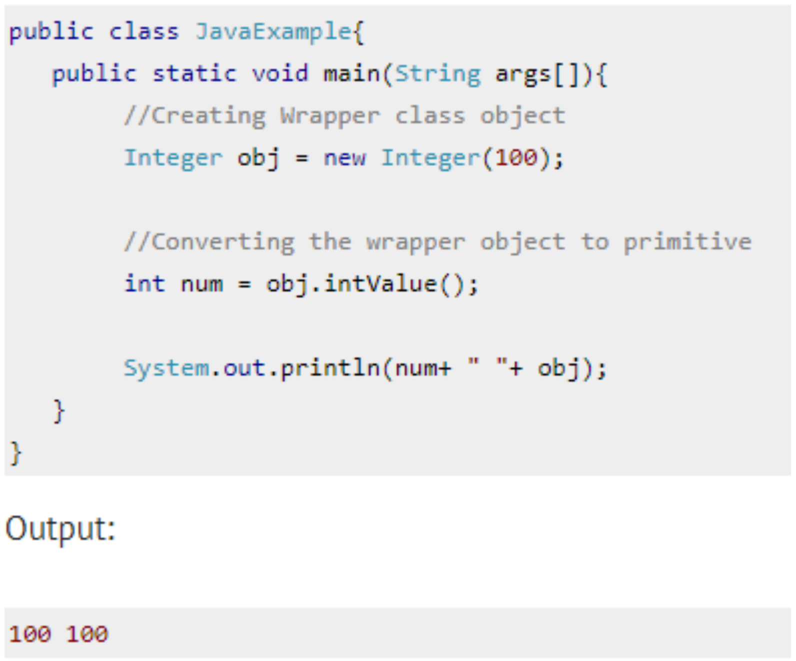
Wrapper classes



```java
public class JavaExample{
   public static void main(String args[]){
        //Converting int primitive into Integer object
        int num=100;
        Integer obj=Integer.valueOf(num);

        System.out.println(num+ " "+ obj);
   }
}
```

Output:

```
100 100
```



```java
public class JavaExample{
   public static void main(String args[]){
        //Creating Wrapper class object
        Integer obj = new Integer(100);

        //Converting the wrapper object to primitive
        int num = obj.intValue();

        System.out.println(num+ " "+ obj);
   }
}
```

Output:

```
100 100
```

Retrieved from: https://beginnersbook.com/2017/09/wrapper-class-in-java/