STRING class

String class properties

- 'String' class is built-in class in Java run-time system.
- Every created string is an object of String type.
  - Even String constants are String object

```
System.out.println("HELLO WORLD");
```

- String class object are immutable
  - Once a String object is created, its content can't be altered.

String class properties

- This seems to be serious restriction, but not –
  - If you need to change, you can create a new one that contains the modification.
  - Java has defined a peer class of String, called StringBuffer, which allows strings to be altered.
    - Like, char array manipulation in C/C++

String Constructors

- String class supports several constructors

```
String s = new String( );
```

  - To create a String with contents of char. Array

```
char c[] = {‘a’, ‘b’, ‘c’};
String s = new String(c);
```

  - Another variation

```
char c[] = {‘a’, ‘b’, ‘c’, ‘d’, ‘e’};
String s = new String(c, 2, 3); // startIndex, numOfChars
```

String Constructors

- String can be constructed from another string

```
char c[] = {‘a’, ‘b’, ‘c’};
String s1 = new String(c);
String s2 = new String(s1);
```

- Another form of ‘String’ creation is from ‘byte’

```
byte ascii[] = {65, 66, 67, 68, 69, 70 };
  String s1 = new String(ascii);
  System.out.println(s1);
  String s2 = new String(ascii, 2, 3);
  System.out.println(s2);
```

ABCDEF

CDE

String Operators

- String literals
  - For each string literal Java automatically constructs a String object.

```
String s = “abc”;
```

- String length

```
int length()          // method format

s.length()// 3
“abc”.length()        // 3
```

String Operators

- String Concatenation

```
String age = “9”;
String s = “He is ” + age + “ years old.”;
    // print - He is 9 years old.
```

- For manipulating very long sentence

```
String longStr = "This could have been " +
 "a very long line that would have " +
 "wrapped around.  But string concatenation " +
 "prevents this.";
```

String Operators

- String Concatenation with other data type

```
int age = 9;
String s = “He is ” + age + “ years old.”;
    // print - He is 9 years old.

String s = “four: ” + 2 + 2;
    // print: 22
String s = “four: ” + (2+2);
    // print: 4
String s = “four: ” + (2/2);
    // print: 1
```

String Operators

- String Conversion & toString()
  - Every class implements ‘toString()’ method, which determine the string representation of objects.
  - Now this method can be override for own implementation

  General form –

  String toString ( )

```
// Override toString() for Box class.
class Box {
  double width;
  double height;
  double depth;
  Box(double w, double h, double d) {
    width = w;
    height = h;
    depth = d;
  }
  public String toString() {
    return "Dimensions are " + width + " by " +
           depth + " by " + height + ".";
  }
}
```

```
class toStringDemo {
  public static void main(String args[]) {
    Box b = new Box(10, 12, 14);
    String s = "Box b: " + b; // concatenate Box object

    System.out.println(b); // convert Box to string
    System.out.println(s);
  }
}
```

```
Dimensions are 10 by 14 by 12.
Box b: Dimensions are 10 by 14 by 12.
```

Character Extraction

- charAt()

```
char ch;
ch = “abc”.charAt(1);          // ch = ‘b’
```

- getBytes( )
  - Stores the characters in an array of bytes.

```
String s = “ABC”;
byte b[] = new byte[s.length()];
b = s.getBytes();
b = “ABC”.getBytes();          // equivalent
```

Character Extraction

- getChars( )

  void getChars(srcStart, srcEnd, char target[], targetStart)

```
String s = "This is a demo of the getChars method.";
  int start = 10;
  int end = 14;
  char buf[] = new char[end - start];

  s.getChars(start, end, buf, 0);
```

Output: demo

String Comparison

General form –

```
boolean equals(String str)
        //      Case sensitive
boolean equalIgnoreCase(String str)
        //      ignores case when comparing
```

```
String s1 = "Hello";
  String s2 = "Hello";
  String s3 = "Good-bye";
  String s4 = "HELLO";
  System.out.println(s1 + " equals " + s2 + " -> " +
                     s1.equals(s2));
  System.out.println(s1 + " equals " + s3 + " -> " +
                     s1.equals(s3));
  System.out.println(s1 + " equals " + s4 + " -> " +
                     s1.equals(s4));
  System.out.println(s1 + " equalsIgnoreCase " + s4 + " -> " +
                     s1.equalsIgnoreCase(s4));
```

```
Hello equals Hello -> true
Hello equals Good-bye -> false
Hello equals HELLO -> false
Hello equalsIgnoreCase HELLO -> true
```

String Comparison

‘equals’ vs. ‘==’

- equals() method compares the characters inside String
- ‘==’ operator compares two object references to see, they refer to the same instance.

```
String s1 = "Hello";
String s2 = new String(s1);
System.out.println(s1 + " equals " + s2 + " -> " +
                   s1.equals(s2));
System.out.println(s1 + " == " + s2 + " -> " + (s1 == s2));
```

```
Hello equals Hello -> true
Hello == Hello -> false
```

String Comparison

int compareTo(String str)

Value	Meaning
<0	invoking string in less then str
>0	invoking string in greater then str
==0	The two strings are equal.

```
// Sorting in Alphabetical order
static String arr[] = {
    "Now", "is", "the", "time", "for", "all", "good", "men",
    "to", "come", "to", "the", "aid", "of", "their", "country"};

    for(int j = 0; j < arr.length; j++) {
      for(int i = j + 1; i < arr.length; i++) {
        if(arr[i].compareTo(arr[j]) < 0) {
          String t = arr[j];
          arr[j] = arr[i];
          arr[i] = t;
        }
      }
      System.out.println(arr[j]);
    }
```

Changing Case

toLowerCase()

toUpperCase()

```
String s = "This is a test.";
System.out.println("Original: " + s);
String upper = s.toUpperCase();
String lower = s.toLowerCase();
System.out.println("Uppercase: " + upper);
System.out.println("Lowercase: " + lower);
```

Command-Line Arguments

```
class CommandLine {
    public static void main (String[] args) {
     int width=0, height=0, area;

      if(args.length == 0 ){
                    //no command line arguments
     System.out.println("CommandLine argument missing”)

          System.exit(0);
     }//if
```

Command-Line Arguments

```
else {
  width =Integer.parseInt(args[0]);
  height=Integer.parseInt(args[1]);
    }//else
 area = width*height;
 System.out.println("Area is " + area);
  }
}
```

Command-Line Arguments

C:\> java CommandLine

CommandLine argument missing

C:\>java CommandLine 4 5

Area is 20