edureka!

JAVA STRING CHEAT SHEET

Learn JAVA from experts at http://www.edureka.co

Java Strings

In **Java**, a **string** is an object that represents a sequence of characters. The **java**.lang.**String** class is used to create **string** object. **String** contains an immutable sequence of Unicode characters.

Creating a String

```
String str1 = “Welcome”;
// Using literal String
str2 = new String(”Edureka”);
 // Using new keyword
```

Immutable Strings

```
class Stringimmutable
{
public static void main(String args[])
{
 String s="JavaStrings";
 s.concat(" CheatSheet");
 System.out.println(s);
 }
}
```

Methods of Strings

```
str1==str2 //compares address;
String newStr = str1.equals(str2);
//compares the values
String newStr =str1.equalsIgnoreCase()
//compares the values ignoring the case
newStr = str1.length()
//calculates length
newStr = str1.charAt(i)
//extract i'th character
 newStr = str1.toUpperCase()
//returns string in ALL CAPS
 newStr = str1.toLowerCase()
//returns string in ALL LOWERvCASE
newStr = str1.replace(oldVal, newVal)
//search and replace
newStr = str1.trim()
//trims surrounding whitespace
newStr = str1.contains("value");
//check for the values
newStr = str1.toCharArray();
// convert String to character type
array newStr = str1.IsEmpty();
//Check for empty String
newStr = str1.endsWith();
//Checks if string ends with the given
suffix
```

Programs

Removing Trailing spaces from string

```
int len = str.length();
for( ; len > 0; len--) {
 if( ! Character.isWhitespace(
str.charAt( len - 1)))
 break;
}
return str.substring( 0, len);
```

Finding Duplicate characters in a String

```
public void countDupChars{
Map<Character, Integer> map = new HashMap
<Character, Integer>();
//Convert the String to char array
char[] chars = str.toCharArray();
Set<Character> keys = map.keySet();
//Obtaining set of keys
public static void main(){
System.out.println("String: Edureka");
obj.countDupChars("Edureka");
System.out.println("\nString:
StringCheatSheet");
obj.countDupChars("StringCheatSheet");
}
}
```

String Conversions

String to Int Conversion

```
String str="123";
int inum1 = 100;
int inum2 =
Integer.parseInt(str);
// Converting a string to int
```

Int to String Conversion

```
int var = 111;
String str =
String.valueOf(var);
System.out.println(555+str);
// Conversion of Int to String
```

JAVA
CERTIFICATION
TRAINING

String to Double Conversion

```
String str = "100.222";
double dnum = Double.parseDouble(str);
//displaying the value of variable dnum
```

Double to String Conversion

```
double dnum = 88.9999; //double value
String str = String.valueOf(dnum);
//conversion using valueOf() method
```

String Joiner Class

```
StringJoiner mystring = new
StringJoiner("-");
// Passing Hyphen(-) as delimiter
mystring.add("edureka");
// Joining multiple strings by using
add() method
mystring.add("YouTube");
```

String reverse using Recursion

```
String str = "Welcome to Edureka";
String reversed=reverseString(str);
ReturnreverseString(str.substring(1))
+ str.charAt(0);
//Calling Function Recursively
```

String reverse

```
String str;
System.out.println("Enter your
username: ");
String reversed = reverseString(str);
// Reversing a String
return reverseString(str.substring(1))
+ str.charAt(0);
//Calling Function Recursively
```

String Pool

```
String str1 = "abc";
String str2 = "abc";
System.out.println(str1 == str2);
System.out.println(str1 == "abc");
```

String vs String Buffer

It is immutable String class overrides the equals() method of Object class..	It is mutable StringBuffer class doesn't override the equals() method of Object class.

String Buffer vs Builder

StringBuffer is *synchronized* i.e. thread safe. StringBuffer is *less efficient* than StringBuilder as it is Synchronized.	StringBuilder is *non-synchronized* i.e. not thread safe. StringBuilder is *more efficient* than StringBuffer as it is not synchronized.