

Example -1 : Exploring String Class

- Explore various methods e.g. length(), equal(), compareTo(), startsWith(), indexOf(), getChars() etc.
- Find number of occurrence of a pattern in a string
- Number of different characters in a sentence
- Copy characters from a string to a char array
- Find out tokens from a string
- Formatting string
- Check whether a sentence is palindrome or not
- Using regular expression for masking and validation.


File : **StringExample.java**

```
package javaapp1;
import java.util.HashMap;           import java.util.Arrays;           import java.util.Calendar;
import java.util.GregorianCalendar; import static java.util.Calendar.*;
import java.util.regex.*;           import java.util.ArrayList;      import java.util.List;

public class StringExample {

    public static final String STR_TEST1 = "Strings receive special treatment in Java, because they are
                                            used frequently in a program";
    public static final String STR_TEST2 = "The + operator is the only operator that is internally
                                            overloaded to support string concatenation in Java";
    public static final String COM_STR1 = "String literals are stored in a common pool";
    public static final String COM_STR2 = "STRING LITERALS are stored in a common pool";

    // Find number of different character in a string
    public void CharCounter() {
        HashMap<Character, Integer> map = new HashMap<>();
        for (char ch : STR_TEST1.toCharArray()) { // create a char[] from this string
            if (map.containsKey(ch)) {
                int val = map.get(ch);      map.put(ch, val + 1);
            } else {
                map.put(ch, 1);
            }
        }
        System.out.println(map);
    }
}
```



```

public void FormatExp() {
    // Format a string containing a date.
    Calendar c = new GregorianCalendar(1995, MAY, 23);

    // Format string is the first argument to the format method. It contains three format specifiers
    // "%1$tm", "%1$te", and "%1$tY" which indicate how the arguments should be processed and
    // where they should be inserted in the text. The remaining portions of the format string are fixed
    // text.
    // Format Specifier :
    // First one is optional argument_index is a decimal integer indicating the position of the
    // argument in the argument list. The first argument is referenced by "1$", the second by "2$",
    // etc. Second part optional flags is a set of characters that modify the output format. The set of
    // valid flags depends on the conversion. t or T is prefix for Date Time conversion.
    // Next Char : 'm' for month, 'e' day of month , 'Y' for year

    String bthDate = String.format("Duke's Birthday: %1$tm %1$te,%1$tY", c);
    // -> bthDate == "Duke's Birthday: May 23, 1995
    System.out.println(bthDate);

    // %1$ represents first argument, %2$ second argument
    String insm = String.format("Our Institute Name" + " is: %1$s, and it is at %2$s", "Heritage Institute
                                of Technology", "Kolkata");
    System.out.println(insm);

    // Output is given upto 8 decimal places
    String str1 = String.format("My answer is %.8f", 47.65734);

    // between "My answer is" and "47.65734000" there are 15 spaces
    String str2 = String.format("My answer is %15.8f", 47.65734);
    System.out.println(str1);    System.out.println(str2);
}

```

```

public void ExploreStrFunctions() {

    // Length : returns the length of the String, if empty length ==0
    System.out.println("Length of STR_TEST1 = " + STR_TEST1.length());

    // Comparison : CANNOT use '==' or '!=' to compare two Strings in Java
    boolean chkEq = COM_STR1.equals(COM_STR2);
    System.out.println("Check Equality COM_STR1 and COM_STR2 = " + chkEq);
    System.out.println("Check Equality COM_STR1 and COM_STR2 = " +
                        COM_STR1.equalsIgnoreCase(COM_STR1));

    // return 0 if this string is the same as another; <0 if lexicographically less than another; or >0
}

```

```

int rtnVal = COM_STR1.compareTo(COM_STR2);
System.out.println("Comparing COM_STR1 and COM_STR2 = " + rtnVal);

// search begins at fromIndex, if ignored start from beginning
System.out.println("Start with operator from 6 : " + STR_TEST2.startsWith("operator", 6));
//chkEq = STR_TEST1.endsWith(STR_TEST1);

// Searching & Indexing
System.out.println("Start Index of operator in STR_TEST2 : " + STR_TEST2.indexOf("operator"));
System.out.println("Last Index of operator in STR_TEST2 : " + STR_TEST2.lastIndexOf("operator"));

// Find occurrence of a pattern in a string
int count = 0; String pattern = "operator";
int i = 0;
// Keep calling indexOf for the pattern.
while ((i = STR_TEST2.indexOf(pattern, i)) != -1) {
    // Advance starting index.
    i += pattern.length();
    // Increment count.
    count++;
}
System.out.println("No of occurrence of operator in STR_TEST2 : " + count);

// The getChars() method of String class generally copies character from this string to the target
// character array object. Four parameters : starting index, end index, target array, char offset on
// the target char array
char[] target = new char[45];
target[0] = '*';
target[1] = '/';
COM_STR2.getChars(7, COM_STR2.length(), target, 2);
System.out.println("Copies character from COM_STR2 to an char array : ");
System.out.println(Arrays.toString(target));

// Find token : recommended String.split() not using StringTokenizer.
String[] tokens = "I,am ,Legend, , of ,you ?".split(",");
System.out.println("Print tokens of I,am ,Legend, , of ,you ? using split with delimiter , :");
for (String token : tokens)
{
    System.out.println(token);
}

// Create a string of YYYY-MM-DD HH:MM:SS
// StringBuilder is more efficient than String concatenation

```

```

int year = 2010, month = 10, day = 10;
int hour = 10, minute = 10, second = 10;
String dateStr = new StringBuilder()
    .append(year).append("-").append(month).append("-").append(day).append(" ")
    .append(hour).append(":").append(minute).append(":").append(second).toString();
System.out.println("Creating Date string using StringBuilder : " + dateStr);

}
// Find palindrom
// Example :
// A nut for a jar of tuna. No lemon, no melon. Some men interpret nine memos.
// Gateman sees name, garageman sees nametag.
public void FndPalindrom() {
    String str1 = "Gateman sees name, garageman sees nametag";
    // Removing space, comma from string using regular expression
    // \s is regular expression for white space tab etc
    String str2 = str1.replaceAll("[\\s,]", "");
    str2 = str2.toLowerCase();
    StringBuffer s1=new StringBuffer(str2);
    StringBuffer s2=new StringBuffer(s1);

    s1.reverse();
    System.out.println("Given String is : "+str1);
    //if(String.valueOf(s1).compareTo(String.valueOf(s2))==0)
    if(s1.toString().compareTo(s2.toString())==0)
        System.out.println(str1 + " is Palindrome");
    else
        System.out.println(str1 + " is Not Palindrome");
}
// Regular expressions are a language of string patterns built in to most modern programming
languages, including Java 1.4 onward; they can be used for: searching, extracting, and modifying
text.

// For advanced regular expressions the java.util.regex.Pattern and java.util.regex.Matcher classes
are used. First create a Pattern object which defines the regular expression. This Pattern object
allows to create a Matcher object for a given string. This Matcher object then allows you to do
regex operations on a String.

// Here we will mask the number after clientid= with "***masked***" so that number will not be
visible to the user when string is returned

public void RegExpDemo() {

    String input = "User clientId=23421. Some more text clientId=33432. This clientNum=100";

```

```
// Notice that in Java, you will need to "double escape" the backslashes of regular expression.
// Pattern "(clientId=)(\\d+)"
// (clientId=)    group everything within the parenthesis as group 1
// clientId=      match the text 'clientId='
// (\\d+)         group everything within the parenthesis as group 2
// \\d+          match one or more digits
```

Pattern **clientId=** followed by any number of digit

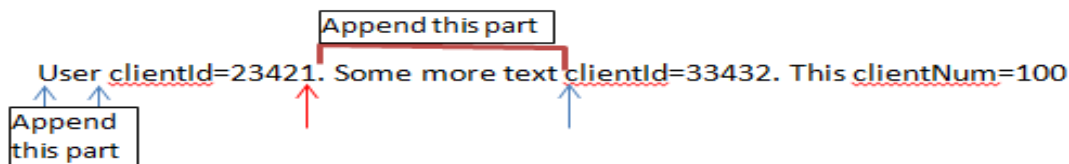
```
Pattern p = Pattern.compile("(clientId=)(\\d+)");
Matcher m = p.matcher(input);
StringBuffer result = new StringBuffer();
System.out.println(input + " : masking the numbers after clientId=");
while (m.find()) {
    System.out.println("Masking: " + m.group(2));
    m.appendReplacement(result, m.group(1) + "***masked***");
}
m.appendTail(result);
System.out.println(result);
}
```

public boolean find()

Attempts to find the next subsequence of the input sequence that matches the pattern.

Next matching start at the first character not matched by the previous match.

This method reads characters from the input sequence, starting at the append position, and appends them to the given string buffer. It is intended to be invoked after one or more invocations of the **appendReplacement** method in order to copy the remainder of the input sequence.



public Matcher appendReplacement(StringBuffer sb, String replacement)

Implements a non-terminal append-and-replace step.

This method performs the following actions:

- It reads characters from the input sequence, starting at the **append position**, and **appends them to** the given string buffer. It stops after reading the **last character preceding the previous match**, that is, the character at index `start() - 1`.
- It **appends** the **given replacement string** to the string buffer.
- It sets the append position of this matcher to the index of the **last character matched, plus one**, that is, to `end()`.

```
// This produces the following output:
// Masking: 23421
// Masking: 33432
// User clientId=***masked***. Some more text clientId=***masked***. This clientNum=100

// Validate SSN using regular expression
public void ValidateSSN() {
    List<String> input = new ArrayList<String>();
    input.add("123-45-6789");    input.add("9876-5-4321");
    input.add("987-65-4321 (attack)"); input.add("987-65-4321 ");
    input.add("192-83-7465");
    // ^          match the beginning of the line
    // ()          group everything within the parenthesis as group 1
    // \\d{n}      match n digits, where n is a number equal to or greater than zero
    // -?          optionally match a dash
    // $           match the end of the line
    for (String ssn : input) {
        if (ssn.matches("^\\d{3}-?\\d{2}-?\\d{4}$")) {
            System.out.println("Found good SSN: " + ssn);
        }
    }
}
// Found good SSN: 123-45-6789
// Found good SSN: 192-83-7465
}
File : Javaapp1.java
private static void driverStringExp()
{
    StringExample se = new StringExample();
    System.out.println("===== OUTPUT FOR STRING FUNCTIONS =====");
    se.ExploreStrFunctions();
    System.out.println("===== OUTPUT FOR STRING FORMATS =====");
    se.FormatExp();
    System.out.println("===== OUTPUT FOR CHARACTER COUNTS IN A STRING =====");
    se.CharCounter();
    System.out.println("===== OUTPUT FOR PALINDROM =====");
    se.FndPalindrom();
    System.out.println("===== OUTPUT FOR REGULAR EXPRESSION =====");
    se.RegExpDemo();
    System.out.println("===== OUTPUT FOR VALIDATE SSN USING REGEX =====");
    se.ValidateSSN();
}

```

Example -2 : Convert Amount to word

File : **AmtWordConversion.java**

```
import java.util.*;
public class AmtWordConversion {
    private static final String one[]= {" ", " one", " two", " three", " four", " five", " six", " seven", " eight",
        "Nine", " ten", " eleven", " twelve", " thirteen", " fourteen", "fifteen",
        "sixteen", " seventeen", " eighteen", " nineteen"};

    private static final String ten[]= {" ", " ", " twenty", " thirty", " forty", " fifty", " sixty", "seventy",
        "eighty", " ninety"};
    // Convert every part of the number in words
    private void prWord(int n, String ch) {
        if(n>19) { System.out.print(ten[n/10]+" "+ one[n%10]); }
        else { System.out.print(one[n]); }
        if(n>0) System.out.print(ch);
    }

    // Convert the Amount to Words
    public void ConvAmt() {
        int n=0;
        System.out.println("Enter an integer number <= 2147483647. If it is > that no. exception will
            occure\n");
        System.out.println("To stop the process enter a -ve number \n");

        while (n >= 0) {
            System.out.println("\nEnter an integer number: ");
            Scanner scanf = new Scanner(System.in);
            n = scanf.nextInt(); // Convert the resulting token to integer
            if(n<0) {
                System.out.println("Terminated");
                return;
            }
            else {
                this.prWord((n/1000000000)," Hundred ");
                this.prWord((n/10000000)%100," crore ");
                this.prWord(((n/100000)%100)," lakh ");
                this.prWord(((n/1000)%100)," thousand ");
                this.prWord(((n/100)%10)," hundred ");
                this.prWord((n%100)," ");
            }
        }
    }
}
```

```
}
```

File : **Javaapp1.java**

```
// Make it static otherwise unable to call from another static method main()
```

```
private static void driverAmtWordConversion()
{
    AmtWordConversion amtWd = new AmtWordConversion();
    amtWd.ConvAmt();
}
```

Example -3 : Create a memory database for Student object using ArrayList and perform various CRUD operations.

File : **Student.java**

```
package javaapp1;
```

```
import java.util.ArrayList;    import java.util.Iterator;
import java.util.HashMap;     import java.util.Map;
```

```
public class Student {
    private String stdname;    private String course;        private Integer age;
```

```
    public Student (String nm, String crs, Integer ag)
    {
        stdname = nm;  course= crs;  age = ag;
    }
```

```
    public void setStdName(String d) { stdname = d; }
    public void setAge(int u)      { age = u; }
    public String getStdName ()    { return stdname; }
    public int getAge()            { return age; }
```

```
// Add a new Student in the List
```

```
public static void AddNewStd(ArrayList<Student> stdList, String nm, String crs, Integer ag)
{
    Student std = new Student(nm,crs,ag);        stdList.add(std);
    System.out.println(std.stdname + " , " + std.course + " , " + std.age + " - Added Successfully");
}
```

```
// Find a student by name, if not found null is returned
```

```
public static Student FindStd(ArrayList<Student> stdList, String nm)
{
    for (Student std : stdList)
    {
        if (std.stdname == nm)
        {
            System.out.println(std.stdname + " , " + std.course +
```



```

        " , " + std.age + " -Found in Index " + stdList.indexOf(std) );
    return std;
}
}
System.out.println(nm + " -Not Found" );
return null;
}
// Print the students in the list
public static void PrintStdList(ArrayList<Student> stdList)
{
    // Reading all elements from ArrayList by using Iterator. We can iterate through the ArrayList
    // based on index too. hasNext returns true if the iteration has more elements.
    System.out.println("List of Students");
    Iterator<Student> itr = stdList.iterator();
    while(itr.hasNext()) {
        Student sd = itr.next();    // Returns the next element in the iteration.
        System.out.println(sd.stdname + " , " + sd.course + " , " + sd.age );
    }
}
// Remove student with name
public static void RemoveStd(ArrayList<Student> stdList, String snm)
{
    Student std = FindStd(stdList, snm); // Find student
    if (std != null)
    {
        stdList.remove(std);
        System.out.println(std.stdname + " , " + std.course + " , " + std.age + " - Removed");
    }
    else
    {
        System.out.println(snm + " -Not Removed");
    }
}
// Edit Student Information
public static void EditStd(ArrayList<Student> stdList, String snm, String crs, int ag)
{
    Student std = FindStd(stdList, snm); // Find student
    if (std != null)
    {
        std.course = crs; std.age = ag; int idx = stdList.indexOf(std); stdList.set(idx, std);
        System.out.println(snm + " - Changed Successfully");
    }
    else
    {

```

```

        System.out.println(snm + " -Not Changed");
    }
}
// Convert keys of map to an array or arrayList
public static void ConvertMapToArrayList ()
{

    Map<String, String> map = new HashMap<>();
    FillData(map);    // Fill some data in the map

    // Convert keys to array
    String[] strArray = KeysAsArray(map);
    for (String echStr : strArray) {    System.out.println(echStr);    }
    // Convert keys to ArrayList
    ArrayList<String> aryLst = KeysAsList(map);
    for (String echStr : aryLst) {    System.out.println(echStr);    }
}

// Fill some data to a map
private static void FillData(Map<String, String> map)
{
    map.put("Android", "Mobile");    map.put("Eclipse IDE", "Java");
    map.put("Eclipse RCP", "Java");    map.put("Git", "Version control system");

    // Print each entry of Map
    for (Map.Entry<String, String> e : map.entrySet())
    {
        // Get Key and Value of each entry
        System.out.println( e.getKey() + " , " + e.getValue());
    }
}
private static String[] KeysAsArray(Map<String, String> map)
{
    return map.keySet().toArray(new String[map.keySet().size()]);
}
// Assumes the key is of type String
private static ArrayList<String> KeysAsList(Map<String, String> map)
{
    ArrayList<String> list = new ArrayList<String>(map.keySet());
    return list;
}
}

```

File : **JavaApp1.java**

```
private static void driverStudent()
{
    // Create Student Database in Memory using ArrayList
    ArrayList<Student> stnDB = new ArrayList<Student>();

    Student.AddNewStd(stnDB, "Bibek Roy", "B.Tech", 19);
    Student.AddNewStd(stnDB, "Sumit Mitra", "B.Tech", 20);
    Student.AddNewStd(stnDB, "Surajit Dalal", "B.Tech", 21);
    Student.AddNewStd(stnDB, "Trisha Das", "M.Tech", 23);
    Student.AddNewStd(stnDB, "Rituparna Kar", "M.Tech", 24);
    Student.AddNewStd(stnDB, "Ahana Sarkar", "B.Tech", 19);
    Student.AddNewStd(stnDB, "Deep Shankar Saha", "M.Tech", 24);
    Student.AddNewStd(stnDB, "Sonu Shreshtha", "B.Tech", 18);
    Student.AddNewStd(stnDB, "Bibek Roy", "B.Tech", 19);
    Student.AddNewStd(stnDB, "Neha Aryani", "B.Tech", 18);
    Student.PrintStdList(stnDB);    // print the content using a static method

    // Find a student and print appropriate message
    Student sd = Student.FindStd(stnDB, "Ahana Sarkar");
    Student sd1 = Student.FindStd(stnDB, "Kamal Podder");

    // Remove a student and print appropriate message
    Student.RemoveStd(stnDB, "Ahana Sarkar");    Student.RemoveStd(stnDB, "Kamal Podder");
    Student.PrintStdList(stnDB);

    // Edit a Student information ( course and age ) of a student with name
    Student.EditStd(stnDB, "Sonu Shreshtha", "M.Tech", 20);
    Student.PrintStdList(stnDB);

    // Swap two elements in the ArrayList by calling Collections.swap() method
    // We have to pass the indexes which you need to swap.
    Collections.swap(stnDB, 4, 7);    Student.PrintStdList(stnDB);

    // Reverse the List
    Collections.reverse(stnDB);    Student.PrintStdList(stnDB);

    // Convert Keys of map to arrayList
    Student.ConvertMapToArryList();
}
```