Java String

- In Java, the string is basically an object that represents a sequence of char values. An array of characters works same as Java string.

- To create strings in java by using these three classes
  - String class
  - StringBuffer class
  - StringBuilder class

- The Java String is immutable, which means it cannot be changed.

  - Whenever we change any string, a new instance is created.

  - For mutable strings, you can use StringBuffer and StringBuilder classes.

- **String class**
  - The *java.lang.String* class is used to create a string object.

  - There are two ways to create String object:
    - string literal
    - new keyword

- **String Literal**

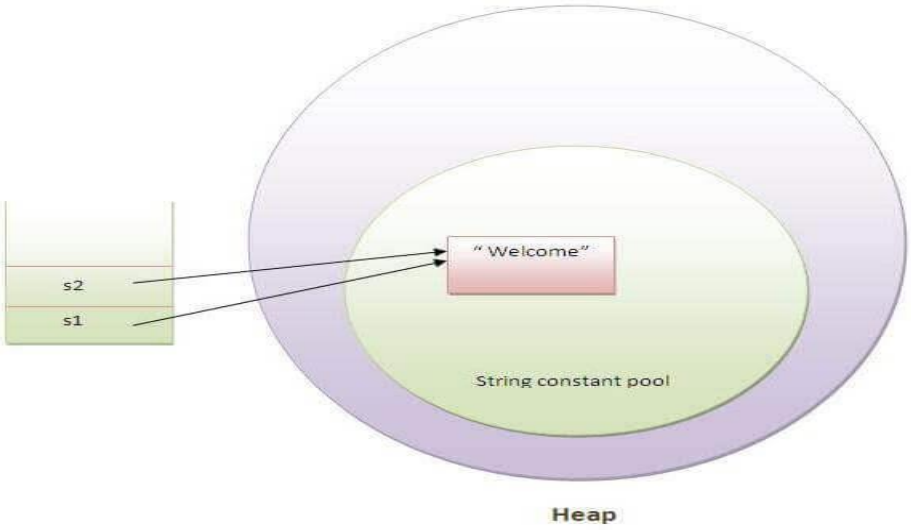
  - String s1="Welcome"; *// the JVM checks the "string constant pool" first. If the string doesn't exist in the pool, a new string instance is created and placed in the pool.*

  - String s2="Welcome"

    //It doesn't create a new instance beca use the string already exists in the

pool, a reference to the pooled instance is returned.

- String objects are stored in a special memory area known as the "string constant pool.



- **New Keyword**
  - String s=new String("Welcome");

    //creates two objects and one reference variable.
    JVM will create a new string object in normal (non-pool) heap memory, and the literal "Welcome" will be placed in the string constant pool. The variable ‘s’ will refer to the object in a heap (non-pool).

Example – 1: Create String by using String literal and new keyword.

```
public class StringExample{

    public static void main(String args[]){

        //creating string by java string literal
        String s1="java";

        char ch[]={'s','t','r','i','n','g','s'};  //character array
```

```
        //converting char array to string
        String s2=new String(ch);

        //creating java string by new keyword
        String s3=new String("example");

        //output
        System.out.println(“1st String :  ” + s1);
        System.out.println(“2nd String :  ” + s2);
        System.out.println(“3rd String :  ” + s3);

    }

}
```

Output:

1st String : java

2nd String : strings

3rd String : example

String Length

The length of a string can be found by the **length()** method.

Example: Find the length of the String.

```
public class Length {

    public static void main(String[] args) {

        String txt = "ABCDEFGHIJKLMNOPQRSTUVWXYZ";

        System.out.println("The length of the text string is: " + txt.length());
    }
}
```

Output:

The length of the text string is: 26

To Upper and To Lower

Make the string to upper or lower case.

Example:

```
public class UpperLower {

	public static void main(String[] args) {

		String txt = "Hello World";

		System.out.println(“The upper case is “ + txt.toUpperCase());

		System.out.println(“The lower case is “ + txt.toLowerCase());
	}
}
```

Output:

The upper case is HELLO WORLD
The lower case is hello world

Finding Index of a Character in a String

- The indexOf() method returns the **index** (the position) of the first occurrence of a specified text in a string (including whitespace).

Example: Find the index of “locate” in a string "Please locate where 'locate' occurs!", by using indexOf() method.

```
public class FindCharIndex {

	public static void main(String[] args) {

		String txt = "Please locate where 'locate' occurs!";
		int i= txt.indexOf("locate");
		//int i= txt.indexOf(‘e’);
```

```
                System.out.println("The index value is " + i);
        }
}
```

Output:

The index value is 7

Processing a String One Character at a Time

- A string's **charAt()** method retrieves a given character by index number (starting at zero) from within the String object.
- To process all the characters in a String, one after another, use a for loop ranging from zero to String.length()-1.

Example : Retrieves a given character by index number

```
public class CharAtExample{

    public static void main(String args[]){

        String name="Hello";

        char ch=name.charAt(4);   /returns the char value at the 4th index

        System.out.println("Character at index 4 is "+ ch);

    }

}
```

Output:

Character at index 4 is o

Example: Process all the characters in a String:

```
public class StrCharAt {

        public static void main(String[] av) {
```

```
        String a = "A quick bronze fox kept a lazy bovine";

        for (int i=0; i < a.length(); i++)              // Don't use for each

            System.out.println("Char " + i + " is " + a.charAt(i));

    }
}
```

Output:

```
Char 0 is A
Char 1 is
Char 2 is q
Char 3 is u
Char 4 is i
Char 5 is c
Char 6 is k
Char 7 is
Char 8 is b
Char 9 is r
Char 10 is o
Char 11 is n
Char 12 is z
Char 13 is e
Char 14 is
Char 15 is f
Char 16 is o
Char 17 is x
Char 18 is
Char 19 is l
Char 20 is e
Char 21 is p
Char 22 is t
Char 23 is
Char 24 is a
Char 25 is
Char 26 is l
Char 27 is a
```

Char 28 is z
Char 29 is y
Char 30 is
Char 31 is b
Char 32 is o
Char 33 is v
Char 34 is i
Char 35 is n
Char 36 is e

Example: process all the characters in a String using toCharArray() method.

```
public class ForEachChar {

	public static void main(String[] args) {

		String str = "Hello world";

		// toCharArray() returns an Array of chars after
		//converting a String into sequence of characters.
		 char[] array= str.toCharArray();

		// for (char ch : s) {...} Does not work, in Java 7

		for (char ch : array) {

				System.out.println(ch);
		}
	}
}
```

Output:

H
e
l
l
o

w

o
r
l
d

Example: A “checksum” is a numeric quantity representing and confirming the contents of a file. Write a program which reads a string from the user and find its checksum.

```
import java.util.*;
import java.util.Scanner;

public class StringDemo {

	public static void main(String[] args) {

		Scanner S = new Scanner(System.in);
		System.out.println("Enter the 1st string ");
		String s1 = S.nextLine();
		System.out.println("The 1st string: " + s1);

		int i, j, sum = 0;
		for (i=0; i<s1.length(); i++) {
			 j= s1.charAt(i);
			System.out.println("The value of " + s1.charAt(i) + " is " + j);
			sum += s1.charAt(i);
		}
		System.out.println("The checksum is " + sum);
	}
}
```

Output:

```
Enter the 1st string
abc
The 1st string: abc
The value of a is 97
The value of b is 98
The value of c is 99
The checksum is 294
```

Taking Strings Apart with Substrings

- **substring() method** returns a part of the string.

- There are two types of substring methods in java string.

    - *substring(int startIndex)*
        - return all the characters from a given index (startIndex) to end of the string.

    - *substring(int startIndex, int endIndex)*
        - return all the characters between the starting index (startIndex) to ending index (endIndex) of the string.

Example:

```
public class SubStringDemo {

    public static void main(String[] av) {

        String a = "Java is great.";
        System.out.println(a);

        String b = a.substring(5); // b is the String "is great."
        System.out.println("The b substring is " + b);

        String c = a.substring(5,7);// c is the String "is"
        System.out.println("The c substring is " + c);

        String d = a.substring(5,a.length());// d is "is great."
        System.out.println("The d substring is " + d);
    }

}
```

Output:

```
Java is great.
The b substring is is great.
The c substring is is
```

The d substring is is great.

Example: Write a program which reads two strings from the user and concat the substring from 0 to 8 index of the first string with the 0 to 5 indexed of the second string and print the resultant string. The two strings are as follows:

ITER is my College.
ITER is in SOA.

```
import java.util.*;
import java.util.Scanner;

public class StringDemo {

	public static void main(String[] args) {
		// TODO Auto-generated method stub

		Scanner S = new Scanner(System.in);

		System.out.println("Enter the 1st string ");
		String s1 = S.nextLine();
		System.out.println("The 1st string: " + s1);

		System.out.println("Enter the 2nd string ");
		String s2 = S.nextLine();
		System.out.println("The 2nd string: " + s2);

		String a = s1.substring(0,8);
		System.out.println("The 1st substring is: " + a);

		String b = s2.substring(0,5);
		System.out.println("The 2nd substring is: " + b);


		// concat() method concatenates one string
		//to the end of another string.
		a=a.concat(b);
		System.out.println("The resultant string is: " + "  " + a);
	}
```

}

Output:

Enter the 1st string
ITER is my College.
The 1st string: ITER is my College.
Enter the 2nd string
ITER is in SOA.
The 2nd string: ITER is in SOA
The 1st substring is: ITER is
The 2nd substring is: ITER
The resultant string is: ITER is ITER

Breaking Strings Into Words

- Take a string and divide the string into words or tokens.
- Two methods:
  - **Split method:** split()
    - The string split() method breaks a given string around matches of the given regular expression.
      - Regular expressions are " ", ",", "-", ";","."
  - **String Tokenizer method**: StringTokenizer()
    - **The java.util.StringTokenizer class** allows you to break a string into tokens.
    - It doesn't provide the facility to differentiate numbers, quoted strings, identifiers etc.
    - construct a StringTokenizer around your string and call its methods hasMoreTokens() and nextToken().
    - hasMoreElements() and nextElement().

- There are 3 constructors defined in the StringTokenizer class.

  - *StringTokenizer(String str)*
    - Creates StringTokenizer with specified string.

  - *StringTokenizer(String str, String delim)*
    - Creates StringTokenizer with specified string and delimeter.

  - *StringTokenizer(String str, String delim, boolean returnValue)*
    - Creates StringTokenizer with specified string, delimeter and returnValue. If return value is true, delimiter characters are considered to be tokens. If it is false, delimiter characters serve to separate tokens.

Example: split() method

```
import java.util.*;
import java.util.Scanner;

public class StringDemo {

	public static void main(String[] args) {

		Scanner S = new Scanner(System.in);
		System.out.println("Enter the 1st string ");
		String s1 = S.nextLine();

		System.out.println("The 1st string: " + s1);

		for (String word : s1.split(" ")) {		//",", "-", ",",","
			System.out.println(word);
		}
	}
}
```

Output:

```
Enter the 1st string
ITER is My College.
The 1st string: ITER is My College.
ITER
is
My
College.
```

Example: *StringTokenizer(String str)* method

```
import java.util.*;
import java.util.Scanner;

public class StringDemo {

	public static void main(String[] args) {

		Scanner S = new Scanner(System.in);
		System.out.println("Enter the 1st string ");
		String s1 = S.nextLine();

		System.out.println("The 1st string: " + s1);

		//StringTokenizer st = new StringTokenizer(s1);

		StringTokenizer st = new StringTokenizer(s1,", ");


		while (st.hasMoreTokens( ))
			System.out.println("Token: " + st.nextToken( ));
	}

}
```

Output:

```
Enter the 1st string
```

```
ITER, is, my, college.
The 1st string: ITER is my college.
Token: ITER
Token: is
Token: my
Token: college.
```

Example: *StringTokenizer(String str, String delim)*

```
import java.util.*;
public class StringDemo {

	public static void main(String[] args) {

		// TODO Auto-generated method stub

StringTokenizer st = new StringTokenizer("Hello, World|of|Java", ", |");
		while (st.hasMoreElements( ))
		System.out.println("Token: " + st.nextElement( ));
	}

}
```

Output:

```
Hello, World|of|Java

Token: Hello
Token: World
Token: of
Token: Java
```

Example: *StringTokenizer(String str, String delim, boolean returnValue)*

```
import java.util.*;
public class StringDemo {

	public static void main(String[] args) {

		// TODO Auto-generated method stub
```

```
            StringTokenizer st = new StringTokenizer("Hello,
                                                    World|of|Java", ", |", true);
            while (st.hasMoreElements( ))
                System.out.println("Token: " + st.nextElement( ));
        }

}
```

Output:

```
Token: Hello
Token: ,
Token:
Token: World
Token: |
Token: of
Token: |
Token: Java
```

Example: Write a program which reads a string and process it, if consecutive delimiter comes ignore it else print the token.

```
import java.util.*;

public class StrTokDemo4 {
        public final static int MAXFIELDS = 5;
        public final static String DELIM = "|";

        /** Processes one String, returns it as an array of Strings */
        public static String[] process(String line) {
        String[] results = new String[MAXFIELDS]; //array declaration

        // Unless you ask StringTokenizer to give you the tokens,
        // it silently discards multiple null tokens.
        StringTokenizer st = new StringTokenizer(line, DELIM, true);
        int i = 0;
        // stuff each token into the current slot in the array.
        while (st.hasMoreTokens()) {
                String s = st.nextToken();
                if (s.equals(DELIM)) {
```

```
                if (i++>=MAXFIELDS)
                // This is messy: See StrTokDemo4b which uses
                // a List to allow any number of fields.
                throw new IllegalArgumentException("Input line " +
                                                   line + " has too many fields");
                continue;
            }
            results[i] = s;
        }
        return results;
    }

    public static void printResults(String input, String[] outputs) {
        System.out.println("Input: " + input);
        int i=0;
        for (String s : outputs) {
            System.out.println("Output " + i + " was: " + s);
            ++i;
        }
}

        public static void main(String[] args) {
            printResults("ABCD|BC|BC|D ", process("ABCD|BC|BC|D"));
            printResults("AB|||C|D", process("AB|||C|D"));
            printResults("AB|||D|E", process("AB|||D|E"));
        }

}
```

Output:

Input: ABCD|B|C|D
Output 0 was: ABCD
Output 1 was: BC
Output 2 was: BC
Output 3 was: D
Output 4 was: null
Input: AB|||C|D
Output 0 was: AB
Output 1 was: null

Output 2 was: null
Output 3 was: C
Output 4 was: D
Input: AB|||D|E
Output 0 was: AB
Output 1 was: null
Output 2 was: null
Output 3 was: D
Output 4 was: E

Example: Write a program using java to read a string and find the number of words present in that string.

```
import java.util.*;
import java.util.Scanner;

public class StringDemo {

	public static void main(String[] args) {

		Scanner S = new Scanner(System.in);
		System.out.println("Enter the 1st string ");
		String s1 = S.nextLine();

		System.out.println("The 1st string: " + s1);

		StringTokenizer st = new StringTokenizer(s1);
		//StringTokenizer st = new StringTokenizer(s1," ");

		int i=0;
		while (st.hasMoreTokens( )) {
			System.out.println("Token: " + st.nextToken( ));
			 ++i;
		}

		System.out.println("Total number of words are " + i);

	}

}
```

Output:

Enter the 1st string
ITER is my college.
The 1st string: ITER is my college.
Token: ITER
Token: is
Token: my
Token: college.
Total number of words are 4

Example: Write a program to read a string whose words are separated by ',' find the number of words and number of ',' present in that string. The input string is: Hello, World,of,Java.

```
import java.util.*;
import java.util.Scanner;

public class StringDemo {

        public static void main(String[] args) {

                final String DELIM = ",";
                Scanner S = new Scanner(System.in);
                System.out.println("Enter the 1st string ");
                String s1 = S.nextLine();

                System.out.println("The 1st string: " + s1);

                int i=0,j=0;
                StringTokenizer st = new StringTokenizer(s1, ",", true);
                while (st.hasMoreElements( )) {
                        Object s = st.nextElement();
                        System.out.println("Token: " + s);
                        if(s.equals(DELIM)) {
                                ++i;
                        }
                        else
                                ++j;
```

```
                }

                System.out.println("Total number of Delimeter are " + i);
                System.out.println("Total number of Words are " + j);
        }
}
```

Output:

```
Enter the 1st string
Hello, World,of,Java.
The 1st string: Hello, World,of,Java.
Token: Hello
Token: ,
Token:  World
Token: ,
Token: of
Token: ,
Token: Java.
Total number of Delimeter are 3
Total number of Words are 4
```

Java StringBuffer class

- The Java StringBuffer class is used to create mutable (modifiable) string.
- The StringBuffer class in java is same as the String class except it is mutable i.e. it can be changed.

- The StringBuffer class is Synchronized .
    - Java StringBuffer class is thread-safe, i.e., multiple threads cannot access it simultaneously. So it is safe and will result in an order.
- Important methods of StringBuffer class are

    1) StringBuffer append() method

    2) StringBuffer insert() method

    3) StringBuffer replace() method

    4) StringBuffer delete() method

    5) StringBuffer reverse() method

1) StringBuffer append() method

The append() method concatenates the given argument with this string.

Example:

```
class StringBufferExample{
    public static void main(String args[]){
        StringBuffer sb=new StringBuffer("Hello ");
        System.out.println(sb);    //prints Hello
        sb.append("Java ");                //now original string is changed
        sb.append("is language");
        System.out.println(sb);    //prints Hello Java  is language
    }
}
```

Output:

Hello Java is language

2) StringBuffer insert() method

The insert() method inserts the given string with this string at the given position.

Example:

```
class StringBufferExample2{
    public static void main(String args[]){
        StringBuffer sb=new StringBuffer("Hello ");
        sb.insert(1,"Java");          //now original string is changed
        System.out.println(sb);    //prints HJavaello
    }
}
```

Output:

HJavaello

3) StringBuffer replace() method

The replace() method replaces the given string from the specified beginIndex and endIndex. The syntax is :

replace(int beginIndex, int endIndex, string String)

Example:

```
class StringBufferExample3{
    public static void main(String args[]){
        StringBuffer sb=new StringBuffer("Hello");
        sb.replace(1,3,"Java");
```

```
        System.out.println(sb);                    //prints HJavalo
    }
}
```

Output:

```
HJavalo
```

4) StringBuffer delete() method

The delete() method of StringBuffer class deletes the string from the specified beginIndex to endIndex.

delete(int beginIndex, int endIndex)

Example:

```
class StringBufferExample4{
    public static void main(String args[]){
        StringBuffer sb=new StringBuffer("Hello");
        sb.delete(1,3);
        System.out.println(sb);    //prints Hlo
    }
}
```

Output:

Hlo

5) StringBuffer reverse() method

The reverse() method of StringBuilder class reverses the current string.

```
class StringBufferExample5{
```

```
    public static void main(String args[]){
        StringBuffer sb=new StringBuffer("Hello");
        sb.reverse();
        System.out.println(sb);    //prints olleH
    }
}
```

Output:

olleH

Java StringBuilder class

- The Java StringBuilder class is used to create mutable (modifiable) string.
- The Java StringBuilder class is same as StringBuffer class except that it is non-synchronized.
- It has been available since JDK 1.5.
- Important methods of StringBuilder class are

    1) StringBuilder append() method

    2) StringBuilder insert() method

    3) StringBuilder replace() method

    4) StringBuilder delete() method

    5) StringBuilder reverse() method

1) StringBuilder append() method

The StringBuilder append() method concatenates the given argument with this string.

Example:

```
class StringBuilderExample{
    public static void main(String args[]){
        StringBuilder sb=new StringBuilder("Hello ");
        sb.append("Java");//now original string is changed
        System.out.println(sb);//prints Hello Java
    }
}
```

Output:

```
Hello Java
```

Example:

```
public class StringBuilderDemo {
    public static void main(String[] args) {
        //using + operator
        String s1 = "Hello" + ", " + "World";
        System.out.println(s1);

        // Build a StringBuilder, and append some things to it.
        StringBuilder sb2 = new StringBuilder();
        sb2.append("Hello");
        sb2.append(',');
        sb2.append(' ');
        sb2.append("World");

        // Get the StringBuilder's value as a String, and print it.
        String s2 = sb2.toString();
        System.out.println(s2);
```

```
        // Now do the above all over again, but in a more
        // concise (and typical "real-world" Java) fashion.
        System.out.println(
        new StringBuilder()
        .append("Hello")
        .append(',')
        .append(' ')
        .append("World"));
    }

}
```

Output:

```
Hello, World
Hello, World
Hello, World
```

2) StringBuilder insert() method

The StringBuilder insert() method inserts the given string with this string at the given position.

Example:

```
class StringBuilderExample2{

    public static void main(String args[]){

        StringBuilder sb=new StringBuilder("Hello ");

        sb.insert(1,"Java");          //now original string is changed

        System.out.println(sb);              //prints HJavaello

    }

}
```

Output:

```
HJavaello
```

3) StringBuilder replace() method

The StringBuilder replace() method replaces the given string from the specified beginIndex and endIndex.

Example:

```
class StringBuilderExample3{

    public static void main(String args[]){

            StringBuilder sb=new StringBuilder("Hello");

            sb.replace(1,3,"Java");

            System.out.println(sb);    //prints HJavalo

    }

}
```

Output:

HJavalo

4) StringBuilder delete() method

The delete() method of StringBuilder class deletes the string from the specified beginIndex to endIndex.

Example:

```
class StringBuilderExample4{

    public static void main(String args[]){

            StringBuilder sb=new StringBuilder("Hello");

            sb.delete(1,3);

            System.out.println(sb);    //prints Hlo
```

```
    }
}
```

Output:

Hlo

5) StringBuilder reverse() method

The reverse() method of StringBuilder class reverses the current string.

Example:

```
class StringBuilderExample5{
    public static void main(String args[]){
        StringBuilder sb=new StringBuilder("Hello");
        sb.reverse();
        System.out.println(sb);//prints olleH
    }
}
```

Output:

olleH

Example: Write a program using a StringBuilder, consider the need to convert a list of items into a comma-separated list, while avoiding getting an extra comma after the last element of the list. (NOTE: Using split() method).

```
public class StringBuilderDemo {
    public static void main(String[] args) {
        String SAMPLE_STRING = "ITER is My College.";
```

```
                System.out.println("The Input String is " + SAMPLE_STRING);

                StringBuilder sb1 = new StringBuilder();

                for (String word : SAMPLE_STRING.split(" ")) {

                        if (sb1.length() > 0) {
                                sb1.append(", ");
                        }
                        sb1.append(word);
                }
                System.out.println("The Input String is " + sb1);
        }
}
```

Output:

The Input String is ITER is My College.
The Output String is ITER, is, My, College.

Example: Write a program using a StringBuilder, consider the need to convert a list of items into a comma-separated list, while avoiding getting an extra comma after the last element of the list. (NOTE: Using StringTokenizer() method).

```
import java.util.*;
public class StringBuilderDemo {

        public static void main(String[] args) {

                String SAMPLE_STRING = "ITER is My College.";
                System.out.println("The Input String is " + SAMPLE_STRING);

                // Method using a StringTokenizer
                StringTokenizer st = new StringTokenizer(SAMPLE_STRING);

                StringBuilder sb3 = new StringBuilder();

                while (st.hasMoreElements()) {
```

```
                    sb3.append(st.nextToken());

                    if (st.hasMoreElements()) {

                                sb3.append(", ");
                    }
            }
            System.out.println(sb3);
        }
}
```

Output:

The Input String is ITER is My College.
The Output String is ITER, is, My, College.

Reversing a String by Character

Example: reverse a string by character easily, using charAt() method.

```
import java.util.*;
public class ReverseString {

        public static void main(String[] args) {
                // TODO Auto-generated method stub

                    String original, reverse = "";
                    Scanner sc = new Scanner(System.in);

                    System.out.println("Enter a string to reverse");
                    original = sc.nextLine();

                    int length = original.length();

                    for (int i = length - 1 ; i >= 0 ; i--)
                      reverse = reverse + original.charAt(i);

                    System.out.println("Reverse of the string: " + reverse);
```

```
    }
}
```

Output:

```
Enter a string to reverse
ITER is My College.
Reverse of the string: .egelloC yM si RETI
```

Example: To reverse the characters in a string, using the StringBuilder reverse() method.

```
import java.util.*;
public class ReverseString {

    public static void main(String[] args) {

        String original, reverse = "";

        Scanner sc = new Scanner(System.in);
        System.out.println("Enter a string to reverse");
        original = sc.nextLine();

       StringBuilder sb=new StringBuilder(original);
       System.out.println("Reverse of the string: " + sb.reverse());
    }
}
```

Output:

```
Enter a string to reverse
ITER is My College.
Reverse of the string: .egelloC yM si RETI
```

Reversing a String by Word

- To reverse the string by word is a 2 step process:

- From the beginning of the string each word has been taken and adds into a Stack.

- Then retrieve the words from stack in LIFO order.

Example:

```
import java.util.*;

public class ReverseString {

        public static void main(String[] args) {

                String original, revword="";

                Scanner sc = new Scanner(System.in);
                System.out.println("Enter a string to reverse");
                original = sc.nextLine();

               // Put it in the stack frontwards
               Stack<String> myStack = new Stack<>();              //Create the stack

               StringTokenizer st = new StringTokenizer(original);

               while (st.hasMoreTokens()) {
                        String word = st.nextToken();
                        myStack.push(word);                        //Insert into the stack
               }

               // Print the stack backwards
               System.out.print("" + original + "" + " backwards by word is:\n\t\"");

               while (!myStack.empty()) {
                        String revword = myStack.pop();        //Retrieve from the stack
                        System.out.print(revword);
                        System.out.print(' ');
               }
               System.out.println("");
        }
}
```

Output:

"ITER is My College." backwards by word is:
"College. My is ITER "

Java String equals() Method

- The **java string equals()** method compares the two given strings based on the content of the string.

- If any character is not matched, it returns false. If all characters are matched, it returns true.

Example:

```
public class StringEquals {

    public static void main(String[] args) {

        String s1="java";
        String s2="java";
        String s3="JAVA";
        String s4="python";

        System.out.println(s1==s2);       //true because content and case is same

        System.out.println(s1.equals(s2));      //true because content and case is
                                                  same
        System.out.println(s1.equals(s3));      //false because case is not same

        System.out.println(s1.equals(s4));     //false because content is not same

    }

}
```

Output:

```
true
true
false
```

false

Java String equalsIgnoreCase() Method

- The **String equalsIgnoreCase()** method compares the two given strings on the basis of content of the string irrespective of case of the string.
- It is like equals() method but doesn't check case.
- If any character is not matched, it returns false otherwise it returns true.

Example:

```
public class StringEquals {

    public static void main(String[] args) {

        String s1="java";
        String s2="java";
        String s3="JAVA";
        String s4="python";

        System.out.println(s1==s2);

        System.out.println(s1.equalsIgnoreCase(s2));

        System.out.println(s1.equalsIgnoreCase(s3));

        System.out.println(s1.equalsIgnoreCase(s4));

    }

}
```

Output:

```
true
true
true
```

false

Java String trim()

- The **java string trim()** method eliminates leading and trailing spaces.
- The unicode value of space character is '\u0020'.
- The trim() method in java string checks this unicode value before and after the string, if it exists, then removes the spaces and returns the omitted string.

Example:

```
public class StringTrim {

	public static void main(String[] args) {

		String s1 ="  hello java string   ";
		 System.out.println("Length of String before trim: " + s1.length());
		System.out.println(s1);			//Without trim()

		String tr = s1.trim();
		System.out.println("Length of String after trim: " +tr.length());
		System.out.println(tr);			//With trim()
	}
}
```

Output:

```
Length of String before trim: 22
  hello java string
Length of String after trim: 17
hello java string
```

Entering Nonprintable Characters

- Put nonprintable characters into strings.

To get:	Use:	Notes
Tab	\t	
Linefeed (Unix newline)	\n	The call System.getProperty("line.separator") will give you the platform's line end.
Carriage return	\r	
Form feed	\f	
Backspace	\b	
Single quote	\'	
Double quote	\"	
Unicode character	\u*NNNN*	Four hexadecimal digits (no \x as in C/C++). See *http://www.unicode.org* for codes.
Octal(!) character	*NNN*	Who uses octal (base 8) these days?
Backslash	\\	

Example:

```
public class StringEscapes {
	public static void main(String[] argv) {
		System.out.println("Java Strings in action:");
		System.out.println("A tab key: \t(what comes after)");
		System.out.println("A newline: \n(what comes after)");
		System.out.println("A UniCode character: \u0207");
		System.out.println("A backslash character: \\");
	}
}
```

Output:

```
Java Strings in action:
A tab key:	(what comes after)
A newline:
(what comes after)
A UniCode character: ?
```

A backslash character: \

Expanding and Compressing Tabs

- Convert space characters to tab characters in a string, or vice versa.

Example: Write a program to replace all space in a string by tabs. Note: Here we have used **replaceall()** method. The **replaceall()** requires two parameters, both are string types.

```
import java.util.*;
public class StringReplace {

	public static void main(String[] args) {

			String str2 = "ITER is My College";
			System.out.println("Input string: " + str2);


			str2 = str2.replaceAll("\\s","\t");
			System.out.println("Output string: "+ str2);
	}
}
```

Output:

```
Input string: ITER is My College
ITER is	My	College
```

Example: Write a program to replace all tab in a string by spaces.

```
import java.util.*;
public class StringReplace {

	public static void main(String[] args) {

		String str2 = "ITER	is	My	College";
		System.out.println("Input string: "+ str2);
```

```
        str2 = str2.replaceAll("\\t"," ");
        System.out.println("Output string: "+ str2);

    }
}
```

Output:

```
Input string: ITER        is      My    College
Output string: ITER is My College
```

Example: Write a program to //replaces all occurrences of "a" to "e".

```
public class ReplaceAllExample1{

    public static void main(String args[]){

        String s1="java is a very good language";

        String replaceString=s1.replaceAll("a","e");

        System.out.println(replaceString);

    }

}
```

Output:

```
jeve is e very good lenguege
```

Example: Write a program to replaces particular word by using replace all.

```
public class ReplaceAllExample2{

    public static void main(String args[]){

        String s1="C is a very good language";

        String replaceString=s1.replaceAll("C","java");
```

```
        System.out.println(replaceString);
    }
}
```

Output:

java is a very good language

Soundex Name Comparisons

- The Soundex algorithm was developed by Robert Russell in 1910 for the words in English.
- The main principle behind this algorithm is that consonants are grouped depending on the ordinal numbers and finally encoded into a value against which others are matched.
- It aims to find a code for every word by above process which is called soundex code.

- The Soundex code is 4-character code:
    - the letter is the first letter of the name, and
    - the digits encode the remaining consonants.

Aalgorithm to find soundex code is as below:

1. Retain the first letter of the name.
2. Change all occurrence of following letter to zero.

    a, e, i, o, u, y, h, w → 0

3. Replace consonants with digits as follows:

    b, f, p, v → 1
    c, g, j, k, q, s, x, z → 2
    d, t → 3
    l → 4

m, n → 5
r → 6

4. Remove all pairs of consecutive digits.
5. Remove all zeros from the resulting string.
6. If the resultant code size is more than 4, then just retain the first 4 by truncating rest code.
7. If the resultant code size is less than 4, then append with zeros.

Example: Soundex Algorithm example.

```
public class Soundex1 {

    public static String getCode(String s) {

        char[] x = s.toUpperCase().toCharArray();

         char firstLetter = x[0];

        //RULE [ 2 ]
        //Convert letters to numeric code
        for (int i = 0; i < x.length; i++)
        {
            switch (x[i])
            {
                case 'B':
                case 'F':
                case 'P':
                case 'V': {
                                x[i] = '1';
                                break;
                            }

                case 'C':
                case 'G':
                case 'J':
                case 'K':
                case 'Q':
                case 'S':
                case 'X':
```

```
                case 'Z': {
                                x[i] = '2';
                                break;
                            }

                case 'D':
                case 'T': {
                                x[i] = '3';
                                break;
                            }

                case 'L': {
                                x[i] = '4';
                                break;
                            }

                case 'M':
                case 'N': {
                                x[i] = '5';
                                break;
                            }

                case 'R': {
                                x[i] = '6';
                                break;
                            }

                default: {
                                x[i] = '0';
                                break;
                            }
        }
}


  //Return first letter of word
  String output = " " + firstLetter;

  // consonants coding by removing duplicate digits and zero
  for (int i = 1; i < x.length; i++)
```

```
                if (x[i] != x[i - 1] && x[i] != '0')

                    output += x[i];


            //Pad with 0's or truncate

                    output = output + "0000";


            return output.substring(0, 4);
    }

        public static void main(String[] args) {
                // TODO Auto-generated method stub
          String name1 = "beer";
          String name2 = "bear";
          String name3 = "bearer";

          System.out.println(Soundex1.getCode(name1) + ' ' + name1);
          System.out.println(Soundex1.getCode(name2) + ' ' + name2);
          System.out.println(Soundex1.getCode(name3) + ' ' + name3);
        }
}
```

Output:

```
B600 beer
B600 bear
B660 bearer
```

Example: Soundex Algorithm example.

```
public class Soundex {
        //static boolean debug = false;
        /* Implements the mapping
        * from: AEHIOUWYBFPVCGJKQSXZDTLMNR
        * to: 00000000111122222222334556
        */
```

```
public static final char[] MAP = {
//A   B   C   D   E   F   G   H   I   J   K   L   M
 '0',  '1', '2',  '3', '0', '1', '2', '0', '0','2','2',  '4',  '5',
//N   O   P   Q   R   S   T   U   V   W   X   Y   Z
  '5', '0', '1', '2',  '6', '2', '3', '0',  '1',  '0', '2',  '0', '2'};

public static String soundex(String s) {
        // Algorithm works on uppercase (mainframe era).
        String t = s.toUpperCase();
        StringBuffer res = new StringBuffer();
        char c, prev = '?', prevOutput = '?';
        // Main loop: find up to 4 chars that map.

        for (int i=0; i<t.length() && res.length() < 4 &&
        (c = t.charAt(i)) != ','; i++) {
        // Check to see if the given character is alphabetic.
        // Text is already converted to uppercase. Algorithm
    // only handles ASCII letters, do NOT use Character.isLetter()!
   // Also, skip double letters.
        if (c>='A' && c<='Z' && c != prev) {
                prev = c;
        // First char is installed unchanged, for sorting.
                if (i==0) {
                        res.append(c);
                }
                else
                {
                        char m = MAP[c-'A'];
                        //if (debug) {
                        //      System.out.println(c + " --> " + m);
                        //      }
                        if (m != '0' && m != prevOutput) {
                                res.append(m);
                                prevOutput = m;
                        }
                   }
             }
        }
        if (res.length() == 0)
        return null;
```

```
            for (int i=res.length(); i<4; i++)
                res.append('0');
            return res.toString();
            }

        public static void main(String[] args) {
            // TODO Auto-generated method stub

                String[] names = {
                "Darwin, Ian",
                "Davidson, Greg",
                "Darwent, William",
                "Derwin, Daemon",
                "Der"
                };

                for (String name : names) {
                    System.out.println(Soundex.soundex(name) + ' ' + name);
                }
        }
}
```

Output:

D650 Darwin, Ian
D132 Davidson, Greg
D653 Darwent, William
D650 Derwin, Daemon
D600 Der

4. Write a program to enter a string and count the frequency of each character present in it.

```
public class Frequency

{

    public static void main(String[] args) {
```

```
  String str = "picture perfect";
  int[] freq = new int[str.length()];
  int i, j;
//Converts given string into character array
  char string[] = str.toCharArray();
   for(i = 0; i <str.length(); i++) {
      freq[i] = 1;
      for(j = i+1; j <str.length(); j++) {
         if(string[i] == string[j]) {
            freq[i]++;
           //Set string[j] to 0 to avoid printing visited character
            string[j] = '0';
         }
      }
  }

  //Displays the each character and their corresponding frequency
  System.out.println("Characters and their corresponding frequencies");
  for(i = 0; i <freq.length; i++) {
      if(string[i] != ' ' && string[i] != '0')
         System.out.println(string[i] + "-" + freq[i]);
```

```
        }

    }

}
```

Output:

Characters and their corresponding frequencies
p-2
i-1
c-2
t-2
u-1
r-2
e-3
f-1

6. Write a program to enter N number of strings, arrange them in ascending order.

```
java.util.Scanner;
public class Ch3Ex6 {

        public static void main(String[] args) {
                int count;
                String temp;
                Scanner scan = new Scanner(System.in);

               System.out.print("Enter number of strings: ");
               count = scan.nextInt();

      System.out.println("Enter the" + count + "Strings one by one:");
      String str[] = new String[count];
      Scanner scan2 = new Scanner(System.in);

      for(int i = 0; i < count; i++)
      {
         str[i] = scan2.nextLine();
```

```
        }
        scan.close();
        scan2.close();

        //Sorting the strings
        for (int i = 0; i < count; i++)
        {
            for (int j = i + 1; j < count; j++) {
                if (str[i].compareTo(str[j])>0)
                {
                    temp = str[i];
                    str[i] = str[j];
                    str[j] = temp;
                }
            }
        }

        //Displaying the strings after sorting
        System.out.println("Strings in Sorted Order:");
        for (int i = 0; i <= count - 1; i++)
        {
            System.out.println(str[i]);
        }

    }

}
```

Output:

```
Enter number of strings: 5
Enter the 5 Strings one by one:
cbi
bca
mca
xyz
aaa
Strings in Sorted Order:
aaa
bca
cbi
```

mca
xyz

10. The global replace function is a string-processing algorithm found in every word processor. Write a method that takes three String arguments: a document, a target string, and a replacement string. The method should replace every occurrence of the target string in the document with the replacement string. For example, if the document is "To be or not to be, that is the question," and the target string is "be,", and the replacement string is "see," the result should be, "To see or not to see, that is the question."

Solutions:

```
import java.util.*;
public class StringProblem {

	public static void main(String[] args) {
		// TODO Auto-generated method stub
		Scanner S = new Scanner(System.in);
		System.out.println("Enter the original string ");
		String s1 = S.nextLine();

		//StringBuilder s2 = new StringBuilder(s1);
		System.out.println("Enter the target string: ");
		String target = S.nextLine();
		System.out.println("Enter the replacement string ");
		String replacement = S.nextLine();

		System.out.println(" ");
		System.out.println("The original string is: "+ s1);
		System.out.println("The target string is: "+ target);
		System.out.println("The replacement string is: "+
                                                        replacement);

		String result = s1.replaceAll(target, replacement);
		System.out.println("The resultant string is: " + result);

	}

}
```

Output:

Enter the original string
To be or not to be, that is the questions.
Enter the target string:
be
Enter the replacement string
see

The original string is: To be or not to be, that is the questions.

The target string is: be

The replacement string is: see

The resultant string is: To see or not to see, that is the questions.

12. Write a method that converts its String parameter so that letters are written in blocks five characters long. For example, consider the following two versions of the same sentence:

This is how we would ordinarily write a sentence.

Thisi showw ewoul dordi naril ywrit easen tence

Solution:

```
import java.util.Scanner;
import java.util.*;
public class NumberToWord {

	public static void main(String[] args) {
		// TODO Auto-generated method stub
		Scanner S = new Scanner(System.in);
		System.out.println("Enter the original string ");
		String s1 = S.nextLine();
		System.out.println("The original string is: "+ s1);

		String s2 = s1.replaceAll("\\s", "");
	System.out.println("After replacing space, string is: "+ s2);
	int count = 0;
```

```
        //StringBuilder s3 = new StringBuilder();
        //s3.append(s2);

        System.out.print("the resultant string is:  ");
        char ch;
            for(int i=0;i<=s2.length()-1;i++)
            {
                    count++;
                    if(count == 6)
                    {
                            ch = ' ';
                            System.out.print(ch);
                            count=1;
                            ch = s2.charAt(i);
                            System.out.print(ch);
                    }
                    else
                    {
                            ch = s2.charAt(i);
                            System.out.print(ch);
                    }
            }
    }
}
```

Output:

The original string is: This is how we would ordinarily write a sentence.

After replacing space, string is: Thisishowwewouldordinarilywriteasentence.

The resultant string is: Thisi showw ewoul dordi naril ywrit easen tence .

25. Write an program to print all permutations of a given String in Java. For example, if given input is "123" then your program should print all 6 permutations e.g. “123”, “132”, “213”, “231”, “312” and “321”.

Solution:

```
import java.util.Scanner;
```

```
public class Ch3Ex25 {
        //Function for swapping the characters at position I with character at position
j
   public static String swapString(String a, int i, int j) {
      char[] b =a.toCharArray();
      char ch;
      ch = b[i];
      b[i] = b[j];
      b[j] = ch;
      return String.valueOf(b);
   }

   public static void main(String[] args)
   {
       Scanner scan = new Scanner(System.in);
      System.out.print("Enter the strings: ");
      String str = scan.nextLine();

      int len = str.length();
      System.out.println("All the permutations of the string are: ");
      generatePermutation(str, 0, len);
   }

   //Function for generating different permutations of the string
   public static void generatePermutation(String str, int start, int end)
   {
      //Prints the permutations
      if (start == end-1)
         System.out.println(str);
      else
      {
         for (int i = start; i < end; i++)
         {
            //Swapping the string by fixing a character
            str = swapString(str,start,i);
            //Recursively calling function generatePermutation() for rest of the
characters
            generatePermutation(str,start+1,end);
            //Backtracking and swapping the characters again.
            str = swapString(str,start,i);
```

```
        }
    }
  }
}
```

Output:

```
Enter the strings: 123
All the permutations of the string are:
123
132
213
231
321
312
```

26. Write a Java program which will take a String input and print out a number of vowels and consonants on that String. For example, if the input is "Java" then your program should print "2 vowels and 2 consonants".

Solution:

```
import java.util.Scanner;
public class Ch3Ex26 {
   public static void main(String[] args) {
        Scanner scan = new Scanner(System.in);
      System.out.print("Enter the strings: ");
      String str = scan.nextLine();

      int vowels = 0, consonants = 0, digits = 0, spaces = 0;
      str = str.toLowerCase();

      for(int i = 0; i < str.length(); ++i)
      {
         char ch = str.charAt(i);
         if(ch == 'a' || ch == 'e' || ch == 'i' || ch == 'o' || ch == 'u')
         {
            ++vowels;
         }
         else if((ch >= 'a'&& ch <= 'z'))
         {
            ++consonants;
```

```
            }
            else if( ch >= '0' && ch <= '9')
            {
               ++digits;
            }
            else if (ch ==' ')
            {
               ++spaces;
            }
        }
        System.out.println("Vowels: " + vowels);
        System.out.println("Consonants: " + consonants);
        System.out.println("Digits: " + digits);
        System.out.println("White spaces: " + spaces);
    }
}
```

Output:

```
Enter the strings: java
Vowels: 2
Consonants: 2
Digits: 0
White spaces: 0
```