will contain
all the string

→ String objects are stored in heap memory or String constant pool memory based on your logic.

→ strings are immutable [we can't able to change].

we don't have in python.

== vs equals() → Content of 2 strings.

↓

Comparison of address 2 string

→ if we want to add any element in string

String s="palle";

s=s+"tech"; system.out.println(s);

* 1-> array example without loops:-

Arrays are stored in heap memory.

```
class Result
{
public static void main(string[] args)
{
int marks[] = new int[5];
marks [0]=55;
marks [1]= 65;
marks [2] = 75;
marks (3) = 85;
marks [4]=95;
System.out.println ("marks of students");
System.out.println (marks [0]);
System.out.println (marks (1));
S.O.P (marks (2));
S.O.P (marks (3));
S.O.P (marks [4]);
}
}
```

```
public class Result{
    public static void main(String()
    int marks[]=new int[5];
    marks[0]=55;
    marks[1]=65;
    marks[2]=75;
    marks[3]=85;
    marks[4]=95;
    S.o.p("marks of students:");
    for(int i=0; i<marks.length; i++){
        System.out.println(marks[i]);
```

* Input from User:-

```
import java.util.Scanner;
public class Result{
    public static void main(String[] args){
        Scanner sc= new Scanner(system.in);
        S.o.p("please enter size of array");
        int a = sc.nextInt();
        int marks()=new int[a];
        System.out.println("please enter marks
                of students");
        for(int i=0; i<marks.length; i++){
            marks[i]=sc.nextInt();
        }
        System.out.println("marks of students");
        for(int i=0; i<marks.length; i++){
            System.out.println(marks[i]);
```

2-dimensional array:-

```
import java.util.Scanner;
public class NameApp
{
public static void main(String[] args)
Scanner sc= new Scanner(System.in);
System.out.println("Please enter no.of classes");
int row=sc.nextInt();
S.O.P("Enter no.of ~~class~~ students in each class");
int col=sc.nextInt();
sc.nextLine();
String names[][]=new String[row][col];
//storing names in array
S.O.P("please enter names");
for(int i=0;i<names.length;i++){
for(int j=0;j<names[i].length;j++){
names[i][j]=sc.nextLine();
}
}
Printing names:
for(int i=0;i<names.length;i++){
for(int j=0;j<names[i].length;j++)
S.O.P(names[i][j]+"\n");
}
System.out.println();
}
}
```

all the string

→ String objects are stored in heap memory or String constant pool memory based on your logic

→ Strings are immutable [we can't able to change].

we don't have in python

== vs equals() → Content of 2 strings.

↓

Comparison of address 2 strings

→ If we want to add any element in string

String s="palle";

s=s+"tech"; system.out.println(s

String builder:-

→ we can modify the string data here it is considered as mutable. Here this are stored only in heap.

→ Normally string is a sequence of charauters whereas in Java string is an object that represents a sequence of charauters.

→ the java.lang.string class is used to creat a string object.

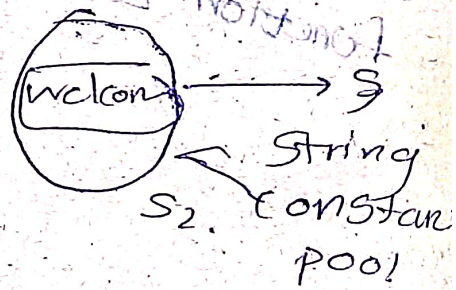
2-ways to create string objects:-

1. By string literal
2. By new keyword.

String s="welcome";

Each time you create a string literal, the JVM checks the string constant pool first. If the string already exists in the pool, what is there that will return, otherwise newone will create

String s1="welcome";
String s2="welcome"; //



Concatenation:-

String s="sachin";
s=s.concat("Tendulkar");
System.out.println(s);

=> Sachin Tendulkar

why string is immutable:-

1.- ClassLoader:-

A classloader in Java uses a string object as an argument. Consider, if the string object is modifable, the value

might be changed
to be loaded might be different.
→ To avoid this kind of interpretation string is immutable.

② Thread safe:-
As the string object is immutable we don't have to take care of the synchronization that is required while sharing an object across multiple threads.

③ Security:-
As we have seen, string is immutable, consider for example, if I banking sector, the username & password cannot be modified by any other intruder.

④ Heap space:-
When we create variable, it will store in string pool management, if it is already there. then new one is not create.

→ Why string class is Final:-
Because no-one can override the methods of the string class.

There are 3-ways to compare string in Java:-
By using equals() method.
By using == operator.
By compareTo() method.

By using equals() method:-
equals() method compares the original content of the string. It compares values of string for equality.
equals
equalsIgnoreCase

(Compare To)
The string class compareTo() method compares value lexicographically and returns an integer value
↓ A string which comes afterwards is said to be lexigraphically greater
Means in which order or preferable order in which words appear.

Suppose s1==s2 The method returns 0
s1>s2 : The method returns positive
s1<s2 : The method returns a negative value.

String Concatenation:

By + String Concatenation operator
By Concat().

1. String Concatenation:

```
s= "sachin" + "Tendulkar";
System.out.println(s);
}
```

2. Class TestStringConcatenation2 {
Public static void main(String args[]){
String s=50+30+"sachin"+40+40;
System.out.println(s)

```
V
80sachin4040.
```

Class TestStringConcatenation3 {
public static void main(String args[]){
String s1="sachin";
String s2="Tendulkar";
String s3=s1.Concat(s2);
S.O.P(s3);
}
}

All data is stored in harddisk to CPU.
But harddisk works very slowly, whereas CPU works very quickly, so if we send data from harddisk to CPU, it sends slowly, so here we are introducing RAM (temporary data). so harddisk sends to RAM, that sends to CPU.
↓
permanent storage.

String methods:-

1. length():-
int len = str.length();
2. charAt()
char ch = str.charAt(1);
3. substring()
↓
substring(1,4);
4. contains()
contains("ell");
5. equals()
6. equalsIgnoreCase()
7. toUpperCase()
8. toLowerCase()
9. trim() → removes leading and trailing whitespace.
10. replace
11. replaceAll
12. startsWith ⇒ boolean starts = str.startsWith("He");
13. endsWith ⇒ boolean ends = str.endsWith("lo");
14. indexOf → Returns -1 if not found.
15. lastIndexOf()
16. split()
17. valueOf()
converts various data type (int, char, boolean) to a string.
String s = String.valueOf(123);
23. join(): → join ans array of iterable of strings with a given delimiter.
String joined = String.join(", ", "apple", "banana", "cherry");
24. compareTo() → 0 (both are equal)
+ve (lexigraphically greater)
-ve (lexigraphically lesser)
21. isEmpty()
25. matches():- check if the string matches the specified regular expression
boolean matches = str.matches(".*ell.*");
26. format :- Returns a formatted string using the specified format string and arguments
String formatted = String.format("Hello %s", "world");

vote");

}

In strings ==
compares address
equals compares
content

Taking i/p from user:-

```
Scanner sc=new Scanner(System.in);
System.out.println("Enter any one character");
char ch=sc.next().charAt(0);
```

↓

this for string which takes so many characters.

charAt(0) is used to get or read character.

Sunday:-

```
class Sunday day{
    public static void main(String() args)
    {
Scanner sc=new Scanner(System.in);
System.out.println("Enter a number");
int a=sc.nextInt();
if(a==1){
System.out.println("sunday");}
```

char[] b = {'a', '*', '3', '?'};

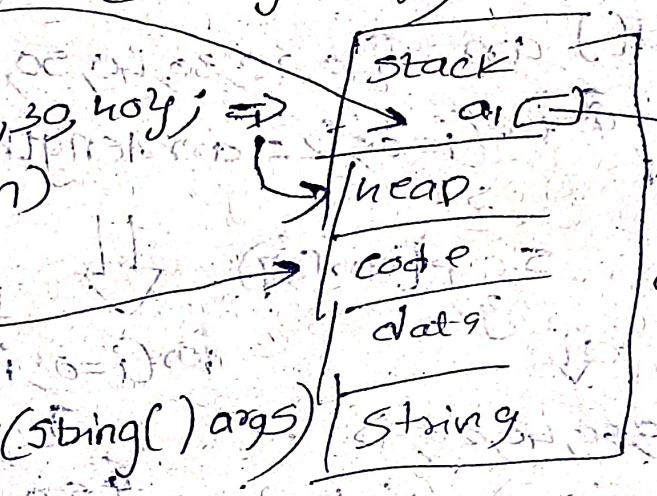
float[] c = {1.5f, 2.5f, 3.5f};

boolean[] c = {true, false, true, false};

long[] c = {4899567891, 9515614689l, 9866852650l, 8331835541, 8367689349};

Arrays are stored in heap memory, arrays are considered as objects in java.

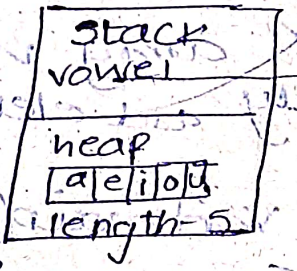
```
class Main {
    public static void main(String[] args) {
        int[] a = {10,20,30,40};
        S.O.P(a.length)
    }
}
```




```
public class Mainclass {
    public static void main(String[] args) {
        char[] vowel = {'a', 'e', 'i', 'o', 'u'};
        System.out.println(vowel.length);
        for(int i=0; i<=4; i++) {
            S.O.P(vowel[i]);
        }
    }
}
```

specific code
generic code

This is called as specific code



Generic code:

```
public class Mainclass {
    public static void main(String[] args) {
        char[] vowel = {'a', 'e', 'i', 'o', 'u'};
        S.O.P(vowel.length);
        for(i=0; i<=vowel.length-1; i++) {
            S.O.P(vowel[i]);
        }
    }
}
```

reverse order

```
for(i=vowel.length-1; i>=0; i--) {
    S.O.P(vowel[i]);
}
```

Store 10, 20, 30, 40, 60, ---100

for(i=10; i<=100; i=i+10)

10, 20, 50, 70, 90

for(i=0; i<100; i=i+20)

100, 80, 60, 40, 20 => for(i=100; i>=20; i=i-20)

20, 40, 60, 80, 100 => for(i=20; i<=100; i=i+20);

→ public class Main {

public static void main(String[] args) {

int[] arr = {10, 20, 30, 40, 50, 60, 810, 80, 90, 100}

for(i=0; i<=arr.length-1; i=i+1)

{

S.O.P[arr[i]

}

100, 80, 60, 40, 20 =>

for(i=100; i>=0; i=i-2)

{

S.O.P(arr[i])

}

10, 30, 50

for(i=0; i<=arr.length-1; i=i+2)

{

S.O.P(arr[i])

}

=> 20, 40, 60, 80

for(i=1; i<=array.length-1; i=i+2)

{

S.O.P(array[i])

}

Create an integer array with 15-elements.

→ print only add elements

Class {

public static void main(String[] args) {

int arr[] = {10, 20, 3, 4, 5, 6, 7, 8, 9, 10, 22, 13, 14, 15, 16}

for(i=0; i<=arr.length-1; i++) {

if(arr[i] % 2 != 0) {

S.O.P(arr[i])

}

}

(2) even index element

public class even {

Creating

String s1 = "palle";
String s2 = new String("palle")

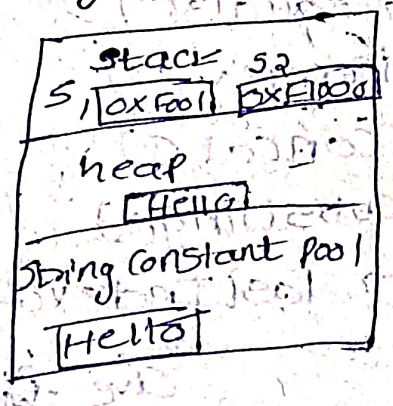
1. With new keyword.
2. Without new keyword.

difference between String with new keyword without new keyword?

```
public class Pupa {
    public static void main(String[] args) {
        String s1 = "Hello";
        String s2 = new String("Hello");
        s.o.p(s1);
        s.o.p(s2);
    }
}
```

→ Arrays are accessed using index

o/p:- Hello → Strings are accessed
Hello using charAt()



The difference between strings and arrays while accessing is that here in strings we can use name only, whereas in arrays we need to give array index.

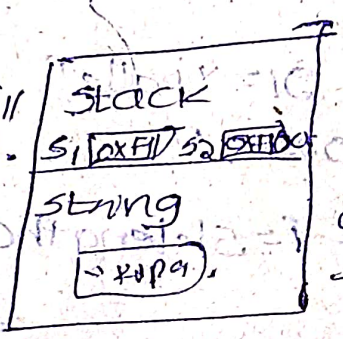
→ The only difference between String with new keyword without new keyword is here in new it will store in heap, whereas normal it store in string constant pool.

→ Internally stored as character arrays.

→ String s1 = "pupa"
String s2 = "pupa"

two objects will be created.

String s1 = new String("palle")
String s2 = new String("palle")



Instead of creating new object it will send same address to second variable.

→ S.C pool will not create a new string object if the same string object is already available in string constant pool, whereas by using new it doesn't happen.

→ A text segment, also known as a code segment.

→ Initialized segment usually called data

→ Stack follows LIFO (Last in First out) Structure.

→ Heap is the segment where dynamic memory allocation takes place.

→ S.o.p (s1.replace('@')); it will change if character presents it will give like that only, otherwise it will

→ indexof()
→ length()
→ equals()
→ isEmpty()
→ concat()
→ substring()
→ lastIndexof()

statement:

system.out.println(indexof('o'))

if it is not in string, then it will return -1

first occurence of the given character

Give last occurence of the character

S.o.println("palle".concat(s1));

String s1="welcome to java class";

for(int i=0; i<=s1.length(); i++)
{
system.out.println(s1.charAt(i));
}

→ String s1="palle"
String s2="";
for(int i=s1.length()-1; i>=0; i--)
{
s2=s2+s1.charat(i);
}

"To find String is palindrome or not"

if (s1.equals(s2))
{
S.o.P("It is a palindrome");
}

→ given string.

→ count the number of words from the given string.

Q) create as string prevention is better than "are" and print total occorence of 'e'.

count how many

② public static void main(string[] args){

String s1 = "Learning python is very Easy"

for(i=0; i<= ~~ter~~ s1.length()-1; i++){

if(i== && s1.charAt(i)==' ' && s1.charAt(i+1) != ' ')

}

Count++

}

}

System.out.println("Count");

Find the given character is present in your string or not?

String str = "Haste makes waste";

for(int i=0; i<str.length(); i++){

if(str.charAt(i)=='a'){

S.O.P("a is present in the string");}

else{

S.O.P("a is not present in the givenstring");

}

⇒ Here so many times the printstatement will print so here we are using if I want to print, which is there then only print.

String str = "Haste makes waste";

int count=0 str Scanner sc = new Scanner(System.in);

for(int ~~i=0~~ ; i<str; char ch = sc.~~charAt~~ next().charAt(0);

for(~~int i=0~~ ; i<b.length()-1; i++){

→ stack + Structure.
→ Heap is allocatio
→

```
if (str.charAt(i) == ch)
{
count++
break;
}
}
if(count==1)
{
S.o.p(ch+" is present in the string");
}
else{
S.o.p(ch+" is not present in the string");
}
```

② String str2 = <your name> Rupa.

o/p:- > RUAP

```
String str = "Rupa", String str2 = "";
for(i=0;i<=str1.length()-1
t = str1.length()
p = t/2;
for(i=0; i<= p; i++){
    str2 = str2 + charAt(i)
for(i=i<=str.length()-1; i--){
    str2 = str2 + charAt(i)
S.o.p(str2);
```

(op)

```
String str = "Deepa";
int mid = str2.length()/2;
S.o.P(mid);
for(int i=0; i<= mid; i++){
    S.o.P(str2.charAt(i));
```

```
for(int i = str2.length()-1 ; i>=mid ; i--)
{
    s.o.print(str2.charAt(i));
}
}
```

→ make first character of every word to capital.

→ replace all constants with @.

→ in your string remove all vowels and print only consonants.

String disadvantages:-

String objects are immutable, if you try to modify e.g. String object, it will create one more string object.

→ programmer should not use string class objects for modification purpose.

→ more you try to modify, more string objects will be created and strings are not meant for modification more memory will get wasted.

String Builder:-

→ predefined class

→ Sun Microsystem Developed.

→ java.lang. package.

→ Why Strings are immutable?

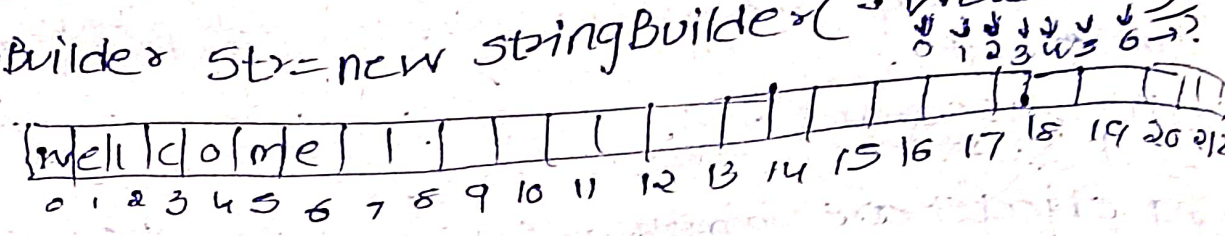
1. performance & memory optimization:- immutable objects can be shared and reused, reducing memory usage and improving performance, when a string is immutable the same memory space can be used by multiple variables.

→ Security:- strings cannot be altered once created which is crucial in contexts, where strings represent sensitive data like usernames, passwords.

→ String Builder objects are stored in heap.
→ StringBuilder comes with a default capacity of 16.
→ String Builder objects are dynamically growable
growthfactor = (previous capacity * 2) + 2
→ String Builder is mutable.
→ StringBuilder objects are not synchronized with multiple threads.

Syntax:-

StringBuilder str = new StringBuilder("welcome");



→ Here ~~for~~ 23 are allocated
→ 16 + 7 [because default 16]
str = str + "To palle technologies");
⇒ then memory will expand.
g.f = (p.c * 2) + 2
= (23 * 2) + 2 ⇒ 48. →

capacity():-
s.o.p(s.capacity); → 23
→ how many charaters it will store.
s.o.p(s.length()); → ~~stored~~ charaters stored [5].

how many

→ adding two stringbuilder.

```
StringBuilder sb = new StringBuilder();
sb.append("palle");
sb.append("tech");
s.o.p(sb);
```

→ using StringBuilder we can modify string data.
→ since StringBuilder are mutable, new strings are not created in heap and hence memory wastage is reduced.

different ways to create strings.

```
class StringDemo
{
public static void main(String arg[])args)
{
String s1= new String("raraj");
System.out.println("s1"+s1);
char ch[]={'P', y, t, h, o, n};
String s2= new String(s2);
S.O.P("s2": +s2);
String s3 = "Html";
S.O.P("s3:"+s3);
}
}
```

→ Duplicates are not allowed in string pool area.
→ Duplicates are allowed outside the string pool area.
→ Here in string pool they are created without using new object.

```
String str= "Rodnest Tech Pvt Ltd";
String s1= str.toUpperCase();
String s2= str.toLowerCase();
String s3= str.substring(8,12); → start at 8 & 12 will be excluded upto 11.
String s3= sb.substring(8); → Here it will start with 8 and continues on till end of text.
String sb= "Rodnest TELL";
int i= sb.length() [no. of characters present in string].
boolean pr= str.contains("T");
char ch[]= str.toCharArray();
String s[]= str.split("-");
String str1= str.concat("_software");
```

contains:- It returns true, if given substring is there, otherwise return= false.

→ String toCharArray:- converts the given string into a character Array.

→ String B a character Array.

→ String B1 . split(delimiter/input):

converts the given string into array by splitting the string according to the input/delimiter.

→ St

- concat() → .concat("input string")

String str = "Kodnest Tech";

String s1 = str + " pvt Ltd";

Char ch = str.charAt(5);

⇒ Kodenet Tech pvt Ltd

- charAt(index):

returns the character that is present at the index given to charAt() as input()

class method?

```
Public static void main(String[] args);
String str = "Kodnest Tech";
Char ch = str.charAt(5);
S.O.P(s1)
S.O.P(ch);
}
```

→ Anonymous object:- objects which doesnot have any variable refering to it are called as Anonymous object (Garbage object)

when there is an object in the heap segment to whom no reference variable is referring to such objects are deleted by garbage collector thread.

As soon as the control leaves/exists the method, the activation record of that particular method will be deleted in the stack segment.

Replace and replaceAll:-

replace() method in Java replaces all occurences of a specified character or substring in a string with a new character or substring. It doesn't support regular expression.

Public String replace(char oldChar, char newChar)

Str.replace('P', 'b');

replaceAll:- method is used to replace all occurences of a substring that matches a regular expression (regex) with a new substring. It is more powerful than replace() because it allows pattern-based replacement.

r = sb.replaceAll("\\d", "#");

→ replace use when you want to replace exact characters or substrings without any pattern matching.

→ replaceAll you need to replace substrings based on pattern or regular expression.

setLength:-

It is the function which is used in StringBuilder and string buffer classes. It is used to set the length of the character sequence in these mutable string classes.

StringBuilder sb = new StringBuilder("Hello World");

sb.setLength(5);

S.o.P(sb); → Hello

StringBuilder sb = new StringBuilder("Hello");

sb.setLength(10);

S.o.P(sb.length()); → 10

S.o.P(sb); → "Hello" followed by 5 null characters (not visible).

→ ['H', 'e', 'l', 'l', 'o', '\0000', '\0000', '\0000', '\0000', '\0000']

ensureCapacity():- StringBuilder sb = new StringBuilder();

sb.ensureCapacity(50); → ensures the internal has enough capacity to accomodate a specified number of characters. Capacity is at least 50.

Format function in string:

This method is used to create a formatted string using format specifiers similar to printf(). It allows you to format numbers, dates, strings and other values.

String a = String.format(format, arguments);

↗ A string that contains format specifiers, which define how the corresponding arguments should be formatted.

↓ The variables are values to be formatted.

%.s → string %.x → Hexadecimal
%d → decimal %.c → character
%.f → float, %.n → newline

→ sting.format("Hello %.s!", name); → Hello, Pupa!

sting.format("I am %d years old.", age); I am 25 years old.

y

stack

Difference between string builder and string buffer:-

→ String buffer is synchronized meaning that it is thread safe. Multiple threads cannot modify a string buffer object at a same time. This makes it safe to use in a multi-threaded enviornment, but it comes with a performance cost due to overhead of synchonization.

→ StringBuilder is not synchonized meaning it is not thread safe. If multiple threads try to access a StringBuilder object simulatenously unpredicatable results may occur. However thre is no synchonization, it is faster

→ Growth factor will work for both string buffer & string builder.

Methods in stringBuilder & string Buffer:-

1. append() → sb.append("Hello");
2. insert() → sb.append(123);
   ↓
   sb.insert(5, "world")
3. delete():- sb.delete(5,10);
4. sb.deleteCharAt(5);
5. sb.replace(5,10, "java");
6. sb.reverse();
7. int cap = sb.capacity()
8. int len = sb.length();
9. char c = sb.charAt(4);
10. substring
11. setLength → sb.setLength(10);
12. ensureCapacity → sb.ensureCapacity(50);
13. toString()
    String result = sb.toString();

We can create a StringBuffer object by first alloting memory to the StringBuffer object using new operator. and later storing the string into it as:-

StringBuffer sb=new StringBuffer(30);

↓ will create with a default capacity of 16 characters.

Here StringBuilder object is created as an empty object with a capacity for storing 30 characters Even if we declare the capacity as 30, it is possible to store more than 30 characters into StringBuffer.

StringBuffer append(x)

↙

x may be float, double, char, string or string Builder. or StringBuffer.

StringBuffer insert(int offset, int end)

↓

x may be int, float, char or string or StringBuilder, it will insert at string offset.

StringBuffer delete(int start, int end)

↓ remove characters from start to end.

StringBuffer reverse()

String methods:

setCharAt(int index, char ch):

sb.setCharAt(0, 'h');

String Builder:

```
sb = "welcome to palle";
sb.charAt(0);
sb.delete(0,8); → it will take two arguments one with start and other with stop.
sb.reverse();
sb.equals(); → Here in string builder it compare address of the 2-strings.
StringBuilder s1 = new StringBuilder("Hello");
StringBuilder s2 = new StringBuilder("Hello");
if (s1.equals(s2)){
    s.o.println("Both are equal");
}
else {
    s.o.println("Both are not equal");
}
```

String s1 = "Hello"
String s2 = "Hello"

if(s1==s2) → compare content

~~sb.tostring();~~
~~sb.~~ ~~sb~~
~~sb.tostring~~ s1.toString();
s2.toString();

Here s1 and s2

string buffer to string

```
s3 = s1.toString();
s4 = s2.toString();
s.o.p(s3.equals(s4));
```

false

→ Here it gives true, because, after converting into string it will store string. so it will give true.

StringBuffer: - It is also predefined class Similar to stringBuilder. String Buffer is synchronized for multiple threads. Not synchronized for String Builder

Thread: - It is a light weight process which has independent path of execution, to make execution faster we will use thread.

to the first came person, given to only for one
until first person complete, thread, rest of threads
second one would wait. should wait.

math.pow() will return value in float.

→ StringBuffer sb = new StringBuffer("Hello");

(or) StringBuffer sb = new StringBuffer();

StringBuffer sb = new StringBuffer(50);

Reverse the strings:-

java class is started
avaj ssalc si detrats

```
class Java {
    public static void main(String[] args) {
        String s1 = "java class is started";
        String s2 = "";
        String[] s3 = s1.split(" ");
        S.O.P(Arrays.toString(s3));
        for(int i=0; i<=s3.length-1; i++) {
            String s4 = s3[i];
            for(int j=s4.length()-1; j>=0; j--) {
                s2 = s2 + s4.charAt(j);
            }
            s2 = s2 + " ";
        }
    }
```

class=5
is=?
started=?

→ Java=>[J, Ja, Jav, Jav

→ started is class Java

Finding element is not there or not:-

Matches:- Is used to check whether a given string matche a specified regular expression (regex). It is apart of th String class and is commonly used for pattern matching.

String str="Hello123";
boolean result=str.matches("Hello\\d{3}");
S.O.P(result); // true. → Hello\\d{3}"

"Hello\\d{3}" checks if the string starts with "hello" followed by exactly three digits. (\d{3}).

→ since "Hello123" fits this pattern, matches() returns true.

String st="1234,5";
boolean result=st.matches("\\d+");
S.O.P(result);

→ The regex \\d+ checks if the string contains one or more digits.

→ String email="test@example.com";
boolean isValid=email.Matches("^[A-Za-z0-9+_-]+
@[A-Za-z0-9,-]+$");
S.O.P(isValid); [+, -, ., or -] followed by an @, and then a domain name.

String date="23/10/2024";
boolean isValidDate=date.matches("\\d{2}/\\d{2}/\\d{4}");
S.O.P(isValidDate); → \\d{2}/\\d{2}/\\d{4} ensures the String is in the format of two digits, a slash two more digits whi matches date format.

```
}  if(i==a[j])
        a1[j]=0; } }
for(int i=0; i<=a1.length-1; i++){
    if (a1[i]!=0) {
        s.o.println(a1[i]);
    }  }
```

Print only the duplicates:-

print sum of duplicate elements
print sum of non-duplicate elements
Count the duplicated elements

ValueOf() Method is present in several classes including String, Integer, boolean, Character and enum. The general purpose of the valueOf() method is to convert or return String or other data types as a specific object type.

String.ValueOf():

Converts different types of data (like integers, booleans, characters, and objects) into a string.

```
Public static String ValueOf(object obj)
  int number = 100;
  String str = String.valueOf(number);
  S.o.P(str);
```

Integer.valueOf():

```
String str = "42";
Integer intvalue = Integer.valueOf(str);
S.o.P(intvalue); // → 42.
int num = 123;
Integer intobj = Integer.valueOf(num);
S.o.P(intobj);
```