# Strings

Recap:

- Arrays.

Today:

- Strings in Java
  - What, Why
  - Java Syntax: Create, Modify
  - String Methods
  - Problems!

Strings:

A sequence of characters.

Characters: a - z, A - Z, 0 - 9, #, !, @, ..

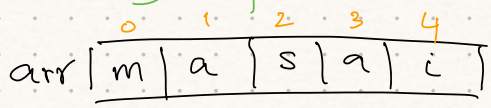
e.g: "Nrupul"

| | |
|---|---|
| Name | "Nrupul", "Masai" |
| Passwords | "masai123#!@" |
| Email Ids | "foo123@gmail.com" |
| Address | "Flat No: 512, Emerald Apts. ... Bangalore" |
| Pincode | 560001 |
| Aadhar<br>PAN NO | "ABC5641X" |

# Strings (Java) (not primitive data type.)

↳ a separate Data type "objects".

Strings ≠ Array of chars.




```
char[] arr = {'m', 'a', 's', 'a', 'i'};
```

## Creating Strings

String value

```
String pwd = "Masai@#12";
int sum = 0;
```

Data Type — variable name

pwd ✗→ "Masai@#12"

↳ "foo!123"

```
pwd = "foo!123";
```

## String Methods.

i) charAt.

positions → 0 1 2 3 4 5 6 7 8 9 10 11 --→ NOT indices

name : Masai School --→ NOT array

```
String name = "Masai School";
S.O.P (name.charAt(0)); // M
S.O.P (name.charAt(8)); // h
S.O.P (name[8]) ✗
```

position

Syntax:

string-name.charAt(i) : returns the character at index i in string-name.

string variable

name.charAt(0) = "P"; ✗

## Length:

S.O.P (name.length()); // 12

E.g: Password greater than 7 characters

```
String passwd = "abc123";
if (passwd.length() > 7)
    S.O.P ("valid Password");
else
    S.O.P ("Invalid Password");
```

Ex: Write a program to print all the characters of a string in a new line.

S → "Masai" (0 1 2 3 4)

O/P: M ✓ a ✓ s a i

| i |
|---|
| 0 |
| 1 |
| 2 |
| 3 |
| 4 |

s.charAt(0) s.charAt(1)

```
for(int i=0; i < s.length(); i++)   // 5
    S.O.P (s.charAt(i));
```

## Concatenation

```
String s = "Masai School";
String snew = "";
for (int i=0; i< s.length(); i++) {
    snew = snew + s.charAt(i);
}
S.O.P (snew);
```

s : "Masai School" (0 1 2 3 4 5 6 7 8 9 10 11)

| i | i < s.length() (12) | snew |
|---|---|---|
| 0 | 0 < 12 ✓ | "" ✗ "M" |
| 1 | 1 < 12 ✓ | "M" "Ma" |
| 2 | 2 < 12 ✓ | "Ma" ✗ "Mas" |

"" + s.charAt(0) = "" + "M"

"M" + s.charAt(i) = "Ma" (i = 1 → a)

"Ma" + s.charAt(2) = "Mas" (2 → s)

empty string ⇒ " "

" " − " "

s : "Masai School" (0 1 2 3 4 5 6 7 8 9 10 11)

```
String s = "Masai School";
String snew = "";
for (int i=0; i< s.length(); i++) {
    snew = s.charAt(i) + snew;
}
S.O.P (snew);
```

| i | i < s.length() (12) | snew |
|---|---|---|
| 0 | 0 < 12 ✓ | "" ✗ "M" |
| 1 | 1 < 12 ✓ | "M" ✗ "aM" |
| 2 | 2 < 12 ✓ | "aM" ✗ "saM" |
| : | | |
| 11 | | |

snew = s.charAt(0) + snew

= "M" + "" = "M"

s.charAt(1) + snew

"a" + "M" = "aM"

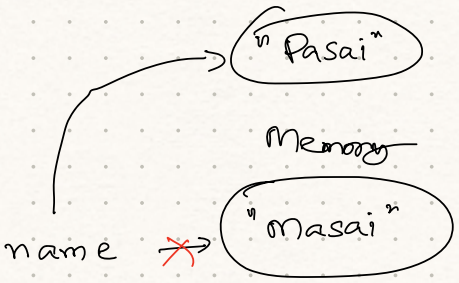
charAt(2) + snew

s + "aM" = "saM"

# Updating Strings

positions 0 1 2 3 4

name : M a s a i




Strings are Immutable

Mutable: something that can be changed.

```
int i = 0;
i++;
```

i : [ 1 ] i is mutable

```
String name = "Masai";
S.O.P (name);
name = "Pasai";
```

A

| 0 | 1 | 2 | 3 | 4 |
|---|---|---|---|---|
| 0 | 0 | 10 | 0 | 0 |

A[2] = 10

"M a s a i"
0 1 2 3 4

To modify (update) we create a new String!

```
String name = "Masai";
int p=2; char c = 'l';   // Change character at position p=2
                            to new character c = 'l';
String newname = "";
for (int i=0; i < name.length(); i++) {
    if (i == p) {
        newname = newname + c;
    } else {
        newname = newname + name.charAt(i);
    }
}
```

```
name:  M a s a i
       0 1 2 3 4
```

```
String name = "masai";
int p=2; char c = 'l';   // Change character at position p=2
                         to new character c = 'l';
String newname = "";
for (int i=0; i < name.length(); i++) {
    if (i == p) {
        newname = newname + c;
    } else {
        newname = newname + name.charAt(i);
    }
}
```

p : 2

c : 'l'

newName : ""

| i | i < name.length() (5) | newname |
|---|---|---|
| 0 | 0 < 5 ✓ | "M" |
| 1 | 1 < 5 ✓ | "Ma" |
| 2 | 2 < 5 ✓ | "Mal" |
| 3 | 3 < 5 | "Mala" |

"" + "M" = "M"

"M" + "a" = "Ma"

"Ma" + 'l' = "Mal"

"Mal" + "a"

## Checking Equality (equals method)

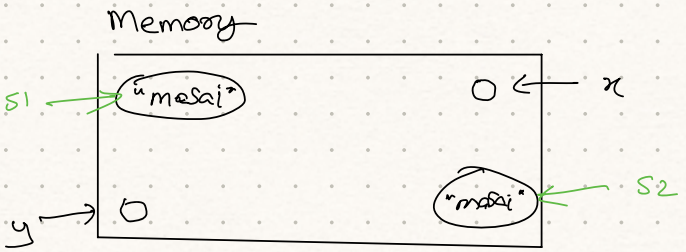
To compare if two strings are equal.

```
String s1 = "masai";
String s2 = "masai";
S.O.P ( s1.equals(s2) )   // s2.equals(s1)
```

// will return true if s1 & s2 have the same sequence of characters.

```
S.O.P (s1 == s2);   ✗
```

false — don't use.




x == y true

x == y — Primitive types — Checks if values are same

s1 == s2 — Non-primitive types — Check if memory location is same

Problems:

1. Write a program to count the number of vowels in a String. e.g s = "Masai"

Vowels: 'a', 'e', 'i', 'o', 'u' o/p : 3

(Assume input is in lower case letters)

0 1 2 3 4

s = M a s a i count = ~~0~~ ~~1~~ ~~2~~ 3.

```
int count = 0;
for (int i = 0 ; i < s.length(); i++) {
    // check if current char is a vowel
    if (s.charAt(i) == 'a' || s.charAt(i) == 'e' ||
        s.charAt(i) == 'i' || s.charAt(i) == 'o' ||
        s.charAt(i) == 'u' ) {
        count++;
    }
}
S.O.P (count);
```

2) Remove all occurrences of a given character

s = "Masai" ; remove all occurrences of c = 'a'

output : "Msi" ← newStr S.O.P(newStr);

3) Check if a string is a Palindrome

s: "madam"

reverse s: : "madam"

if a string s & its reverse are the same

⇒ s is a Palindrome

s : m a s a i ✗

rev s : i a s a m

s : a b b a ✓

rev s : a b b a

---

2) s : m a s a i String s

```
String s = "masai";   // 0 1 2 3 4
char c = 'a';
String snew = "";
for (int i = 0; i < s.length(); i++) {
    if (s.charAt(i) == c) {
        continue;
    } else {
        snew = snew + s.charAt(i);
    }
}
```

```
String s = "person";
s = "school";
```