# String and StringBuilder.

① What → int → Arrays (mutable) [★★Not arrays, Arrays i.e the class.]

char → String. (immutable). ↳ for security reasons!

char ↓ datatypes, String ↓ classes

② ⇒
```
String a = "Tyagi";
String b = "Tyagi";
```
} will point to same thing in memory bcz there is a 'String Pool'.
↳ A data structure in heap to store strings.

③
```
String a = 'b';
a = 'c';
```
} will make a = c but, that is bcz a is created again, not overwritten. 'b' → goes to garbage collection.

④ ⇒ Comparision

→ == } checks if ref var are pointing to same obj.
⇒ So it works for ②

but if obj created as 'new' i.e
```
[ String a = new String("Tyagi")
  String b = new String("Tyagi") ]
```
then == will not work, bcz then new memory allocated.

then .equals() should be used. a.equals(b) → true.

⑤ arr1[0] ——is——→ name1.charAt(0);

PrintStream class → introduced (has 'out' variable/obj refered to funcs ~~as~~ println, print, etc).
↳ many println methods defined using methodoverloading.

⇒ Interesting to know that, everything printed is converted to string before being printed.

I@5acf98 → might be printed bcz java can't understand what you want to print.
↳ (Something example).

⑥ Pretty Printing

⇒ fast input/output (bufferreader, etc) is used in competitive programming.

eg:-
⇒ till 2 digit decimals → %  → placeholder
.2f → f, for float
'2' for 2 digit decimal.

⇒ PI → Math.PI;
→ 3.141592653589793

* Many placeholders for diff types, eg:- %s for strings
%c for char.

---

⑦ ⇒ System.out.println('a'+'b'); → 195 [ASCII value added]
⇒ " " " ("a"+"b"); → ab [concatinated].
⇒ " " " ("a"+1); → a1 [1 converted to string].

---

⑧ ⇒ '+' operator is only defined if one of the value is a string (or) for [primitives] of same type.
└→ int, char, etc.

⑨ ⇒ operator overloading is not supported in Java.

⑩ ⇒ System.out.println((char)(103)); → 'd'.

---

⑪ String performance

a) ⇒
```
for (int i=0; i<26; i++){
    char ch = (char)(a+i);
    series += ch;
}
System ..... (series);
```
out → a,b,cdefgh ---- yz.

26 obj created memory wasted!
O(N²) in space.

b) Stringbuilder → Mutable String type/class. for memory problem.
name.(append, deleteCharAt, etc.---)
many methods in Stringbuilder.

## 12) Write All methods here. for Strings.

a) String name 1 = 'Kunal Kushwaha';
Arrays.toString(name.toCharArray()); $\longrightarrow$ [K, u, n, a, l, ...]

b) toLowerCase() $\longrightarrow$ creates new obj which is lowercase, original one doesn't change.

c) indexOf() $\longrightarrow$ is of 4 types.
- indexOf (ch char)
- indexOf (String str)
- indexOf (String str, int fromIndex)
  ↳ custom/relative distance.

d) isBlank(), isEmpty(), & .contains()

e) lastIndexOf $\longrightarrow$ same indexOf parameters.

f) stripLeading()/stripTrailing()
Trim()
subString (start Index); [abg → ab or bg]
subSequence (start Index, end index) [abg → a, bg, abg, ag etc]

g) (name 1. split( ));
↳ Arrays. toString required.
↳ Regex is added here.

~~h) String[] strArray = new String[] {str};
then System.out.println(strArray[0]) → str can be accessed as an array.~~

~~i) If new not needed,~~ String[] s = "name".split("")
↳ can also be used.

j) instead of h & i, a) can also be used.

classmate
Date
Page

# Additional Strings & Array Notes

k) ch = sc.next().charAt(0); → suppose c.
ch++;
ch → d now.

l) Check out Math. → class.

m) Array to list → for(int i=0; i<n; i++){
arr[i] = List.get(i);
} } M1

for(int i=0; i<n; i++){ is used above but while can also be used.

M2 { → String[] arr = List.toArray(new String[0]);

M3 { → Using stream method.

---

n) 1) System.out.println((int)'A'); //prints 65
System.out.println((char)65); //prints A.

2) sc.useDelimiter("\\D"); → for (integer) input seperated by a non-digit.
↳ 2,3,4,5. → nums seperated by string.

3) int a,b,c;
String line;
String[] lineVector; } [2,3,4,5,6] → only strings.

line = sc.nextLine(); // read input as whole.
lineVector = line.split(","); // seperates line into elem seperated by commas.

a = Integer.ParseInt(linevector[0]);
b = " " " [1]);
c = " " " [2]); } Seperated strings to integers.
∴ Now a,b,c can be used as integers now.

from 2) & 3)

» String[] nums = sc.next().split(":");
int A = parseInt statement from 3) } nums seperated by [:].

** Use Arrays.toString to print all of the above.

(o) → sc.nextLine(); → takes input till end of line i.e till enter pre

sc.next(); → till space pressed.

sc.nextInt(); → takes integer input.

(p) → length(); is used for strings, for arrays length; is used.

→ char[] c1 = s.toCharArray(); } seen regularly.

alphabet [c1[i] - 'a'] ++;

↳ in for loop.

(q) Sliding Window problems.