Strings ⇒ array of characters

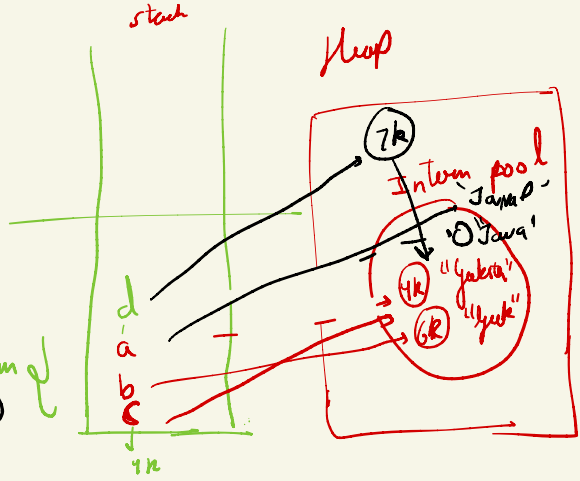
String a = "geekster";

' ' → character literal

Memory

space minimize

String a = "geekster";
String b = "geek";
String c = "geekster";



String d = new String ("geekster")

a = "java"

a = a + 'D';

a = a + "DSA";

why? → because of intern pool

1) Strings are immutable in Java.

↘ you cannot change anything once defined.

String a = "Java"

String b = a;

a = "geek";

print (a), print (b);

geek

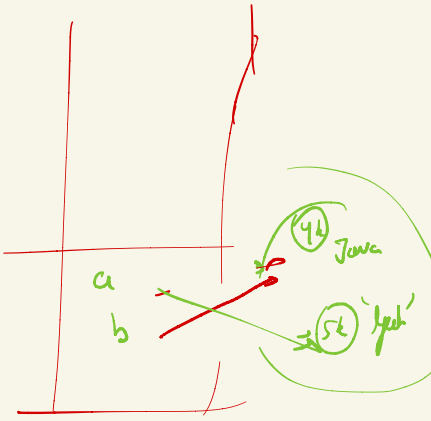
geek

b = a;

print (c), print (b);

Java

Java



```
String a = "Java";
String b = "Java";
String c = new String ("Java");

if (a == b){
    print ("a is equal to b");
}
if (a == c){
    print ("a is equal to c");
}
```

==

addresses check

```
if (a.equals(c)){
    print ("a is equal to c");
}
```

.equals

address check

character by character check

How to take Input

```
String a = scn.nextLine();
```

String → array of characters

	0	1	2	3	4	5	6	7
string a =	"	g	e	e	k	s	t	e	re "

```
char ch = a.charAt(3); // ch == 'k'
```

length of string:

```
String str = scn.nextLine();
int len = str.length();
```

Ques Take a input string. Count the number of vowels in that string.

eg ⇒ geekipu

output → 4

[e, e, i, u]

Ques Print the string in reverse order.

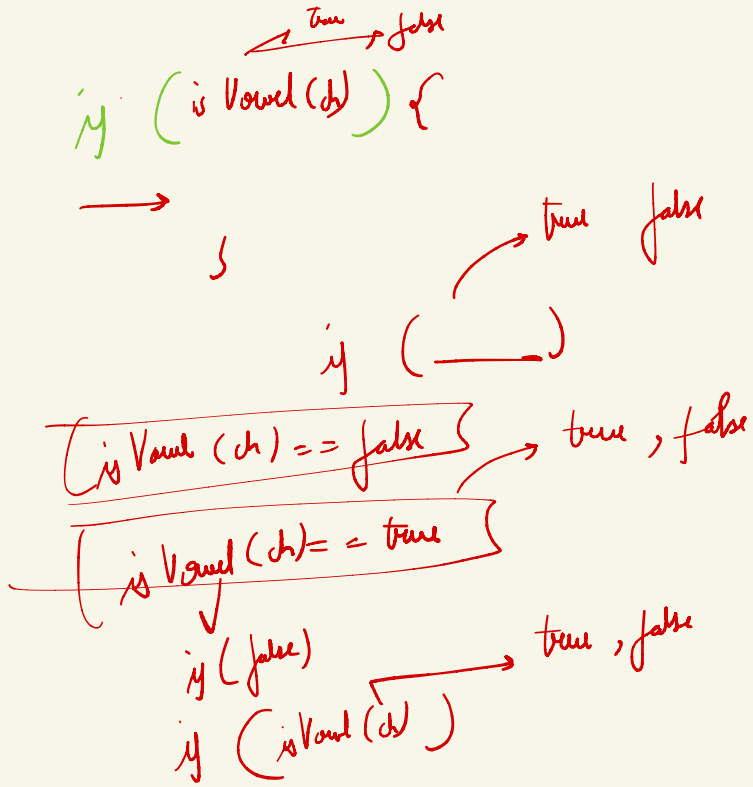
eg → "geekipu"

output → "upikeeg"

```
String a = "nano"
for (int i=0; i<10; i++) {
    a = a + i;
}
```

(1) → O(n) → n^2 abcd

a = nano0
a = nano09
a = nano012
a = nano0123
a = nano01234



Ques count spaces.

Ques alternate elements.

next(), nextLine()

Ques

Ques length | 3

Ques Ideal string

Ques Take input string. Count all the spaces in that string.

eg. "geekster is organisateur"

output = 2

```java
public static int countSpaces(String str){
    int n=str.length();
    int count=0;

    for(int i=0; i<n; i++){
        char ch=str.charAt(i);

        if(ch==' '){
            count++;
        }
    }

    return count;
}
```

Ques Print alternate elements of string

0 1 2 3 4 5 6 7 8

eg → "a b c d e f g h i"

output → a
c
e
g
i

```java
public static void printAlternateElements(String str){
    int n=str.length();

    for(int i=0; i<n; i=i+2){
        char ch=str.charAt(i);
        System.out.println(ch);
    }
}
```

How to take input in String.

String s1 = scn.nextLine(); → will input the whole line.

String s2 = scn.next(); → will take input before 'space'.

"geekster is organisateur".

s1 = "geekster is organisation".

s2 = "geekster".

Ques Input a string, find if length of this string is divisible by 3.

Ques Take one input string. check if it is ideal. If not, change it to ideal.

Ideal String ⇒ first letter uppercase, rest lowercase.

string ⇒ Geekster

string ⇒ g E E k

ch = E

ch = e

```java
public static String convertToIdeal(String str){
    String ans="";

    // checking if first charcater is lowercase, we will change it to uppercase
    char ch=str.charAt(index: 0);

    if('a'<=ch && ch<='z'){
        ch=Character.toUpperCase(ch);
    }

    ans=ans+ch;

    for(int i=1; i<str.length(); i++){
        ch=str.charAt(i);

        if('A'<=ch && ch<='Z'){
            ch=Character.toLowerCase(ch);
        }

        ans=ans+ch;
    }

    return ans;
}
```

if ('A' <= ch && ch <= 'Z')

dec	hex	oct	char	dec	hex	oct	char	dec	hex	oct	char	dec	hex	oct	char
0	0	000	NULL	32	20	040	space	64	40	100	@	96	60	140	`
1	1	001	SOH	33	21	041	!	65	41	101	A	97	61	141	a
2	2	002	STX	34	22	042	"	66	42	102	B	98	62	142	b
3	3	003	ETX	35	23	043	#	67	43	103	C	99	63	143	c
4	4	004	EOT	36	24	044	$	68	44	104	D	100	64	144	d
5	5	005	ENQ	37	25	045	%	69	45	105	E	101	65	145	e
6	6	006	ACK	38	26	046	&	70	46	106	F	102	66	146	f
7	7	007	BEL	39	27	047	'	71	47	107	G	103	67	147	g
8	8	010	BS	40	28	050	(	72	48	110	H	104	68	150	h
9	9	011	TAB	41	29	051	)	73	49	111	I	105	69	151	i
10	a	012	LF	42	2a	052	*	74	4a	112	J	106	6a	152	j
11	b	013	VT	43	2b	053	+	75	4b	113	K	107	6b	153	k
12	c	014	FF	44	2c	054	,	76	4c	114	L	108	6c	154	l
13	d	015	CR	45	2d	055	-	77	4d	115	M	109	6d	155	m
14	e	016	SO	46	2e	056	.	78	4e	116	N	110	6e	156	n
15	f	017	SI	47	2f	057	/	79	4f	117	O	111	6f	157	o
16	10	020	DLE	48	30	060	0	80	50	120	P	112	70	160	p
17	11	021	DC1	49	31	061	1	81	51	121	Q	113	71	161	q
18	12	022	DC2	50	32	062	2	82	52	122	R	114	72	162	r
19	13	023	DC3	51	33	063	3	83	53	123	S	115	73	163	s
20	14	024	DC4	52	34	064	4	84	54	124	T	116	74	164	t
21	15	025	NAK	53	35	065	5	85	55	125	U	117	75	165	u
22	16	026	SYN	54	36	066	6	86	56	126	V	118	76	166	v
23	17	027	ETB	55	37	067	7	87	57	127	W	119	77	167	w
24	18	030	CAN	56	38	070	8	88	58	130	X	120	78	170	x
25	19	031	EM	57	39	071	9	89	59	131	Y	121	79	171	y
26	1a	032	SUB	58	3a	072	:	90	5a	132	Z	122	7a	172	z
27	1b	033	ESC	59	3b	073	;	91	5b	133	[	123	7b	173	{
28	1c	034	FS	60	3c	074	<	92	5c	134	\	124	7c	174	|
29	1d	035	GS	61	3d	075	=	93	5d	135	]	125	7d	175	}
30	1e	036	RS	62	3e	076	>	94	5e	136	^	126	7e	176	~
31	1f	037	US	63	3f	077	?	95	5f	137	_	127	7f	177	DEL

www.alpharithms.com

Printable ASCII Table (Click to Enlarge)

Ques Take whole line as input consisting of diff. strings separated by space. If these strings are not ideal, change it to ideal and print.

eg → Geekster geek gEEk

⇒ Geekster Geek Geek

str ⇒ "Geekster geek gEEk" gEEk" i=0 1 2 3 i<4

String [] arr = str.split(" ");

arr(0) = "Geekster" arr(1) = geek

arr(2) = gEEk

arr(3) = gEEk.

```
Run | Debug
public static void main(String[] args) {
    Scanner scn=new Scanner(System.in);

    String str=scn.nextLine();

    // String ans=convertToIdeal(str);

    // System.out.println(ans);

    String[] arr=str.split(regex: " ");

    for(int i=0; i<arr.length; i++){
        String s=arr[i];

        String idealString=convertToIdeal(s);

        System.out.print(idealString+" ");
    }
}
```

arr ⇒ | 0: Geekste | 1: geek | 2: gEEk | 3: gEEk |

Geekster Geek Geek Geek

Ques Input one string. Sort the string (using any function)

eg → str = bcad

output ⇒ abcd

char[] arr = str.toCharArray();

arr ⇒ | 0: 'b' | 1: 'c' | 2: 'a' | 3: 'd' |

Arrays.sort(arr);

arr ⇒ | 0: 'a' | 1: 'b' | 2: 'c' | 3: 'd' |

```
String s = new String(owl);
```