Strings

↔ property

Array of characters

- ↳ lowercase 'a'-'z'
- ↳ uppercase 'A'-'Z'
- ↳ digits '0'-'9'
- ↳ special symbols
- ↳ null character '\0'

→ full form can be asked

ASCII

(American Standard Code for Information Interchange)

Java uses Unicode (codes for many languages) [this is why char in Java is of 2 bytes]

```
Scanner scn = new Scanner(System.in);
String str = "hello";
System.out.println(str);
System.out.println(str.length());

for(int i=0;i<str.length();i++) {
    System.out.println(str.charAt(i));
}
```

ASCII TABLE

Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char
0	0	[NULL]	32	20	[SPACE]	64	40	@	96	60	`
1	1	[START OF HEADING]	33	21	!	65	41	A	97	61	a
2	2	[START OF TEXT]	34	22	"	66	42	B	98	62	b
3	3	[END OF TEXT]	35	23	#	67	43	C	99	63	c
4	4	[END OF TRANSMISSION]	36	24	$	68	44	D	100	64	d
5	5	[ENQUIRY]	37	25	%	69	45	E	101	65	e
6	6	[ACKNOWLEDGE]	38	26	&	70	46	F	102	66	f
7	7	[BELL]	39	27	'	71	47	G	103	67	g
8	8	[BACKSPACE]	40	28	(	72	48	H	104	68	h
9	9	[HORIZONTAL TAB]	41	29	)	73	49	I	105	69	i
10	A	[LINE FEED]	42	2A	*	74	4A	J	106	6A	j
11	B	[VERTICAL TAB]	43	2B	+	75	4B	K	107	6B	k
12	C	[FORM FEED]	44	2C	,	76	4C	L	108	6C	l
13	D	[CARRIAGE RETURN]	45	2D	-	77	4D	M	109	6D	m
14	E	[SHIFT OUT]	46	2E	.	78	4E	N	110	6E	n
15	F	[SHIFT IN]	47	2F	/	79	4F	O	111	6F	o
16	10	[DATA LINK ESCAPE]	48	30	0	80	50	P	112	70	p
17	11	[DEVICE CONTROL 1]	49	31	1	81	51	Q	113	71	q
18	12	[DEVICE CONTROL 2]	50	32	2	82	52	R	114	72	r
19	13	[DEVICE CONTROL 3]	51	33	3	83	53	S	115	73	s
20	14	[DEVICE CONTROL 4]	52	34	4	84	54	T	116	74	t
21	15	[NEGATIVE ACKNOWLEDGE]	53	35	5	85	55	U	117	75	u
22	16	[SYNCHRONOUS IDLE]	54	36	6	86	56	V	118	76	v
23	17	[ENG OF TRANS. BLOCK]	55	37	7	87	57	W	119	77	w
24	18	[CANCEL]	56	38	8	88	58	X	120	78	x
25	19	[END OF MEDIUM]	57	39	9	89	59	Y	121	79	y
26	1A	[SUBSTITUTE]	58	3A	:	90	5A	Z	122	7A	z
27	1B	[ESCAPE]	59	3B	;	91	5B	[	123	7B	{
28	1C	[FILE SEPARATOR]	60	3C	<	92	5C	\	124	7C	|
29	1D	[GROUP SEPARATOR]	61	3D	=	93	5D	]	125	7D	}
30	1E	[RECORD SEPARATOR]	62	3E	>	94	5E	^	126	7E	~
31	1F	[UNIT SEPARATOR]	63	3F	?	95	5F	_	127	7F	[DEL]

65 - 90 (A-Z)

97 - 122 (a-z)

48 - 57 (0-9)

{'h', 'e', 'l', 'l', 'o'}

⇕

"hello"

(No need of null ('\0') termination of character array in Java)

Toggle Case

'a' − 'A' = 32
'A' − 'a' = −32
} for every upper & lower case character.

→ if char is lower case subtract 32 from it
→ if char is uppercase add 32 to it.

if lowercase → [97 - 122]
else → [65 - 90]

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

    for(int i=0;i<str.length();i++) {
        if(str.charAt(i) >= 65 && str.charAt(i) <= 90) {
            char ch = (char)(str.charAt(i) + 32);
            ans += ch;
        } else {
            char ch = (char)(str.charAt(i) - 32);
            ans += ch;
        }
    }
```

Type Conversion

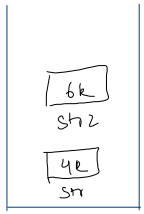
char → int → long
short
byte

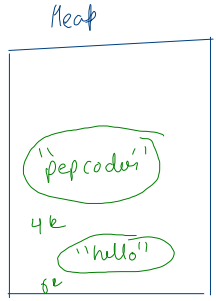
Big type → Small type
↓
lossy conversion warning
↓
avoid it using type casting

long x = (int)(x);

Memory Mapping for Strings

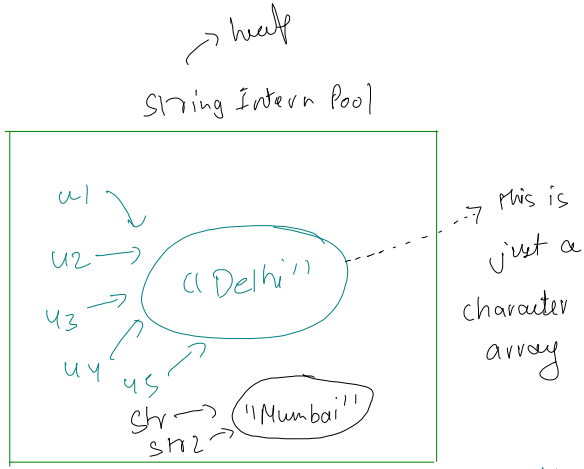
String str = "pepcoder"
↓
String literal





String str2 = new String("hello");

String Interning

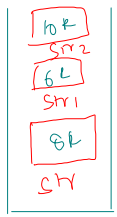


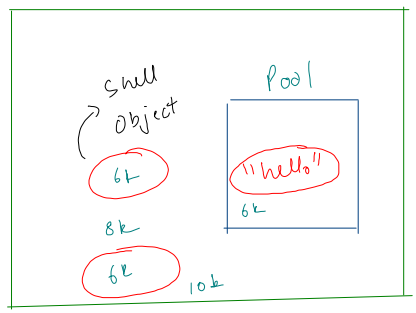
String u1 = "Delhi"
String u2 = "Delhi"
⋮
String u5 = "Delhi"

String str = "Mumbai"
String str2 = "Mumbai"

When we do

String str = new String("hello");
str1 = "hello"
str2 = new String("hello");





Implicationss of Interning

↳ difference b/w == and equals()

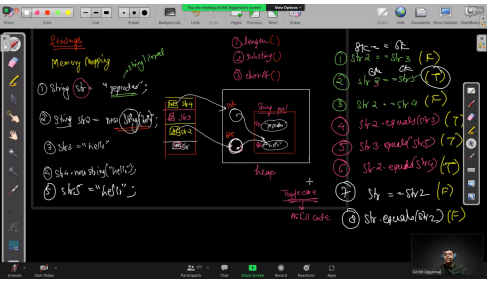
↳ Immutability

①

```
String str1 = "pepcoder";
String str2 = new String("hello");
String str3 = "hello";
String str4 = new String("hello");
String str5 = "hello";

System.out.println(str2 == str5);
System.out.println(str3 == str5);
System.out.println(str2 == str4);
System.out.println(str2.equals(str3));
System.out.println(str2.equals(str3));
System.out.println(str2.equals(str3));
```

"==" compares only addresses

.equals() → first compares address
↳ if address is not equals)
it compares the content
↳ if both are not equal
then false



8k 8k
str2 == str3
(F)

8k 8k
str3 == str5
(T)

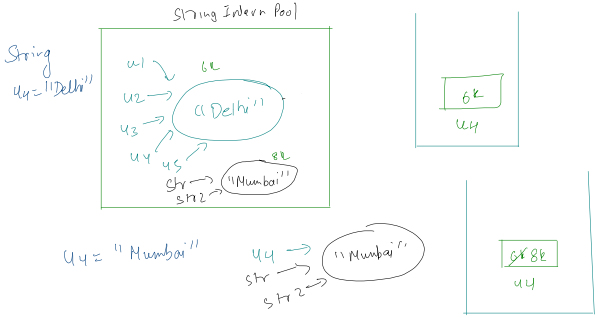
& so on

Do not use '==' to compare strings. Use .equals() to compare strings always

↳ Immutability

We cannot make changes in the objects using references.

** References are mutable but instances are not

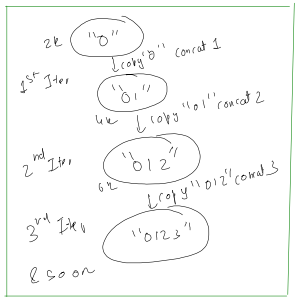


** Thus setting/updating a character in String in Java is NOT POSSIBLE / ALLOWED

String operations are costly in Java

```
String s = "0";
for(int i=1; i<=1000; i++)
{
    s = s+i;
}
```

This is not O(n) loop.
This is $O(n^2)$



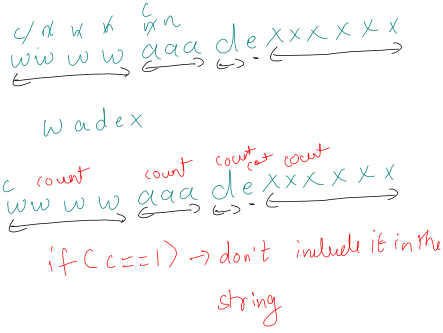
String Compression

Sample Input

wwwwaaadexxxxxx

Sample Output

wadex
w4a3dex6




```java
public static String compression1(String str){
    // write your code here

    String ans = "";

    int curr = 0;
    while(curr < str.length()) {

        int prev = curr-1;
        if(prev == -1) {
                ans += str.charAt(curr);
                curr++;
                prev++;
        }

        if(str.charAt(curr) == str.charAt(prev)) {
            curr++;
            prev++;
        } else {
            ans += str.charAt(curr);
            curr++;
            prev++;
        }


    }

    return ans;
}
```

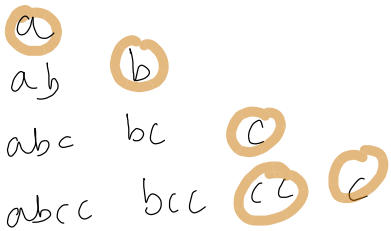
```java
public static String compression2(String str){
    // write your code here

    String ans = "";

    int curr = 0;

    while(curr < str.length()) {

        int count = 0;
        int next = curr;

        ans += str.charAt(curr);

        for(next = curr;next<str.length() && str.charAt(next) == str.charAt(curr);next++) {
            count++;
        }

        if(count != 1) {
            ans += count;
        }

        curr = next;

    }

    return ans;
}
```

Substring(i,j) Method

Ques) Print all palindromic substrings

"bb abcc"

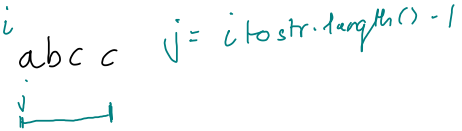
→ Substrings are:



(Palindromic are marked)

Generate substrings and

if (isPalindrome())
    print




```java
public static boolean isPalindrome(String str) {

    int i = 0;
    int j = str.length() - 1;

    while(i <= j) {
        if(str.charAt(i) != str.charAt(j)) return false;
        i++;
        j--;
    }

    return true;
}

 public static void solution(String str){
    //write your code here

     for(int i=0;i<str.length();i++) {
         for(int j=i;j<str.length();j++) {
             String subStr = str.substring(i,j+1);
             if(isPalindrome(subStr)) {
                 System.out.println(subStr);
             }
         }
     }

 }
```

substring func
includes starting
idx & excludes ending idx.

StringBuilder ⇒ mutable (both reference & instance)

(Just like C++ String)

* There is no interning.

```java
StringBuilder sb = new StringBuilder("hello");
System.out.println(sb);
System.out.println(sb.length());

for(int i=0;i<sb.length();i++) {
    System.out.print(sb.charAt(i) + " ");
}
System.out.println();

//updation
sb.setCharAt(3,'d');
System.out.println(sb);

//deletion
sb.deleteCharAt(1);
System.out.println(sb);

//insert
sb.insert(2,'A');
System.out.println(sb);

//add character at the end
sb.append('S');
System.out.println(sb);
```

string & char both can be appended

deletion & insertion are O(n) because it requires shifting

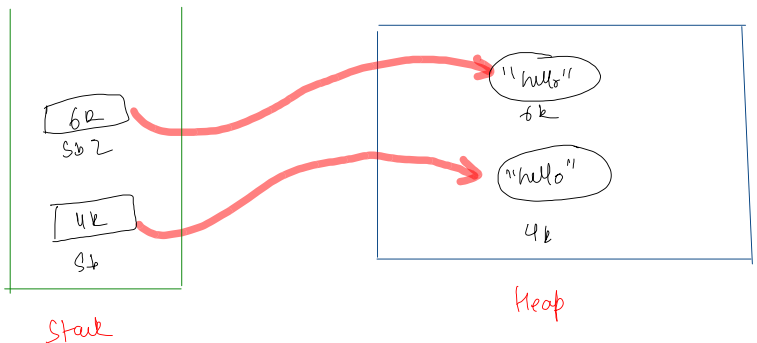
→ Input a String always then convert to SB.

```java
String str = scn.nextLine();
StringBuilder sb = new StringBuilder(str);
```

Memory Mapping

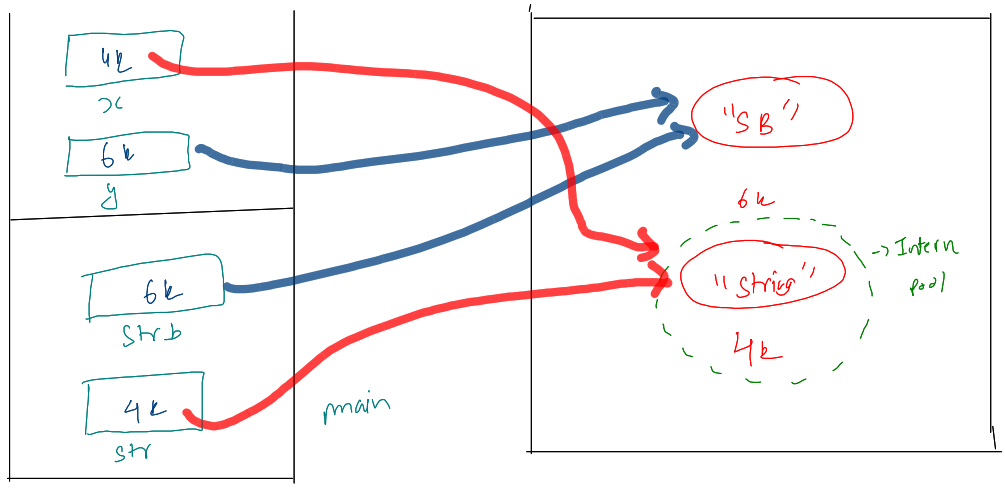
StringBuilder sb = new StringBuilder("hello");

StringBuilder sb2 = new StringBuilder("hello");



Passing String & StringBuilder to function

```java
public static void fun(String x, StringBuilder y){
    x = x + " is changed";
    y.append(" is changed");
}

public static void main(String[] args) {
    String str = "string";
    StringBuilder strb = new StringBuilder("stringbuilder");
    fun(str, strb);
    System.out.println(str);
    System.out.println(strb);
}
```



When we try to append in x, a new String will be formed where changes will occur which will not reflect in main()

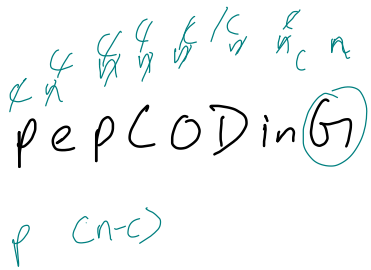
Ques String with diff of every 2 consecutive characters

Sample Input

pepCODinG

Sample Output

p-11e11p-45C12O-11D37i5n-39G



Optimised approach uses a StringBuilder instead of a String

```java
public static String solution(String str){
    // write your code here

    StringBuilder ans = new StringBuilder("");
    int curr = 0;
    int next = curr + 1;

    while(next < str.length()) {

        char currChar = str.charAt(curr);
        char nextChar = str.charAt(next);

        ans.append(currChar);
        int diff = (nextChar-currChar);
        ans.append(diff);

        next++;
        curr++;
    }
    ans.append(str.charAt(curr));
    return ans.toString();
}
```

```java
public static String solution(String str){
    // write your code here

    String ans = "";
    int curr = 0;
    int next = curr + 1;

    while(next < str.length()) {

        char currChar = str.charAt(curr);
        char nextChar = str.charAt(next);

        ans = ans + currChar;
        int diff = (nextChar-currChar);
        ans = ans + diff;

        next++;
        curr++;
    }
    ans = ans + str.charAt(curr);
    return ans;
}
```

ArrayList

↳ Arrays are static

↳ Size of arrays cannot be grown or shrink

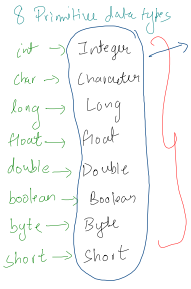
So, ArrayList is dynamic

↳ also, ArrayList is Generic Class (Generic Programming)

Remove Primy

0	1	2	3	4	5	6	7
2	3	4	5	6	7	8	9

8 Primitive data types



default value is NULL

Wrapper Classes are also immutable ***

So, for all collection framework DS, use Wrapper Classes within the < > brackets.

** Refer to auto boxing & auto unboxing article to know the reason behind this

ArrayList can also be directly printed using

```
public static void main(String[] args) {
    Scanner scn = new Scanner(System.in);
    int n = scn.nextInt();

    ArrayList<Integer> arr = new ArrayList<>();

    // Input
    for(int i=0; i<n; i++){
        Integer x = scn.nextInt();
        arr.add(x);
    }

    // Update
    arr.set(3, 100);

    // Removal / Deletion
    arr.remove(2);

    // Traversal / Get / Output
    for(int i=0; i<arr.size(); i++){
        System.out.print(arr.get(i) + " ");
    }
```

Syso(arr);

↓

[e1, e2 e3 $ e4]

In fact, if we do Syso(arr.toString()) same arraylist as above will be printed ie toString() is invoked automatically.

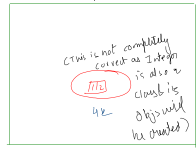
For Each Loop

```
for(Integer val: arr){
    System.out.print(val + " ");
}
```

Print Each Value (Can be applied on various DS like Strings, Arrays & so on)

Memory Mapping

ArrayList<Integer> arr = new ArrayList<>();



Ques) Remove All Primes

Example

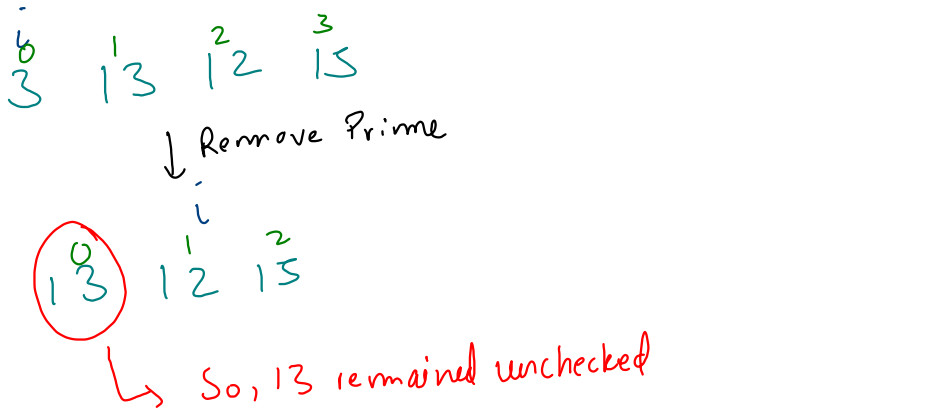
Sample Input

4
3 12 13 15

Sample Output

[12, 15]

```java
public static void solution(ArrayList<Integer> al){
    // write your code here
    int i = al.size() - 1;

    while(i >= 0) {
        if(isPrime(al.get(i))) {
            al.remove(i);
        }
        i--;
    }
}

public static boolean isPrime(int n) {

    for(int i=2;i*i<=n;i++) {
        if(n%i == 0) {
            return false;
        }
    }

    return true;
}
```



∴ Traverse in reverse & it won't be a problem or travel in same dirbut do not change idx pos if a prime no is found.