Day 40
DSA

13 Oct, 2021

String continued.....

Page No.
Date

DSA Strings Agenda for Today

1. String Interning & Immutability → done
2. Introduction to StringBuilder & its purpose → done
3. Toggle Case → done
4. String with diffrences of chars → hw
5. String permutations → Test/hw
6. Introduction to Arraylist → Pending
7. Remove Brunes → Pending

1. String Interning & Immutability → explained seriously in Day 36 Notes

String Interning

What?
Searching for an identical string literal in the Intern Pool area, if there is same literal present, then share same memory

Why
It's done to save memory space

Implications

Implication1
We should never use "==" for comparision of strings, use (equals) method because it will check the correct address of string

Implication2
Strings became Immutable

- **What?** Reference are mutable, Instances are not.
- **WHY?** Because of Interning many string points to same Instance, & changing 1 Instance would change many strings (to avoid that, we are not allowed to any Instance by JAVA
- **Implication1** Some function are missing like we are not able to change the Instance
- **Implication2** Performance issue, अगर हम किसी String की data change करते हैं, तो new String बनती है, in result Time & Space Complexity increase
- **Solution** StringBuilder solves all these problems of String

Reference होता है जिसमे address पड़ा है।
Instance होता है

Page No.
Date

String Immutability का Implication

↳ Either modification function for Instance (data of String) is not available (like setCharAt, remove) ⟶ Not available

Or If there is Modification function available for Instance of String, then its performance is Poor because it makes a complete new copy of entire String & then makes change. It increases the Space & Time and Space Complexity increases as a result.
Time Complexity become $O(n^2)$ instead we expect it to be $O(n)$

STRING में Modification करने के लिए एक function होता है → **Replace**

```
String s = "hello";
s.replace('h','b');          ✗   → replace function
System.out.println(s);
output → hello
```

इसमे hello ही print होगा क्योंकि s variable में address तो पहले वाली String का ही saved है।

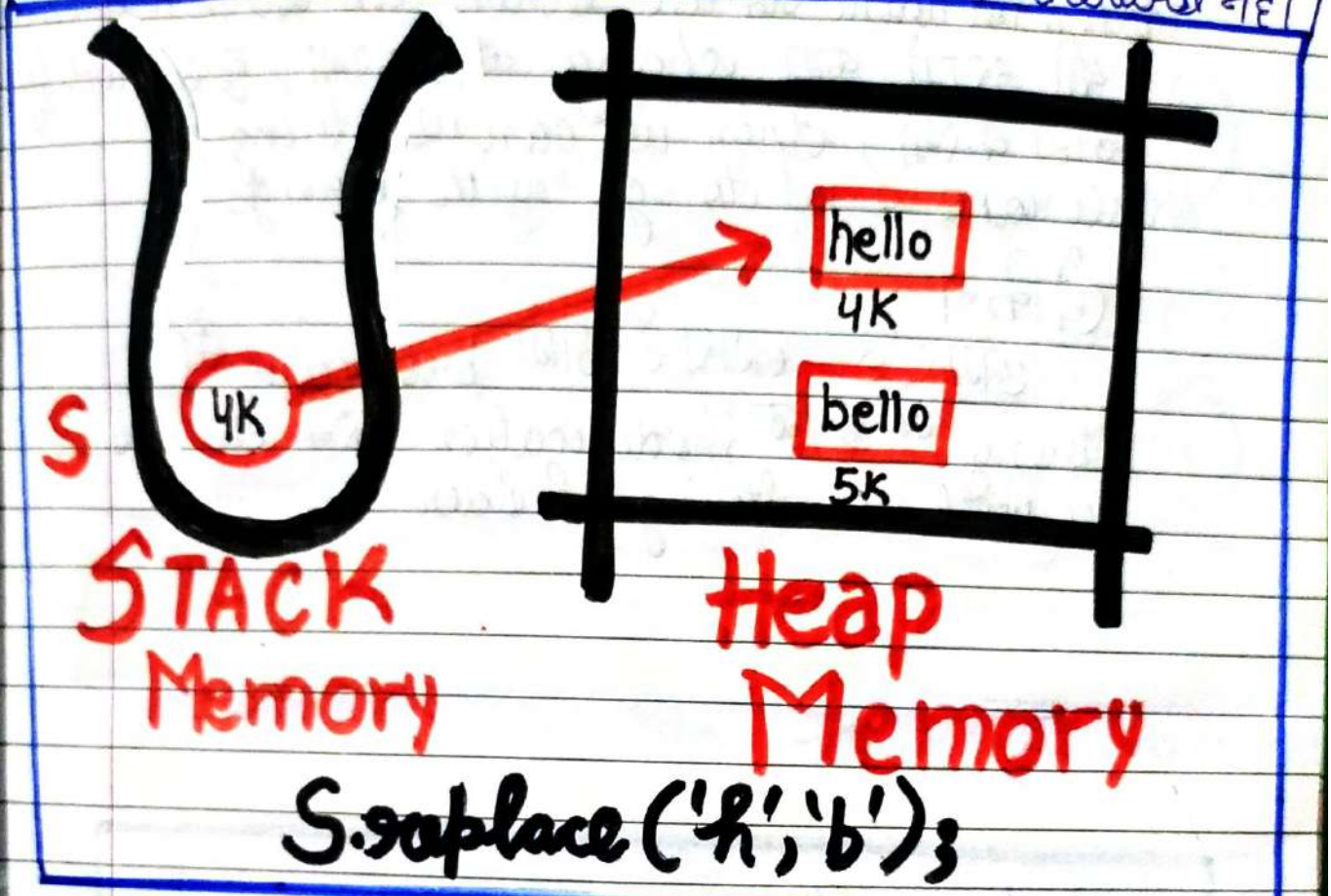
New String तो की replace करने पर उसका address reference

```
s = s.replace('h','b');      ✓
System.out.println(s);
output → bello
```

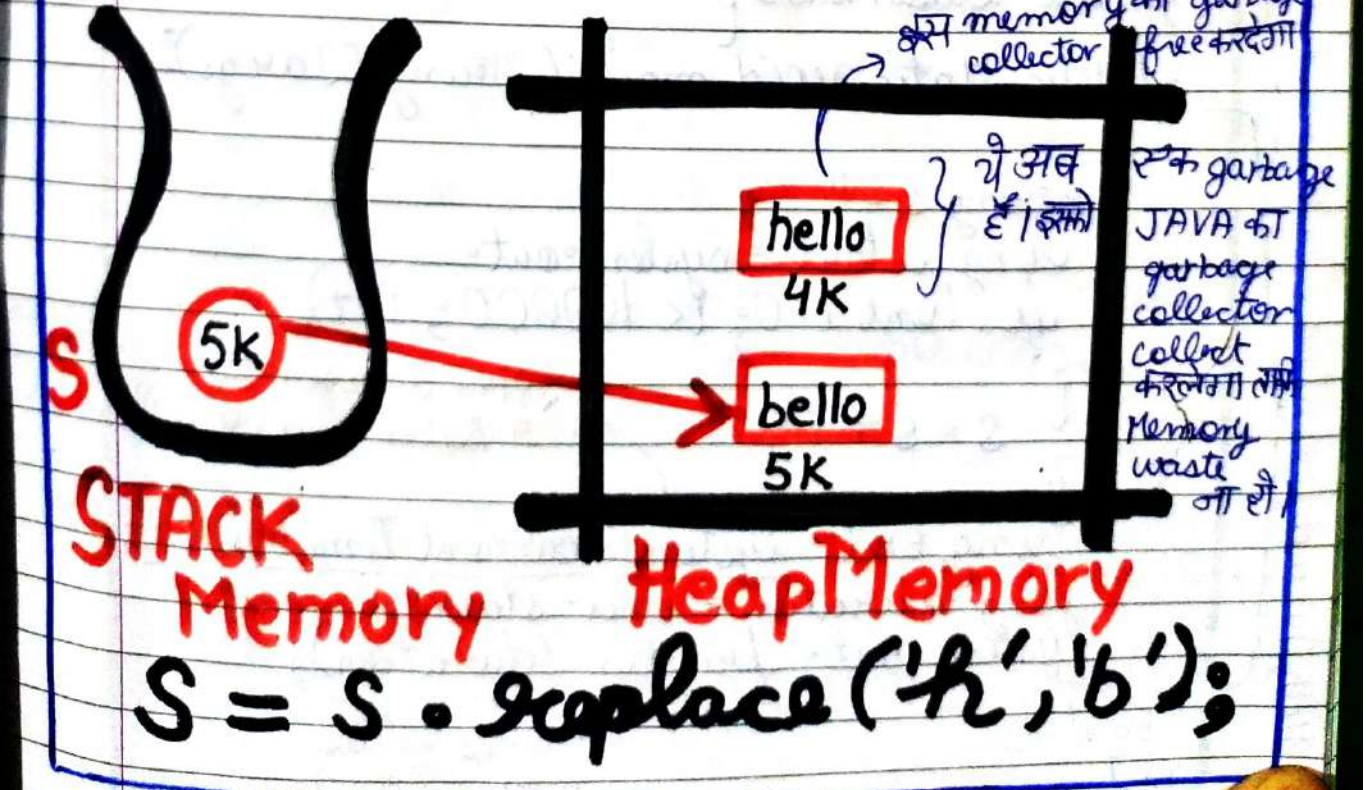
Same करो तो s variable में तो ही changed String print होगी।

Replace करके receive करना is most important part here, तभी हमें replaced string output में मिलेगी otherwise नहीं

S = S.replace('h', 'b');



S.replace('h','b');



S = S.replace('h','b');

Date

StringBuilder is the solution to String.
String is not meant for modification.
When we want to use String in such a form.
की उसमे कुछ change नही करना, कुछ modify नही करना, then we can use String.
We are good to go with Strings

लेकिन

अगर हमे लगता है आगे program में String मे कुछ modification करनी होगी, then we will use String Builder.

Proof → Performance of String is Poor } Both Space & Time Complexity

Program to modify/change value/data of String 1L times

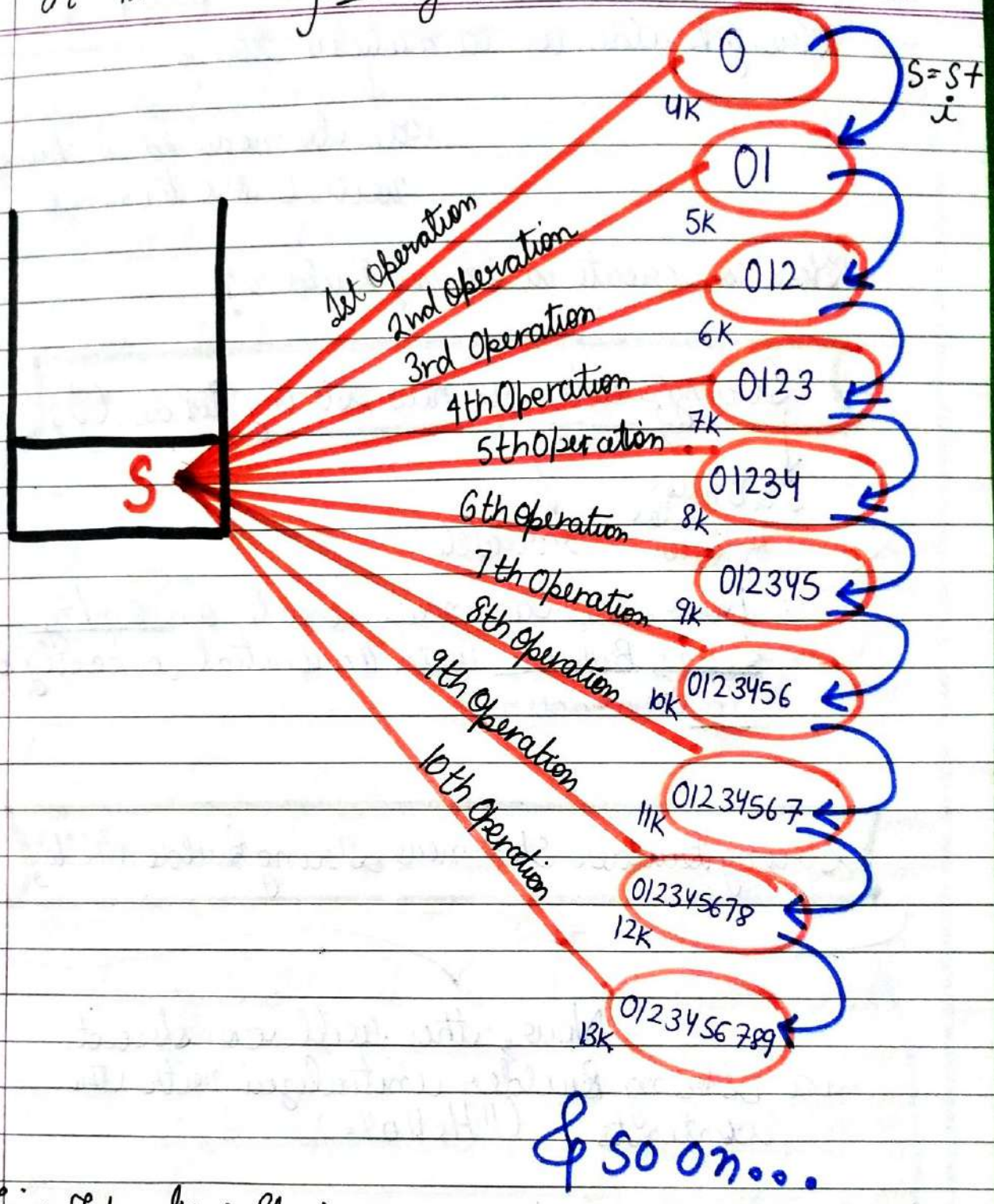
```
import java.util.*;
public class Main {
    public static void main(String[] args)
    {
        String s = "";
        long start = System
        for(int i=0; i< 100000; i++)
        {
            s = s + i;
        }
        long end = System.currentTimeMillis();
        long duration = end-start;
        System.out.println(duration);
    }
}
```

for i=1Lakh, This loop gives time limit Exceeded TLE error

इसको Javascript में Date.now() बोलते हैं।

gives millisecond since 1 Jan 1970

Each Modification on String is taking n^2 Time Complexity...



& so on...

So Time Taken by 1 String =

→ Time of 1st Op + Time of 2nd Operation + ——— Time of nth operation

→ 1+2+3+4+5+ ——— n

→ $\frac{n(n+1)}{2}$ = Time Complexity = n^2 } ∴ 1 Modification/1 Loop on String is Very Harmful

STRING BUILDER

→ String Builder is a mutable String
↓
can be modified & changed with 0 shortcomings

How to create a String Builder ?

```
StringBuilder sb = new StringBuilder();
```

S & B must be capital letter

variable

→ Now, this line will create an empty String Builder with an initial capacity of 16 characters.

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

Now, this will construct a String Builder initialized with the contents "Hello".

Program to modify/change value of String Builder 1L Times

```
→ import java.util.*;
→ public class Main {

public static void main (String[] args)
{
  long start = System.currentTimeMillis();
```

long use किया है क्योंकि digit बड़ी होगी

currentTimeMillis() → gives milliseconds passed since 1 Jan 1970

String (commented):

```
// String s=" ";
// for(int i=0; i<10000; i++)
// {
//   s=s+i;
// }
// long end =System.TimeMillis();
// System.out.println(duration);
```

∴ $(10000)^2$ operations $= n^2$ $= 10^8$ operations

long duration = end-start

for string, if we run the loop only for 10K times the string will take lots & lots of Time.

Output = 145 Milliseconds

StringBuilder:

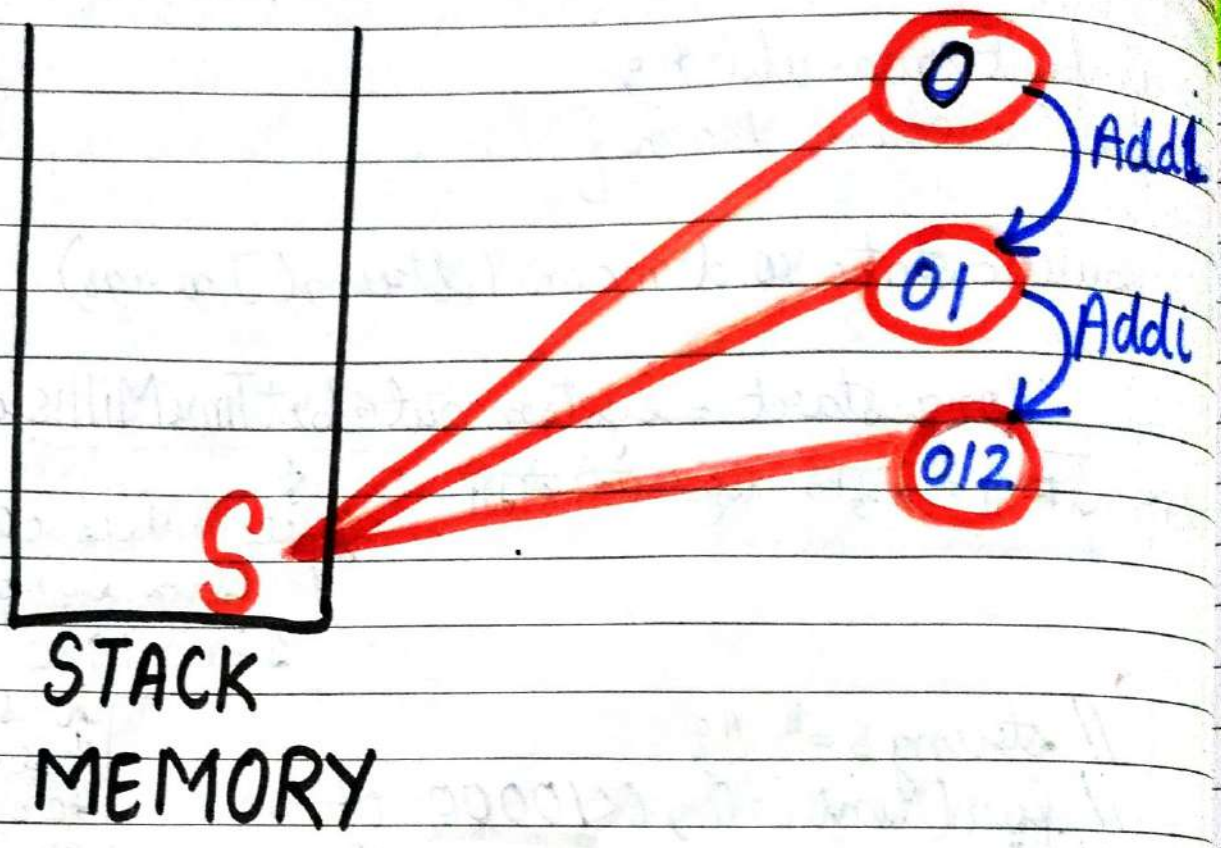
```
  StringBuilder sb = new StringBuilder();
  for ( int i=0; i<10000; i++)
  {
    sb.append(i);
  }
  long end=System.currentTimeMillis();
  long duration= end-start;
  System.out.println(duration);
}
}
```

Use append func here to add i

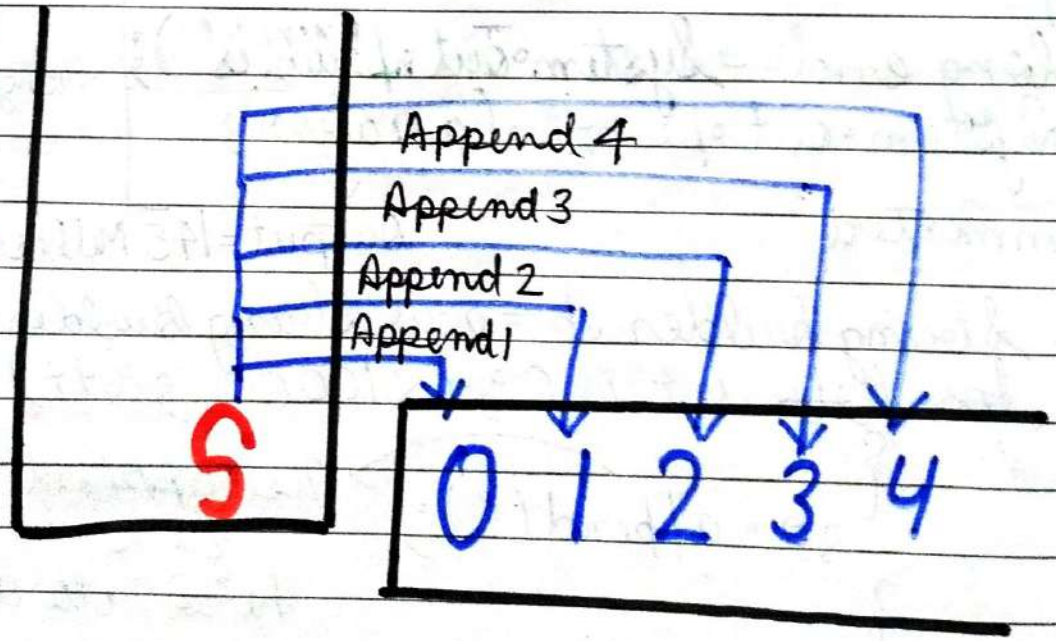
Output = 10 Milliseconds

HUGE TIME DIFFERENCE

USING STRING



USING STRINGBUILDER



Methods of STRINGBUILDER

StringBuilder sb = new StringBuilder("Hello");

1. sb.length() → ये Parathesis आएगा, string, stringbuilder की length निकालते () आते हैं
   would return the length (character count of StringBuilder)

   array की length निकालते टाइम () नहीं लगाते।

2. sb.charAt(int index) (A capital)
   returns the character at specified Index.

3. sb.setCharAt(int index, char ch); (C capital)

   ये Camel Case follow हुआ है

   Sets (over writes) the character at specified index as content of ch.

   → ये function String में नहीं होता था (setchar करने का) लेकिन String Builder provide करता है)

4. sb.append(Object obj) **append is O(1)**
   Appends the parameter passed as string.
   The object can range from datatype
   - → int
   - → float
   - → string
   - → character

   → Sequence of character & anything

5. sb.deleteCharAt(int idx) (C capital)
   camel case

   Deletes a character at the specified index.

Date

6. sb.replace(int startidx, int end_idx, String str)

Replaces the substring contained between start and end index with the specified string str.

7. sb.delete(int start_idx, int end_idx)

Deletes the substring contained between the start and end index.

8. sb.toString()

returns a String representing the contents of sb(StringBuilderObject).

StringBuilder :- Methods <code>

```
import java.util.*;
public class Main{
    public static void main (String [] args){
    StringBuilder sb = new StringBuilder ("Hello");
    // Printing sb StringBuilder
    System.out.println("String :  " + sb);
```

→ Output → String : Hello

1. // Printing the length of sb StringBuilder

length

```
System.out.println ("Length of String :  "
                    + sb.length());
```

→ Output → Length of String : 5

2. // Printing the first character of sb StringBuilder

charAt

```
System.out.println ("First Character :  "
                    + sb.charAt(0));
```

→ Output → First Character : H

3. // Setting the first character as 'h'

setCharAt

```
setCharAt (0,'h');
System.out.println ("String : " + sb);
```

→ Output → String : hello

↳ small letter

↳ first character को 'h' set करादिया, पहले 'H'

4. // Appending the String 'reader' to sb StringBuilder.

append

```
sb.append("  reader");
System.out.println("String : " + sb);
```

→ Output → String : hello reader

5. // Delete the character at index 5

deleteCharAt

```
sb.deleteCharAt(5);
System.out.println("String :" + sb);
```

→ Output → String : helloreader

deleted the space at 5th index

→ यहाँ hello reader था

→ अब helloreader है

→ String और StringBuilder में even spaces are counted as a character

6. // Replace the characters / substring b/w 5th index to 10th index with "bello"

replace

```
sb.replace(5, 10, "bello");
System.out.println("String : " + sb);
```

→ Output → String : hellobello

यहाँ helloreader था

7. // delete the substring contained between the start & end index

5 10

```
Sb.delete(5, 10);
System.out.println("String : " + sb);
```

→ Output : String : hello

(पहले hellobello था)

8. // Insert a character at 5th index

Eg1) Insert o at 5th index

```
Sb.insert(5, 'O');
System.out.println("String : " + sb);
```

→ Output → String : helloo

(inserted o at 5th index)

Eg2) Insert 'i' at 0th index.

```
Sb.insert(0, 'i');
System.out.println("String : " + sb);
```

→ Output → String : ihelloo

Eg3) Insert 'l' at 4th index

```
sb.insert(0, 'l');
System.out.println("String : " + sb);
```

→ Output → String : ihellloo

(→ 4th index)

Difference between setCharAt & insert

Sb = "hello"

setCharAt

```
Sb.setCharAt(1,'d');
System.out.println(Sb);
```

Output → hdllo

setCharAt उस index पर जो character है उसको given character से replace करेगा।

insert

```
Sb.insert(1,'d');
System.out.println(Sb);
```

Output → hdello

insert उस index पर character insert करेगा, बाकी उस index और आगे वाले सारे character 1 1 index आगे shift हो जाएंगे।

Ques-1

TOGGLE CASE

Q- We are given a string and are expected to change the case of each character from if lower case to upper case or, if, upper case, then to lower case

Eg !! pepCODinG → हमें सभी lower case letters को upper case letter में change करना है और सभी upper case characters/ letters को lower case letters में

↓

PEPcodING

* हर character का एक ASCII Code होता है।
* 'A' का ASCII code 65 होता है। +1
* 'B' का ASCII code 66 होता है। +1
* 'C' का ASCII code 67 होता है। +1
* 'D' का ASCII code 68 होता है। +1
* 'E' का ASCII code 69 होता है। +1

! & so on...

'Z' का ASCII code 90 होता है।

65-90

26 Capital Letter

* 'a' का ASCII code 97 होता है। +1
* 'b' का ASCII code 98 होता है। +1
* 'c' का ASCII code 99 होता है। +1
* 'd' का ASCII code 100 होता है। +1
* 'e' का ASCII code 101 होता है। +1

! & so on

'z' का ASCII code 122 होता है

97-122

Date

A को a में convert करने के लिए एक formula चाहिए।

a से A convert करने के लिए भी एक formula चाहिए।

'E' - 'A' = 4 ये difference 4 है

Similarly,

'e' - 'a' = 4 ये difference भी 4 है

तो हम कह सकते हैं कि दोनों equal हैं

$$'E' - 'A' = 'e' - 'a'$$

Now, after juggling with this equation, we get

Converting lowercase to uppercase

$$'E' = 'e' - 'a' + 'A'$$

'E' → uppercase; 'e' → lowercase

Converting uppercase to lowercase

$$'e' = 'E' - 'A' + 'a'$$

'e' → Lowercase; 'E' → Uppercase

Page No.
Date

Uppercase
$UC = LC - 'a' + 'A'$

$LC = UP - 'A' + 'a'$
Lowercase.

STEPS TO CODE:-

Step 1) Take the input String and pass the input String to ToggleCase function
(Name of function that we will create is ToggleCase)

Step 2) Since, we need to toggle (make changes) to the string, we will use a StringBuilder object.

We have to toggle each character, thus, we will traverse through every character using a loop (from index 0 till the last index).

Step 3) When we reach at some index; lets say 3, we store the character at index 3 in ch. Now, we have the character to toggle, we will now check, if it is an uppercase character or a lowercase character.

If the character is between A and Z (both inclusive), then its surely an uppercase character, otherwise we will check if its a lowercase letter using a similar condition.

Step 4) Let's take an example :-

Let ch = "f"

Then the else if condition would become true.

Using, `char uc = (char)(ch - 'a' + 'A');`

uppercase.

we will convert "f" to "F"

$$UC = LC - 'a' + 'A'$$

'f'

$= ('f' - 'a') + 'A'$

$= (102 - 97) + 65$

$= 5 + 65$

$= 70$

↳ returns an integer which needed to be be converted to char using explicit conversion.

Date

Since, the calculation, results in an integer corresponding to the required uppercase character (Type Casting - explicit type casting)

Step 5) Now, using setCharAt() function, we overwrite the character present at this index i with the character in the opposite case we managed to find.

→ जो result आया तो उसको (char)(result) explicitly Typecast करके setCharAt से ith index पर overwrite करदो।

Step 6) Now, return the new toggled string back to main().

Toggle case <code>

```
→ import java.io.*;
→ import java.util.*;
→ public class Main{
    public static String toggleCase(String str)
    {
StringBuilder sb = new StringBuilder(str);
for(int i=0; i<sb.length(); i++){
char ch = sb.charAt(i);
if(ch>='A' && ch<='Z')
{
  ch = (char)(ch+'a'-'A');
}
```

यह Togglecase function है।

→ इस String pass की जाती है as parameter

Given String को पहले StringBuilder में convert करो ताकि हम modification कर पाएं।

→ StringBuilder के हर एक character पर Traverse करना है हमें

→ ith index पर जो भी character है उसको ch में store करलो।

→ अगर char 'A' से 'Z' के बीच में मिला तो उसे lowercase में convert करो

→ यह explicit Type Casting हो रही है।

converting to lower case

अगर हमने else condition में enter किया है तो इसका मतलब ch तो lowercase character है और उसको uppercase में हमे convert करना है।

```
else
{
    ch = (char)(ch + 'A' - 'a');
}
```

converting to upper case.

```
sb.setCharAt(i, ch);
```

↳ उस ith Index पर जो char को changed character से overwrite करदो।

```
}
return sb.toString();
```

↳ StringBuilder से वापस String में convert करदो और return करदो।

```
}
public static void main(String[] args)
```

↳ ये main function

```
{
    Scanner scn = new Scanner
                  (System.in);
```

String में Input लिया

```
    String str = scn.next();
    System.out.println(toggleCase(str));
}
```

toggleCase function को call किया।

– Tamanna Verma

14 Oct 2021

DAY 41

STRING CONTINUED.....

Agenda

1. String Permutations → HW
2. String with difference of chars.
3. Introduction to ArrayLists → CW
4. Remove Primes.
5. Print Decreasing
6. Print Decreasing Increasing
7. Factorial
8. Power Linear.
9. Power Logarithmic

Ques. 1

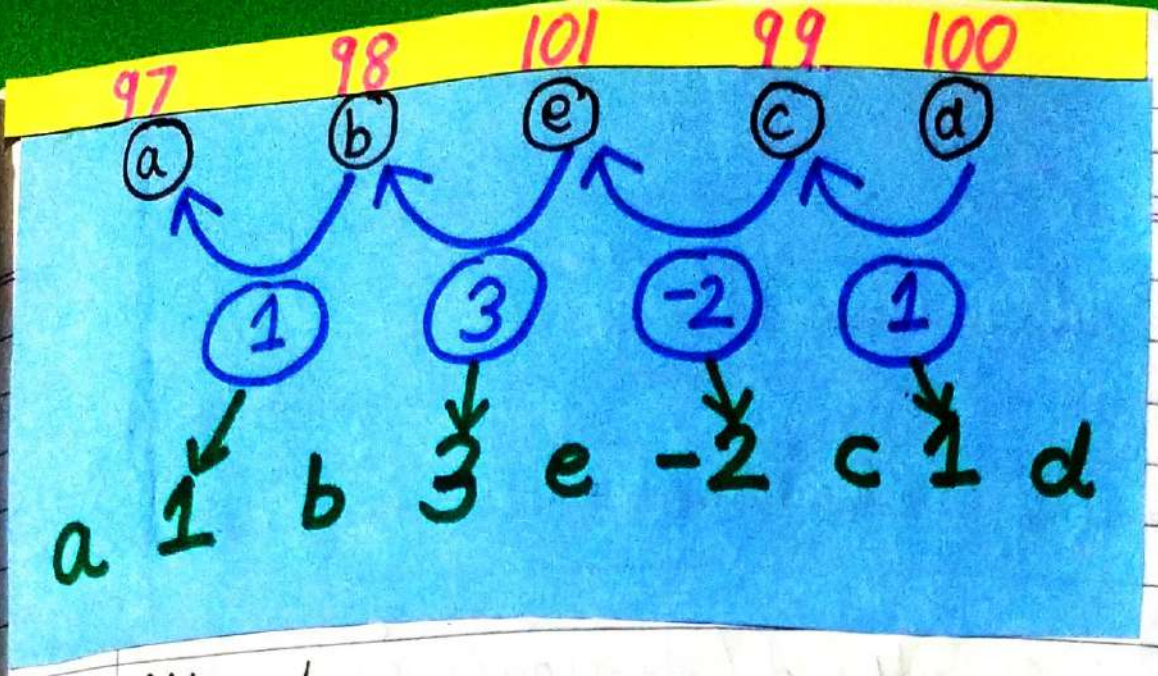
STRING WITH DIFFERENCE OF EVERY TWO CONSECUTIVE CHARACTER

→ We are given a string that contains only lower case & uppercase alphabets.

→ हमें एक String देनी है that contains the difference of ASCII values of every 2 consecutive character in between those characters.

Eg → For abecd

Answer → a1b3e-2c1d



Approach

1. We will use a StringBuilder and initially append the first character of the string.
2. Run a for loop from 1 to Str.length()-1 and at each index, find the difference between the current character and the previous character.
3. Append the difference & then append the current character.

The reason why we will be using a StringBuilder is that it is much easier to operate on StringBuilder as compared to String

To get the difference in the ASCII value, we can just simply subtract the characters Because implicitly all the characters are represented by their ASCII value, so while subtracting it is their ASCII value that gets subtracted.

Finally, we will append both the difference and the current char to maintain the form the output is expected.

```
→ import java.io.*;
→ import java.util.*;
  public class Main{
    public static String solution(String str){
→     StringBuilder sb = new StringBuilder();
→     sb.append(str.charAt(0));
→     for(int i=1; i<str.length(); i++){
→       char curr = str.charAt(i);
→       char prev = str.charAt(i-1);
→       int gap = curr - prev;
→       sb.append(gap);
→       sb.append(curr);
      }
→     return sb.toString();
    }
  }
```

String is the return type

Solution is func name

String is passed as the parameter

↳ empty string builder बना दी है।

sb में सबसे पहले 0th index वाला char डाल दो।

↳ 1 index से str length - 1 index तक loop चलाओ।

current element को और previous element को store कराओ curr और prev variable में।

current char का (ASCII value) में से prev को subtract करके save the differences ∵ Remember that characters are represented by their ASCII values

Now append this difference in the StringBuilder

Now append the current character in the StringBuilder

```
→ return sb.toString();
        ↳ String Builder से
          वापस String में convert करो
          और return करो।
    }
→   public static void main (String [] args)
    {
      {Scanner scn = new Scanner (System.in);
       String str = scn.next();
       System.out.println (solution (str));
      }
    }
}
```

* Difference b/w append & insert

append
↓
last index के आगे ही add करता है.

insert
↓
किसी भी index पर add कर सकता है।

Introduction to ArrayList → its a class

* अब हम ArrayList class को use करना सीखेंगे।

अगर हम एक array बनाते हैं 5 size का

```
int [] arr = new int [5];
```

तो अब ये array size 5 का ही रहेगा।
लेकिन Arraylist में बढ़ सकता है, कम हो सकता है size
हम जितना चाहे उतना size कम ज्यादा कर सकते हैं Arraylist का

शुरुआत में तो size 0 रहता है, अब हम इसमे जितनी भी size की value डालना चाहे, हम डाल सकते हैं।

```
ArrayList<Integer> list = new ArrayList<>();
```

हमने ऐसे declare किया है ArrayList को।

किसी भी datatype का Arraylist बना सकते हैं।

Name of Variable of Arraylist Type.

int नहीं लिख सकते यहा Integer ही लिखना होगा है with capital I. (small int fast चलते हैं, Capital Integer थोड़ा Slow चलते हैं)

Performance array जैसी ही देगा, बस लिखने में array जैसा नहीं है।

① → ArrayList<Integer> list = new ArrayList<>();
→ System.out.println
(list + " –> " + list.size());

↳ अभी तो empty है

अभी arraylist तो blank है

arraylist blank है तो Size भी 0 ही होगा

Size of ArrayList और Length of String print करते हुए हमे Paranthesis लगाना चाहिए।

Output →
[] –> 0

② → ArrayList में content add करने के लिए func होता है add()

→ list.add(10);
→ list.add(20);
→ list.add(30);
→ System.out.println(list + " –> " + list.size());

↳ Arraylist में add करता ये values.

Output →
[10, 20, 30] –> 3

③ → ArrayList में insert करने के लिए func होता है किसी भी index पर कोई भी value
add(index, value)

→ list.add(1, 1000);
→ System.out.println(list + " –> " + list.size());

→ [10, 1000, 20, 30] –> 4

④ → Arraylist से कोई भी value get करनी हो तो उसके लिए भी function होता है.

Page No.
Date

यहाँ वही लिखना है

get(index);

```
int val = list.get(1);   //not list[1]
System.out.println(val);
```

→ Output → 1000

array में list[1] लिखते थे लेकिन यहाँ get func use करना पड़ेगा हमें।

⑤ → Arraylist में value set करने के लिए भी set function होता है।

यहाँ यहाँ वही चलेगा

set(index, value);

```
list.set(1, 2000); // not list[1]=2000
System.out.println(list);
```

→ Output → [10, 2000, 20, 30] → 4

⑥ → Remove भी Arraylist का एक function होता है; इससे किसी भी index से value हटा सकते हैं।

```
list.remove(1);
System.out.println(list + " -> " + list.size());
```

→ Output → [10, 20, 30] → 3

⑦ → Arraylist हम Strings की भी बना सकते हैं।

```
ArrayList<String> l2 = new ArrayList<>();
l2.add("Hello");
l2.add("Bello");
l2.add("Cello");
System.out.println(l2 + " -> " + l2.size());
```

→ Output → [Hello, Bello, Cello] → 3

(8) → ArrayList में हम Loop भी लगा सकते हैं।

Page No.

Way 1. → ये for loop है।

```
for (int i=0; i< list.size(); i++)
{
  int val1 = list.get(i);
  System.out.println(val1);
}
```

for loop जो कि ArrayList के हर index की value get करती और print करादी

Displaying an Arraylist

Way 2 → Using for each loop

→ for each integer i in the list, print i

```
for (int i : list)
{
  System.out.println(i);
}
```

ArrayList in brief.....

Page No.
Date

1. ArrayList is a data-structure (just like array but very different).
2. In Java, Developers have written a class for ArrayList and its methods. Therefore, ∴ It is also called "ArrayList class" many times.

If you don't know what is class don't worry, इसके बारे में बाद में OOPS में पढ़ेंगे।

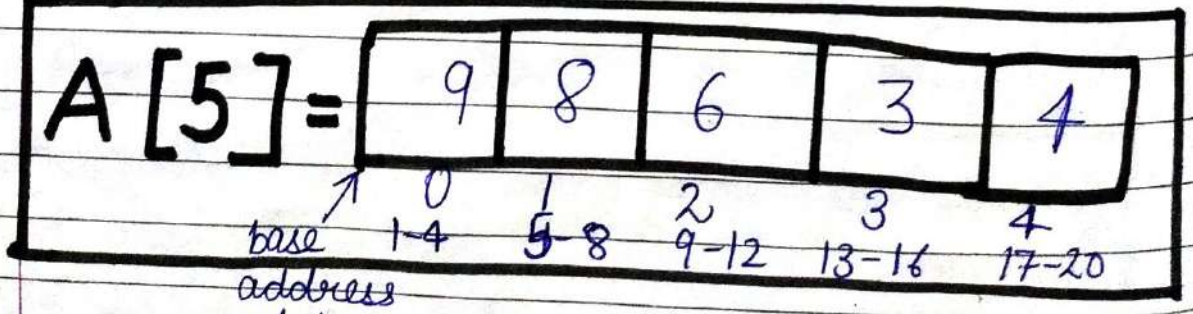
3. What is ARRAY List?

इससे पहले discuss करते हैं।

What is ARRAY?

An Array is a data-structure, which is a collection of similar data types which are arranged in a contiguous memory.

एकसाथ bilkul



हमने एक Array given है Size 5 का
ऊपर जो Array बनी है, उसमे हमे
ये तो दिख ही रहा है कि सारे elements
array के साथ store होते है (Therefore, we can
say that Array has contiguous memory).

इस Array का Base Address 201 है।
और हर Integer 4 byte का होता है।

∴ हर element 4 byte of memory space को
consume करेगा

∴ The Ist element present at Index 0
starts at memory address 201 & ends at
memory address 204.
The Second element at Index 1
starts at 205 memory address without
any gap of even a single byte in
between 2 index.

This happens in the entire array & so,
the memory is Contiguous.

ये सब तो हमे पहले से पता है,
फिरसे क्यो पढ़रहे है Array के Memory Allocation
के बारे में?

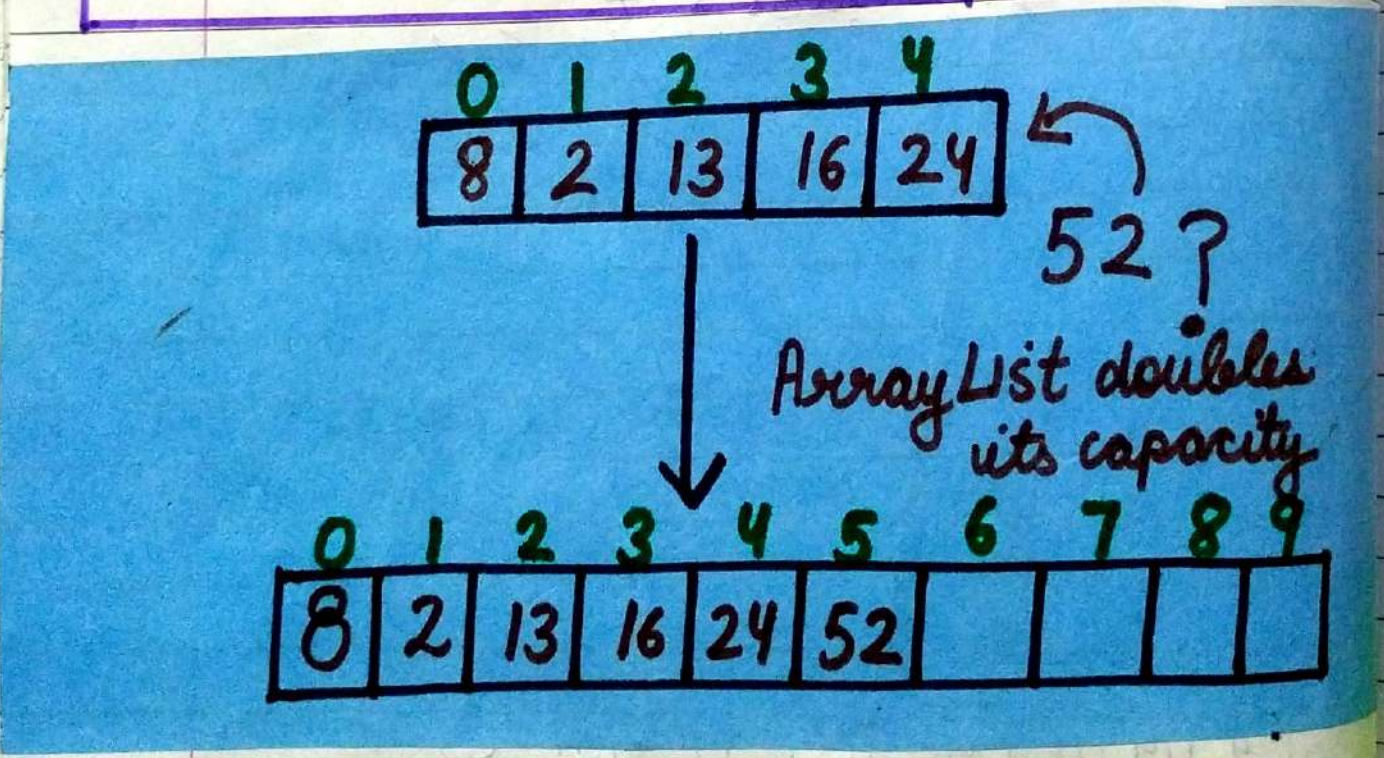
क्योकि ArrayList also follow these rules

* ArrayList will also have a contiguous memory, but there is a difference

Page No.
Date

We need to define the size of array for creating an array in the memory. But we do not define the size of an ArrayList. This is the major difference between Array and ArrayList.

4. How an ArrayList work?



So, there is something called as Size and Capacity

Size of an Array/ArrayList is the number of elements that are currently present in it.

Capacity is the total number of elements that can be present in the Array / ArrayList.

→ Size of the above ArrayList was 5 & the Capacity was also 5. So the ArrayList was complete.

अब अगर हमें उस completed ArrayList में 52 add करना है, तो अब कैसे होगा?

→ जैसा कि पहले भी बताया जा चुका है

ArrayList will manage the size itself.

लेकिन कैसे?

ArrayList actually doubles the total capacity of the List.

So, the current capacity was 5. When 52 was inserted, the capacity became 10 and after the insertion of 52, there are still 4 vacant spaces for insertion.

Doubled the Capacity

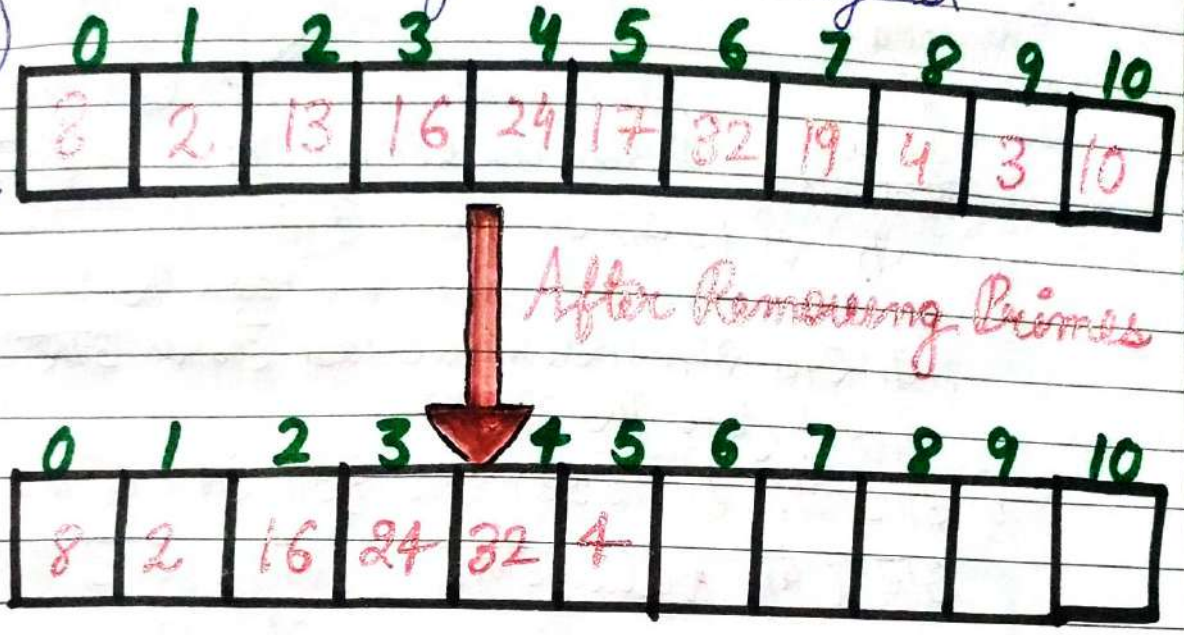
In previous ArrayList, After 52, we inserted 43, 36, 54, 92, and the arrayList did not expand as the capacity was enough to store all these elements. When we entered 7, the total capacity doubled again & became 20. Similarly, the capacity will not expand on entering 9 more elements.

The initial size of ArrayList is 0, as it is empty and the initial capacity of an ArrayList in Java is 10

(Ques-2) # REMOVE PRIMES

↳ We have to remove all prime numbers from an ArrayList.

Eg 1)



↳ This is an arrayList with both prime & non-prime no. As well. We have to remove all the prime no. from the arrayList.

↳ This should be the arrayList we get after removing prime no.

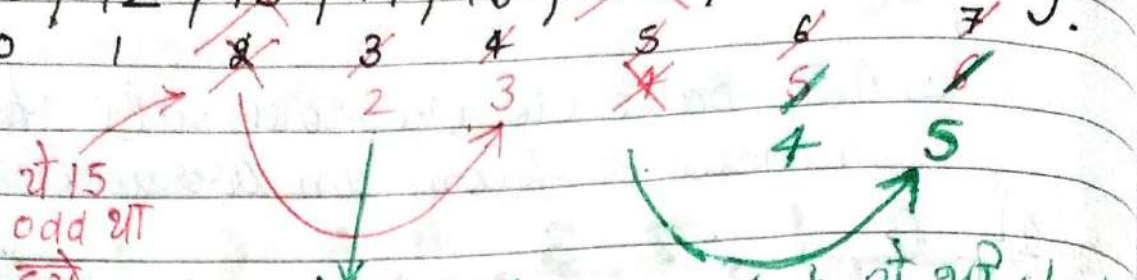
Approach

↳ ArrayList में element को remove करने के लिए कभी भी Loop सीधा नहीं चलाते, Loop को हमेशा उल्टा चलाते है।

To → Remove Odd

Eg → [10, 12, 15, 17, 18, 19, 21, 22].

0 1 2 3 4 5 6 7



ये 15 odd था इसे remove किया लेकिन

ये element तो कभी check ही नहीं हुआ

↳ ये भी check नहीं हुआ

a) जैसे ही 15 को remove किया जो element 3, 4, 5, 6, 7 index पर थे, वो अब एक index decrease होगए और 2, 3, 4, 5, 6 पर आगए

b) और जैसे ही 15 remove हुआ i++ हुआ, अब i की value होगई 2 से 3 और अब 3rd index से loop continue होगा, 2nd index वाला element 17 तो कभी check ही नहीं हुआ तो loop पूरा traverse करने के बाद भी answer तो गलत ही आएगा, because हम सीधा traverse कररहे हैं।

∴ Arraylist में element को remove करने के लिए Loop को हमेशा उलटा traverse कराओ। सीधा traverse किया तो answer कभी सही नहीं आएगा।

Page No.
Date

Pseudocode:-

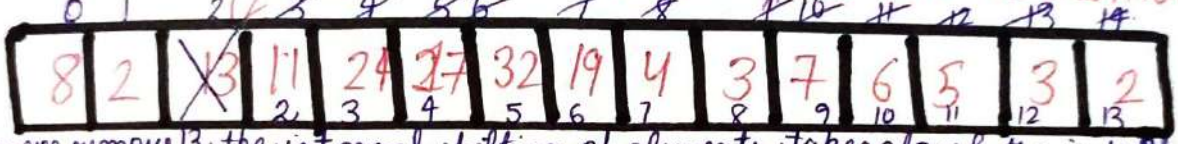
1) Declare a variable 'i' and keep it at the end of ArrayList i.e the the index at which it has its last value.

* Also, write the function for checking whether a number is prime or not.

2) Iterate through the ArrayList by decrementing the 'i' variable and keep on removing the prime numbers by checking the numbers with the help of your creation function

When we start iteration from FRONT↓

Since 13 is Prime, we have to remove it



When we remove 13, the interval shifting of elements takes place & the index has now 11, 11 is at Index 2 & it is also a prime no., but if we look at code carefully, we simplify increment the value of 'i' variable, so it became 3. This means, that in the next iteration, we will check whether the element at index 3 is prime or not, but we had a prime number at index 2 that we already missed. This causes a problem & due to this, all the prime no. might not get removed from the ArrayList.

How can we solve this issue? This issue can be solved by starting the iteration from back of the ArrayList.

When we start iteration from REAR

7 will be removed

4 took the index of 7 & index of 3 is not affected

The ith variable started from index 10. At index 10 it found 4, which is not a Prime no. So it moved backward at index 9 where it found 7, which is a Prime no. So, 7 got removed & 4 took its place/index and the index of 3 did not change. So, next time when 'i' will be decremented & it checks for Index 8 & find out that value 3 is also prime, 3 will also get removed. So when we iterate the Arraylist in reverse direction, there will be no issue

ARRAYLIST :- REMOVE PRIMES

WAY 1
REAR ITERATION

<CODE>

Iteration Start from END....

```
→ import java.util.*;
→ import java.io.*;
→ public class Main{
    public static void solution(ArrayList<Integer> al){
        for(int i = al.size(); i>=0; i--)
        {
            if(isPrime(al.get(i))
            {al.remove(i);
            }
        }
    public static boolean isPrime(int n)
    { for(int i=2; i*i<=n; i++)
    { if(n%i==0)
        {return false;
        }
    }return true;
    }
```

Integer type की ArrayList pass किया है as parameter

ArrayList को reverse order में traverse करना है।

is Prime function में array list के हर element को Pass किया अगर वो prime है → then remove that element from the arraylist

element prime है या नहीं है, ये check करके boolean return करेगा।

```
public static void main (String args)
{ Scanner scan = new Scanner (System.in);
  int n = scn.nextInt();
ArrayList <Integer> al = new ArrayList<>();
 for (int i=0; i<n; i++)
  { al.add (scn.nextInt()); }   Taking an Arraylist as input
  }
  solution (al);
  System.out.println (al);
  }
}
```

अगर हमे front traverse करने की ही condition given है तो हम तब एक prime element को remove करेंगे, साथ में i-- भी करेंगे ताकि तब element remove हो next, तो वो just next ith element को भी check कर सके।

WAY2 → REMOVE ALL PRIMES <code>

ITERATION START FROM FRONT

```
→ import java.util.*;
→ import java.io.*;
→ public class Main{
    public static void main( String [] args)
  { Scanner scan= new Scanner (System.in);
    int n = scn.nextInt();
    ArrayList<Integer> al= new ArrayList<>();
    for(int i=0; i<n; i++){
    {al.add( scn.nextInt());
    }
    solution (al);
    System.out.println(al);
    }
  }
→ public static void solution (ArrayList<Integer> al)
  { for(int i=0; i< al.size(); i++)
    { int val = al.get (i);
      boolean isThisValPrime = isPrime(val);
      if (isThisValPrime == true)
      { al.remove(i);
        i--;                    ] Most Important Step
      }
    }
  }
→ public static boolean isThisValPrime (int n)
  { for(int i=2; i*i<=n ; i++)
    { if( n%i == false)
      { return false; }
    }
    return true;
  }
```

HW

Page No.
Date

Ques-1)

Print All Permutations of a String Iteratively

Permutations of "abc" are :-

1. abc
2. bac
3. cab
4. acb
5. bca
6. cba

"abc" has 6 permutations

A general formula for permutations is (n factorial) where n is the length of the string.

* Assumption :- We are assuming that all characters are unique.

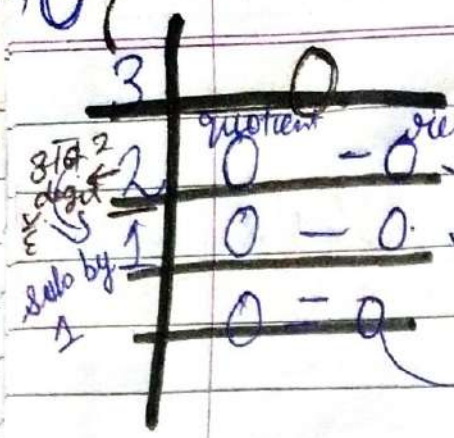
Approach.

1. सबसे पहले string की length check करो
   = 3.
   अब इसमें no. of permutation बनेगी = 13
   = 3X2X1
   = 6.

2. अब 6 permutations दें।

③ 6 permutation → 0-5 को 3 से divide करना शुरू करें 2 फिर 1 & for so on

for 0 → no. of digit in string

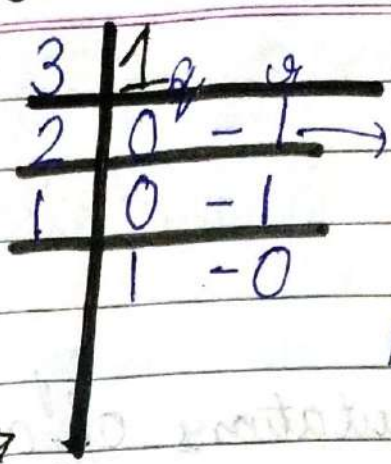


$a_0\ b_1\ c_2$

a
b
c

abc

for 1



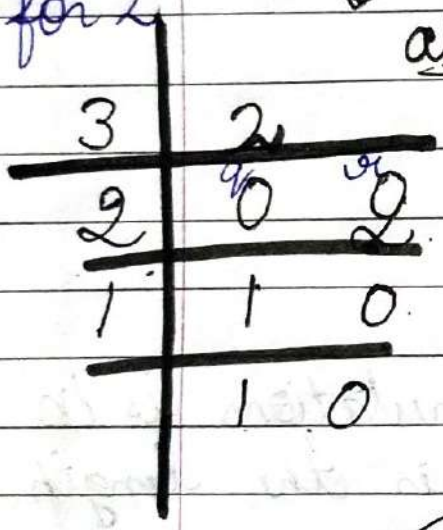
a bc
ac

अब पहले 1st index वाले element को फिर print फिर 1, फिर 0

bca

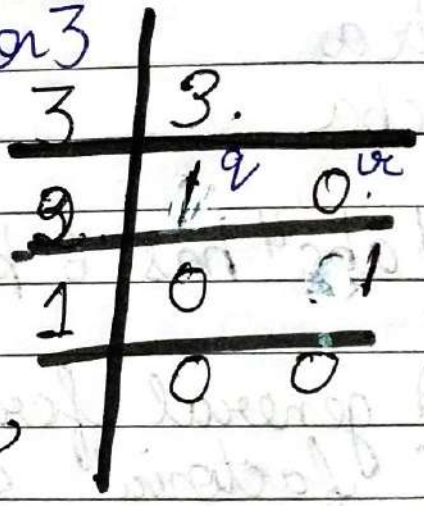
abc → ab → b

for 2



cab

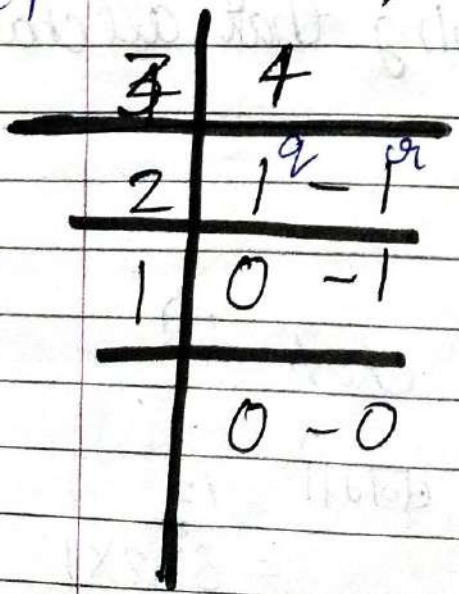
for 3



abc

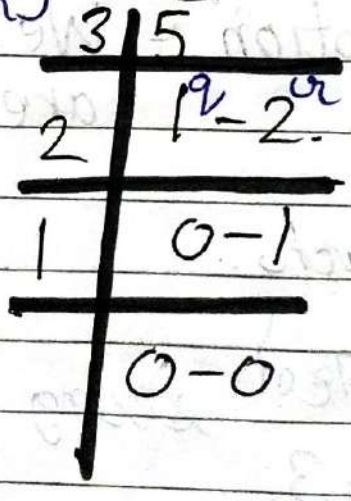
a.cb

for 4



bca

for 5



cb a.

Page No.
Date

Approach :-

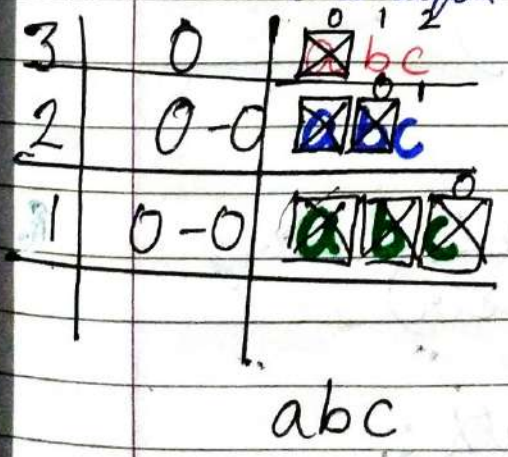
1. We will iterate through 0 to $(n-1)$

2. For each number, divide it by n, then by n-1 & then by n-2 & so on...

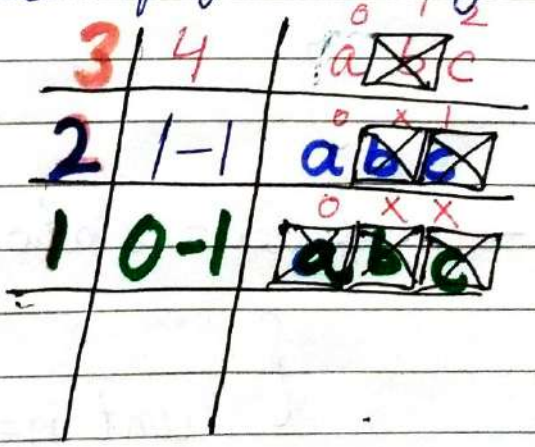
factorial n is represented as $n!$ & $\underline{n}$

3. Whatever comes as the remainder in each case, we will remove that char from the string, print it & use the quotient for next division purpose (we will not make any direct changes to the original string, we will makes changes to a copy.

4. This way, for every iteration 0 to $n!-1$, we will get one string of length n. & Interestingly all these strings will be unique permutation of given string



for 0
↳ 0 is being divided



for 4.
↳ 4 is being divided

<code>

```
→ import java.io.*;
→ import java.util.*;
→ public class Main {
    public static void main (String [] args)
    {
    Scanner scn = new Scanner (System.in);

    String str = scn.next();
    solution (str);
    }

→ public static int factorial (int n)
    {int val=1;
    for (int i=2; i<=n; i++)
      { val *= i;
      }
    return val;
    }

→ public static void solution (String str)
    {
      int n = str.length;
      int f = factorial (n);
```

Main function

Taken string as Input.

factorial function

To calculate the factorial.

```
for(int i=0; i<f ; i++)
{
  StringBuilder sb = new StringBuilder (str);
  int temp = i;
  for(int div = n; div >=1 ; div--)
  {
    int q = temp / div;
    int r = temp % div;
    System.out.print(sb.charAt(r));
    sb.delete(charAt(r));
    temp=q;
  }
  System.out.println();
}
```

quotient

remainder

length ↓ equal loop ↓ decrease

STRING DONE

— Tamanna Verma