DAY 36

9 Oct, 2021

DSA

अगर कभी String को define करके बताना हो तो ये बोला जा सकता है।

STRINGS

1st Discussion on String
- → declare
- → define
- → memory
- → charAt
- → length
- → subString

→ String is an object that represents an array or a sequence of character. A character (char) is a primitive data type, whereas string in Java is a non-primitive data type because it refers to an object. The String object has methods that are used to perform certain operation on String.

1. Declaration of a String in Java

Name of the datatype used to declare a string is "String" and not "string"

String S; } Declared a String

(S हमेशा capital होगा)

2. Definition of a String

1) String S = "pepcoding"; } Using String literal

2) String S = new String ("pepcoding"); } Using ("new") keyword.

So ये दो तरीके होते हैं हमारे पास किसी भी String को define करने के लिए

इन दोनों में क्या difference है? ये जानना भी बहोत important है।

1) Way 1 of Initialization → Using String Literal

Page No.
Date

String is a non primitive datatype
(ये एक primitive datatype (char) को मिलाके बनाई)

String name = "Tamanna" ; } ये हमने initialize किया है String को.

हम mostly यही तरीका use करेंगे initialization का

ये एक Class है।
Class क्या होती है वो बाद में सिखेंगे।
Java Developers ने String Class को बनाया है।
Class Java में datatype की तरह use होता है।

इस तरीके से direct initialize कर देना,
इसको बोलते हैं String Literal

→ Literals Java में काफी Type के होते हैं
जब हम एक datatype लिखते हैं और साथ उसके एक variable में कोई value assign करदे, इसको बोलते हैं Initialization using Literal.
Literal काफी Type के होते हैं eg→ 1) Integral Literal
2) Floating Point Literal
3) char Literal
4) String Literal

Integral Literal
↳ int x = 10;

String Literal
String s = "hello";

A Sequence of Characters within double quotes is referred as String Literal.

2) Way 2 of Initialization → Using "new" keyword

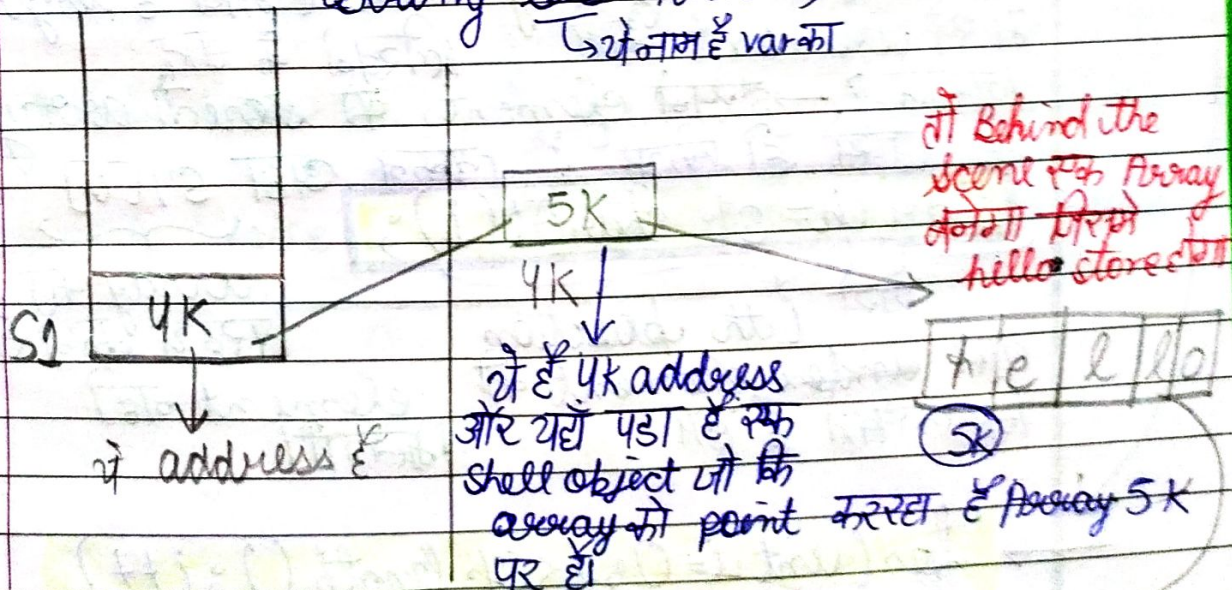
Page No.
Date

String s = new String ("hello");

↳ ये "new" Keyword है।
इससे एक new object बनजाता है
heap memory में।

How are Strings stored in memory?

String s1 = "hello";
↳ ये नाम है var का



अब ये s1 stored तो है एक array कि तरह लेकिन हम इसके element वैसे नहीं extract करसकते जैसे Integer array के करते थे s1[0] लिखके

X ये तरीका use नहीं होता Java में String के characters निकालने का, ऐसे हम C++ में करसकते है Java में नहीं।

Java में हमें charAt use करना पड़ेगा

→ s1.charAt(0)

Page No.
Date

Output in Strings

```
String s1 = "hello";
System.out.print(s1);
```

Print all elements of a String

a) तो पहले देखते है String के length के लिए कौनसा function है।

```
int len = s1.length();
```

function ↑ Paranthesis

Array मे ये Paranthesis नही आता जब array की length निकालते है।

b) दूसरी बात हमे ये देखनी है कि String मे से element कैसे निकाले? → इसने elements को store तो array के जैसे ही किया है लेकिन यहा s1[0]

Paranthesis लगाने है length लिखने के बाद

```
char ch = s1.charAt(i);
```

ऐसे ith position वाले element को निकाल सकते है हम

array की तरह ये करने से element नही मिलेगा।

```
for(int i=0; i<s1.length(); i++)
{   //char ch = s1[i]; this is wrong way
    char ch = s1.charAt(i);
    System.out.println(ch);
}
```

CHANGING DATA OF A Defined String IS Not Possible

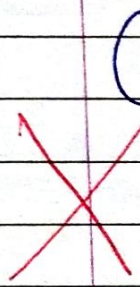
Page No.
Date

```
String s1 = "hello";
```

ये String s1 अब बनचुकी है, हम अब इसके Data को हम change नहीं करसकते।

We can extract the data but we can't change the data.

हम hello को bello नहीं बनासकते h को b set करके



ऐसा कुछ possible नहीं होता

```
s1[0] = 'b'.   } Not possible
s1.charAt[0] = 'b' → Not Possible
```

For now, इतना ही समझना अभी काफी होगा कि We can get the character]
We cannot change it.

एक तरीका होता है Data change करने के लिए, उसको बाद में देखेंगे लेकिन

```
s1.charAt[0] = 'b';
```

} ऐसा कुछ तो नहीं होता ये तो हमने जानलिया।

How to add more data in a String

```
String s1 = "hello";
s1 += " world";
s1 = s1 + " world";
```

(space) — एक ही बात है।

→ System.out.println(s1);

hello world.

→ System.out.println(10+20+"hello"+10+20);

30 + "hello"

तो अब 30 को String बनना पड़ा

"30 hello" + 10

अब 10 को String बनना पड़ेगा

"30hello10" + 20

अब 20 को भी String बनना पड़ेगा

30hello1020.

String और Integer मिलके बनते हैं String

charAt

~~S1[0] = "hello"~~ → ऐसा कुछ नहीं होता Data set /change करने के लिए

~~char ch = S[0]~~

✓ char ch = s1.charAt(0);

↳ अगर 0th element या character निकालना है तो

Substring

The substring() method returns the part of the string between the start & end index or the end

→ s = ("Hello");

Dono S छोटे होने चाहिए

s.substring (1,3); → el — end वाला include नहीं होगा

s.substring (0,4); ↳ hell

// 1,2 = Index
// 0,1,2,3 = Index

s.substring (0,5) → 5 तो कोई index नहीं है तो जहाँ तक है उसे वहा तक ही देगा i.e 0-4.

→ String s = "hello";

System.out.println (s.substring (1,3));

```
String s = "hello";
System.out.println (s.substring (1,3)); //el
System.out.println (s.substring (0,4)); //hell
System.out.println (s.substring (0,5)); //hello
System.out.println (S.substring (0)); //hello
System.out.println (s.substring (3));
```

0 से end तक

3 से end तक

Taking String as Input & then Print

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

Eg 1)

```
String s1 = scn.next();
```

→ next अगले space तक read करता है।

```
String s2 = scn.nextLine();
```

→ nextLine अगले enter तक read करता है।

```
System.out.println(s1+" "+s2); //Output
```

→ hello world hello . = Input

hello, world hello.
read by s1. s2.

Eg 2)

```
String s1 = scn.nextLine();
System.out.println(s1);
```

Input Output → Hello world

Hello world

Eg 3)

```
String s2 = scan.next();
System.out.println(s2)
```

Input Output → Hello

Hello world

Page No.
Date

→ String s = "Hello"

s.substring

↓ ↓

Both s will be small

→ s.substring (0,4) → selects index 0, 1, 2, 3

→ s.substring (0) → select 0 to end indexes.

→ s.substring (3) → select 3 to end indexes

Ques. 1

A substring can be formed by deleting 0 or more characters from start & end of string

A substring is a contiguous subsequence of that string

Print all Palindromic Substrings

→ एक String given होगी
→ उस String की सारी substrings को analyze करो
→ जो भी Palindrome substrings होगी उनमे से, उन सभी Palindromic substrings को print करदो।

Eg → a b c c

a	b	c	c
ab	bc	cc	
abc	bcc		
abcc			

ये सब होगी abcc की Substring

Palindrome वो होता है जिसका reverse उसके equal हो।

eg1 → ni tin

अब nitin को उलटा करदो Print

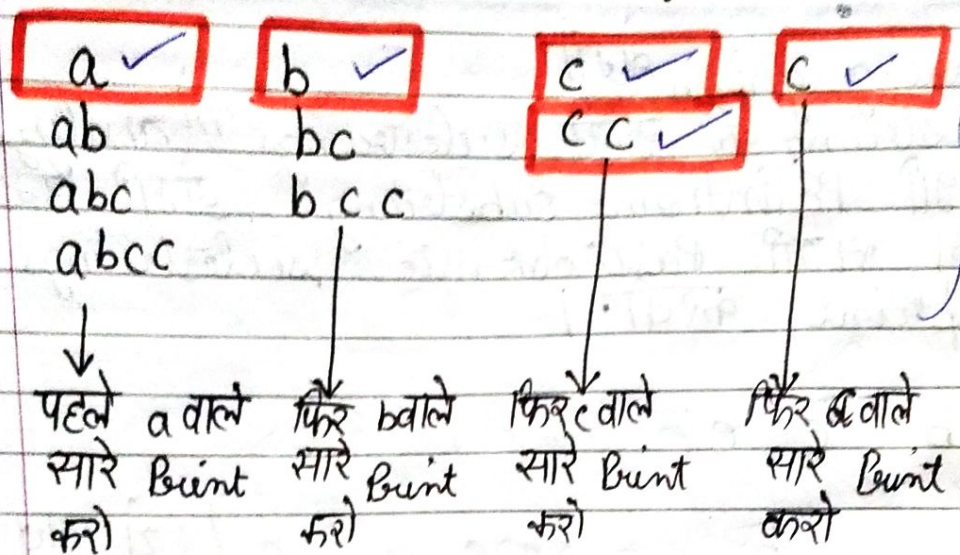
nitin ← ये दोनो तो equal हुए

eg 2 → malayalam

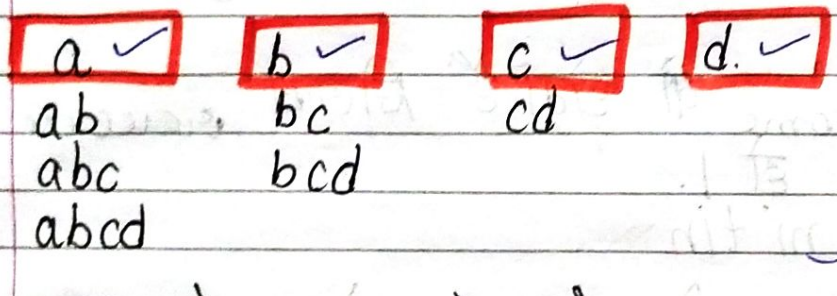
It is also a palindrome

तो जो आगे और पीछे से बराबर हो, वो String होगी Palindrome String.

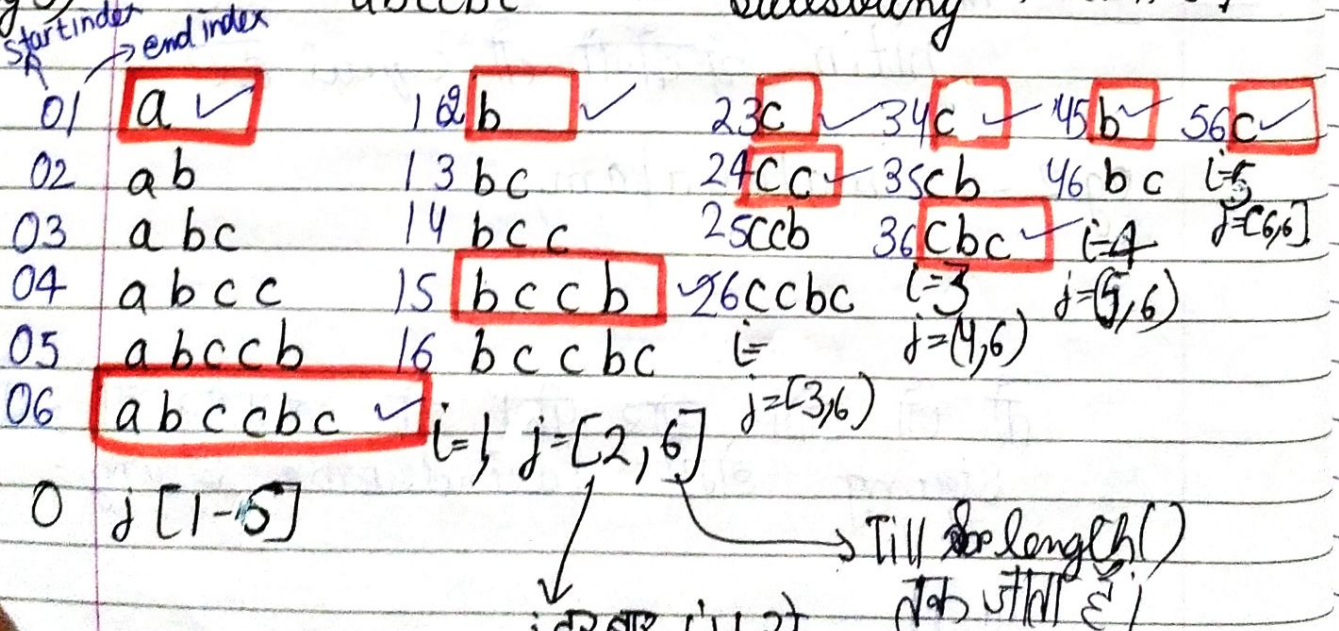
eg 1) अगर तो अब हमे abcc की Palindrome Substring print करनी है।



eg 2) अगर हमे abcd की सारे Substring निकालनी है।



eg 3) अगर हमे abccbc मे सारे Substring निकालनी है।



Page No.
Date

// Loop to print all the substrings

```
for(int i=0; i<s.length(); i++)
{
    for(int j= i+1; j<=s.length(); i++)
    {
        String sub = s.substring(i,j);
        System.out.println(sub);
    }
}
```

Function to check palindrome है कि नहीं → flag = true रखो
एक loop चलाओ जो while (left < right)
2 index लेते हैं एक left और एक right
और left वाला element right वाले के equal है. के equal नहीं
है तो loop से बाहर आजाओ break करदो, flag
को false करदो ;
और अगर left right element के equal है तो
left++,
right--
और loop अपने आप end होगया तो flag अभी भी
true है so return true else false.

Pseudocode

1. Initialize 2 pointers i & j as 0 & string length - 1 respectively.
2. Now run a loop until i<j :
   a) if character at index i is not equal to character at index j, then return false
   b) Increment i pointer by 1 & decrement j by 1
3. Return true

All Palindromic Substrings

```
import java.util.*;
import java.io.*;
public class Main
{ public static void main(String[] args)
  {Scanner scn = new Scanner(System.in);
   String s = scn.nextLine();
   for(int i=0; i<s.length(); i++)
     {
       for(int j=0; j<=s.length(); j++)
         {String sub = s.substring(i, j);
          boolean isPalindrome = IsPalendrome
                                 (sub);
   if (isPalindrome == true)
   {
    System.out.println(sub);
   }
}}}
public static boolean IsPalendrome(String sub)
{boolean flag = true;
 int left = 0;
 int right = sub.length()-1;
 while(left<right)
 {char chleft = sub.charAt(left);
  char chright = sub.charAt(right);
  if(chleft != chright)
   {flag = false;
    break;
   }
  left++;
  right--;
 }return flag;
}
```

Time Complexity
$= O(n^3)$

Space Complexity
$= O(n)$

Ques-2

STRING COMPRESSION

Page No.
Date

हमे एक String given है
aaa bb cc aa bbb ccc d ee

ये है String

हमे इस String पर 2 compressions करनी है i.e compression 1 & compression 2

↓
that is

Applying Compression 1 on given String
a b c a b c d e

इस compression में ये नही लिखना कि वो alphabets कितने no. of times occur हुआ

Applying Compression 2 on given String
a3b2c2a2b3c3de2,

↳ 1 लिखने की जरूरत नही है

इस compression में ये लिखना है कि वो alphabets कितने no. of times occur हुआ

Compression 1 S1 = aaa bb cc aa bbb ccc d ee

↳ String है S1 जिसमे ये stored है

Next element ↳ इसके जैसा नही है तो print

Next वाला इसके जैसा ही है तो नही करना Print

जब S1.charAt(i) equal नही हो S1.charAt(i+1) के तब उसे Print करना है और last index वाला element तो हमेशा ही Print करना क्योंकि ये एक side case में count होगा क्योंकि इसको किसी next से compare नही कर सकते

Page No.
Date

Compression 1

```
String ans = "";
for(int i=0; i<str.length-1; i++)
{
  if(str.charAt(i) == str.charAt(i+1))
  {
    // nothing
  }
  else
  {
    ans += str.charAt(i);
  }
}
ans += str.charAt(str.length()-1);
return ans;
}
```

→ last 2nd place को ही check करेंगे क्योंकि last को तो print करना ही है।

अगर next alphabet के equal नहीं है तो ans में add करते चलो उस alphabet के

Compression 2

इस Compression में वो सब तो होगा ही जो Compression 1 में था, एक extra काम भी करेंगे, इसमें हम बस Count को बढ़ाते चलेंगे i.e count++ अगर current element next element के equal हुआ तो, बस यही करना है कि count को बढ़ा दो और अगर नहीं है equal तो भी करना है हमें 1) (count को ++)
2) Print alphabet
3) Print count.

इसमें एक exception case होगा अगर count 1 है तो count print नहीं करना / अगर Count non-1 है तो Print

Compression2

```
String ans = "";
int count = 1;
for (int i = 0; i < str.length() - 1; i++)
{
    if (str.charAt(i) == str.charAt(i+1))
    {
        count++;
    }
    else
    {
        ans += str.charAt(i);
        if (count > 1)
        {
            ans += count;
            count = 1;
        }
    }
}
ans += str.charAt(str.length() - 1);
if (count > 1)
{
    ans += count;
}
```

```
→ import java.util.*;
→ import java.io.*;
→ public class Main{
public static void main(String args[]){
  Scanner Sch=new Scanner(System.in);
  String S= scn.next();
System.out.println(compression1(S));
System.out.println(compression2(S));
}
```

Page No.
Date

Method 1st function String return type

```
public static String compression1(String str)
{
  String str = "";   → Blank string
  for(int i=0; i<str.length()-1; i++)
{
  if(str.charAt(i)==str.charAt(i+1))
  {
    //do nothing
  }
  else
  {
    ans+=str.charAt(i);
  }}
  ans+=str.charAt(str.length()-1);
  return ans;
}
```

```
public static String compression2(String str)
{ String ans = "";
  int c = 1;          -> 2) count &
  for(int i=0; i<str.length()-1; i++)
  {
    if(str.charAt(i)==str.charAt(i+1))
    {
      c++;
    }
    else
    {
      ans += str.charAt(i);
      if(c>1)
      {
        ans+=c;
        c=1;
      }
    }
  }
  ans += str.charAt(str.length()-1);
  if(c>1)
  {
    ans+=c;
  }
}
```

Time Complexity :- We are traversing the input string once, hence the time complexity will be O(n).

Space Complexity :- If we are taking an extra space for output (i.e String ans), then the solution is taking O(n) space. If we ignore the space taken by output (which we generally do), the soln is said to take O(1) space.

String interning and Immutability

हमें 2 चीजो को इग्नल से समझना होगा कि Interning किसे बोलते है और Immutability किसे?

1. What is Interning?

एक String बनाई S1

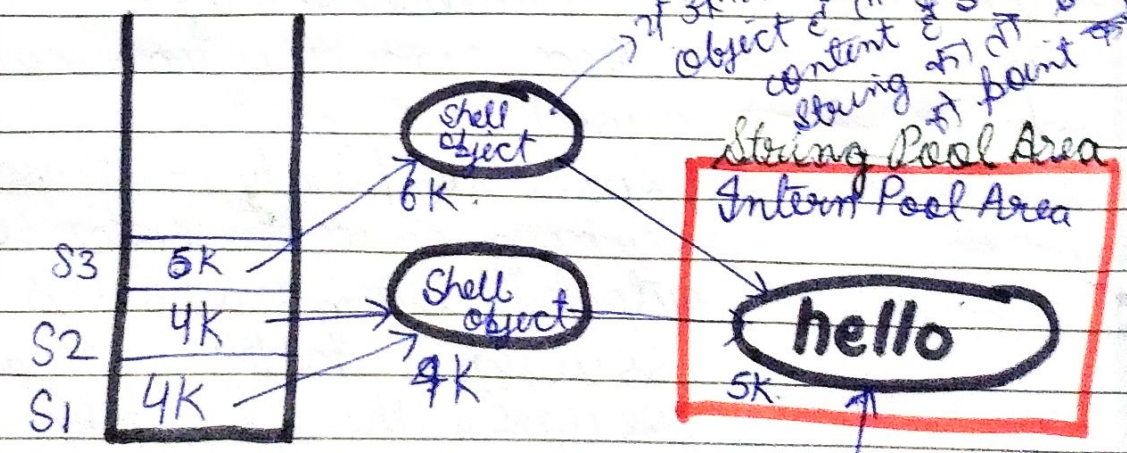
```
String S1 = "hello";
```

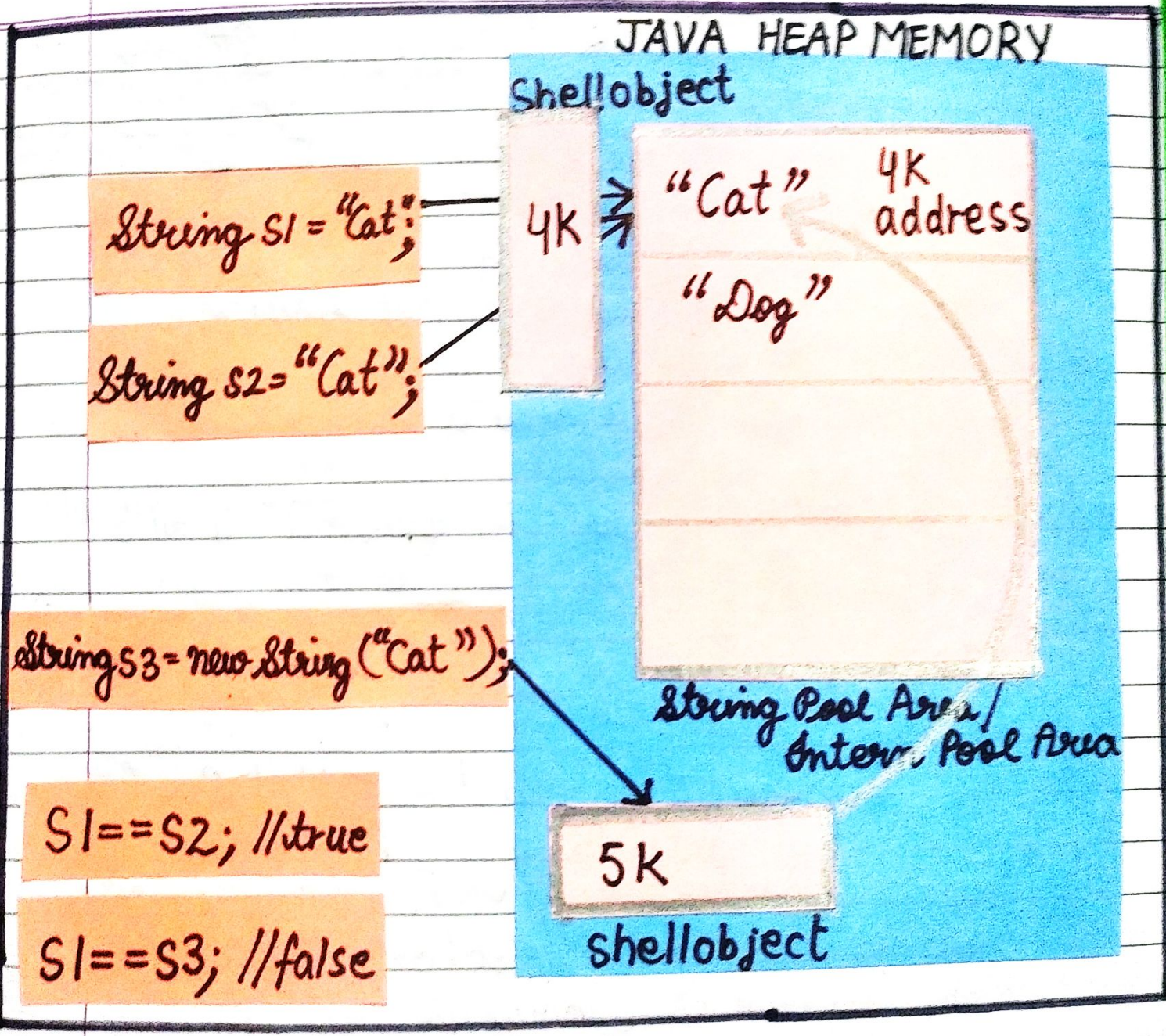
एक String बनाई S2

```
String S2 = "helo";
```

एक String बनाई S3

```
String S3 = new String("hello");
```




JAVA HEAP MEMORY
Shellobject
String S1 = "Cat";
String S2 = "Cat";
4K
"Cat"
4K address
"Dog"
String S3 = new String ("Cat");
String Pool Area / Intern Pool Area
S1==S2; //true
S1==S3; //false
5K
Shellobject

S1 String बनाई, तो S1 नाम का variable बना उसमें Shell object का address था 4K, वो shell object Hello String को Point करता है
↳ जो कि array जैसी है String.
↳ 5K address पर

तो इसका मतलब

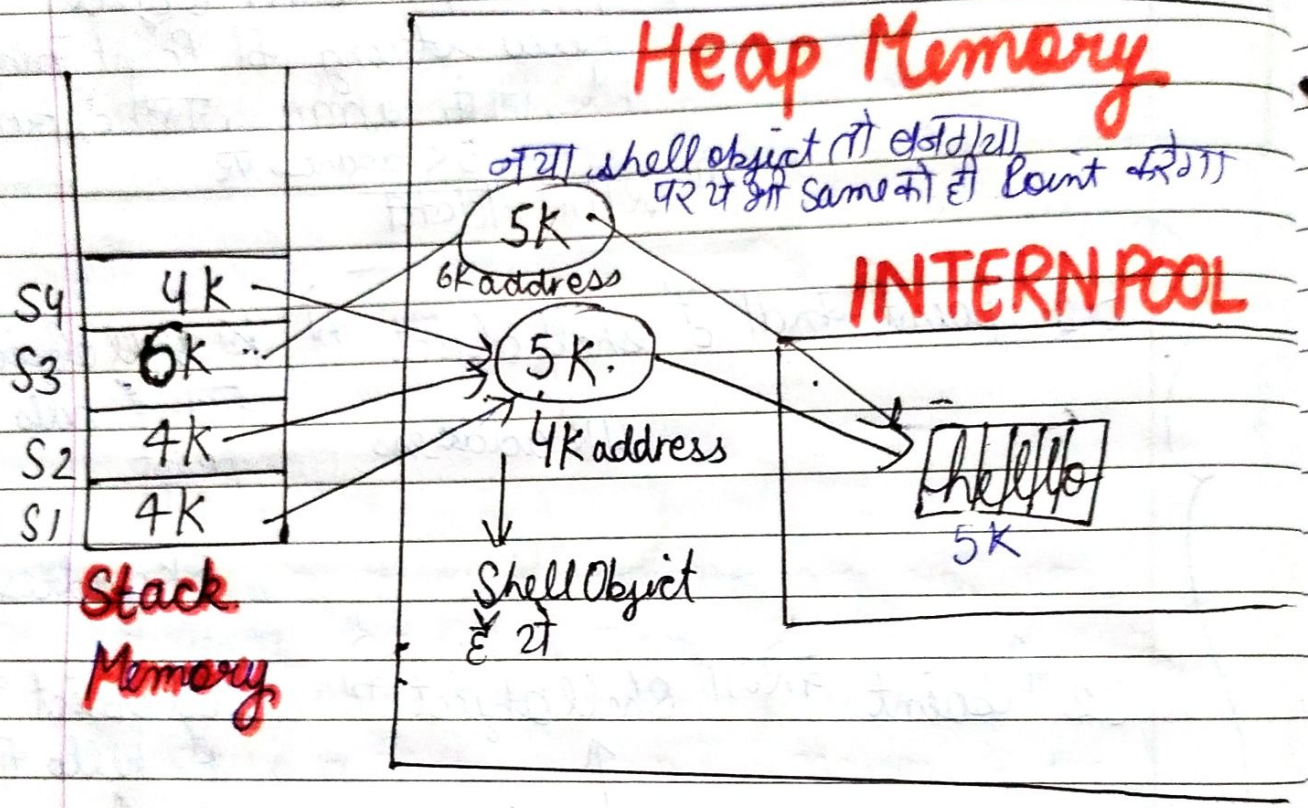
S1 point करता है shell object को, shell object करता है hello को point
↓ ↓
4K address 5K address.

S2 भी point करेगा shell object को, shell object करता है hello को point
↑ ↑
with 4K address with 5K address

→ तो इसका मतलब दोनों String एक ही Hello को point कर रही है।
इसको बोलते हैं Interning।
अगर same content है, अलग-अलग String भी एक ही जगह point करेगी।

→ S3 point करता shell object को, shell object करता है hello को point
↑ ↓
with 6K address with 5K address

String S1 = "hello";
String S2 = "hello"; यहाँ Second Line चलती तो इसे पता है कि पहले से hello है. यह Interning हो रही है।
String S3 = new String ("hello");
String S4 = S1;



1. * There are mayorly 2 types of memory associated with a program, that are stack memory & heap memory associated with a program.

2. * You should know that only Primitive data type : Byte, Short, Int, Long, Double, Float, Char and Float, Boolean are stored directly in Stack memory.

3. * Since, Strings are non-primitive data type in Java, hence, they are not stored directly in stack memory.

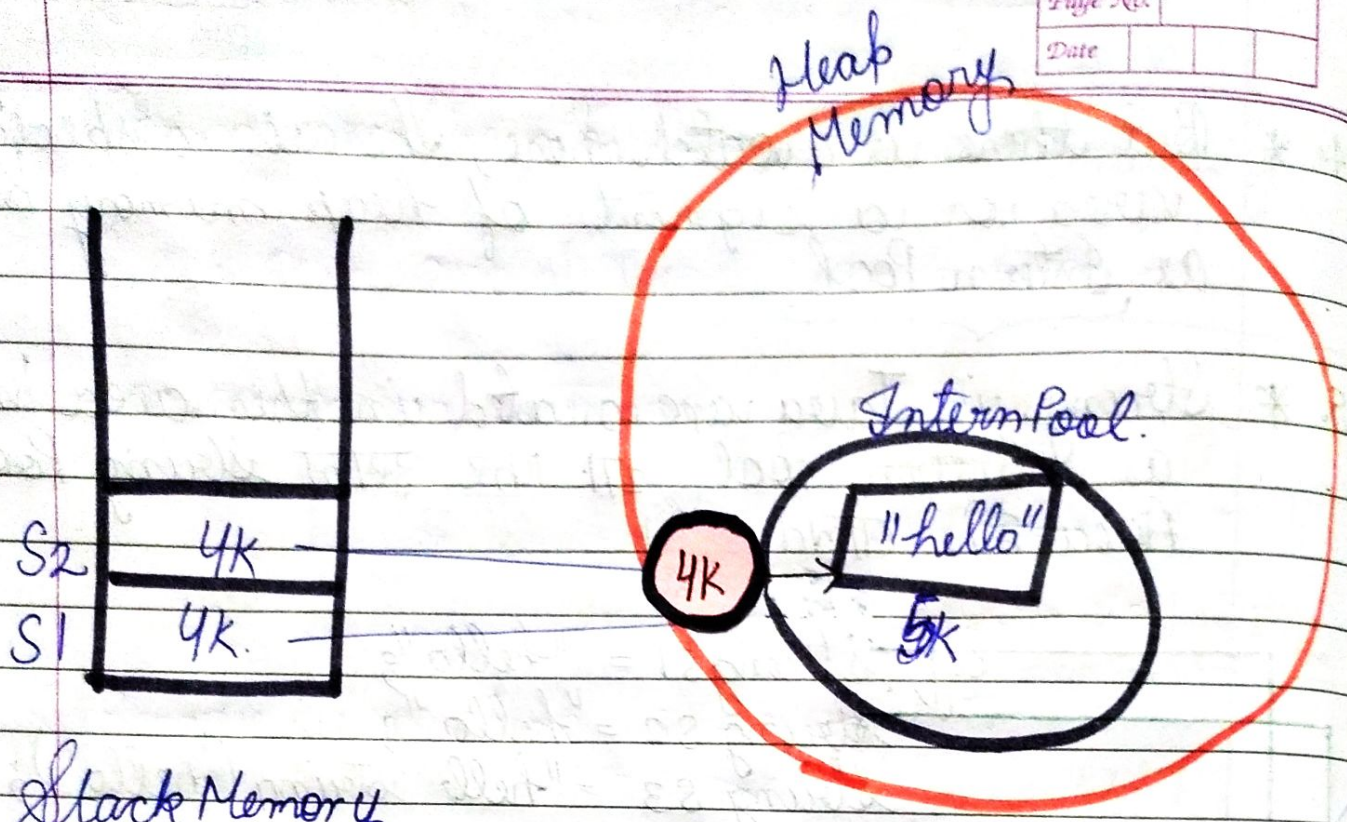
4. * But there is a catch here, There is a special area or a segment of heap memory known as Intern Pool.

5. * Strings in Java are created in this area known as "Intern Pool" या फिर इसको String Pool Area भी बोलते हैं।

Eg) String s1 = "hello";
String s2 = "hello";
String s3 = new String("hello");

→ This line will make a string of 5 characters "hello" in the intern pool.
Lets suppose, at starting address = 4K
And this 4K address is Stored in the Stack Memory with variable name s1. 4K shell object point to char array at 5K address

→ This line will search for a character array {'h', 'e', 'l', 'l', 'o'} in the Intern Pool Area. If the very same character array or String already exists (which in this case, Yes, as string at location 4K = "hello" is already present), then JVM (Java Virtual Machine) will not create another String ("hello" in the Intern Pool. Instead, it will store the reference of the same 4K location to the same shell object for the s2 string as well.
Hence, now 2 strings s1 & s2 point to same "hello" literal in the Intern Pool at 5K address



Stack Memory

* If the String "hello" would have not been already present, then JVM would have created another string literal and allocated a new memory location to the String.

This process of searching for an identical string literal in the Intern Pool, & if present, sharing the same memory of string literal for different Strings is known as

String Interning in Java

Why is Interning Done?

↳ To save space

String s3 = new String ("hello");

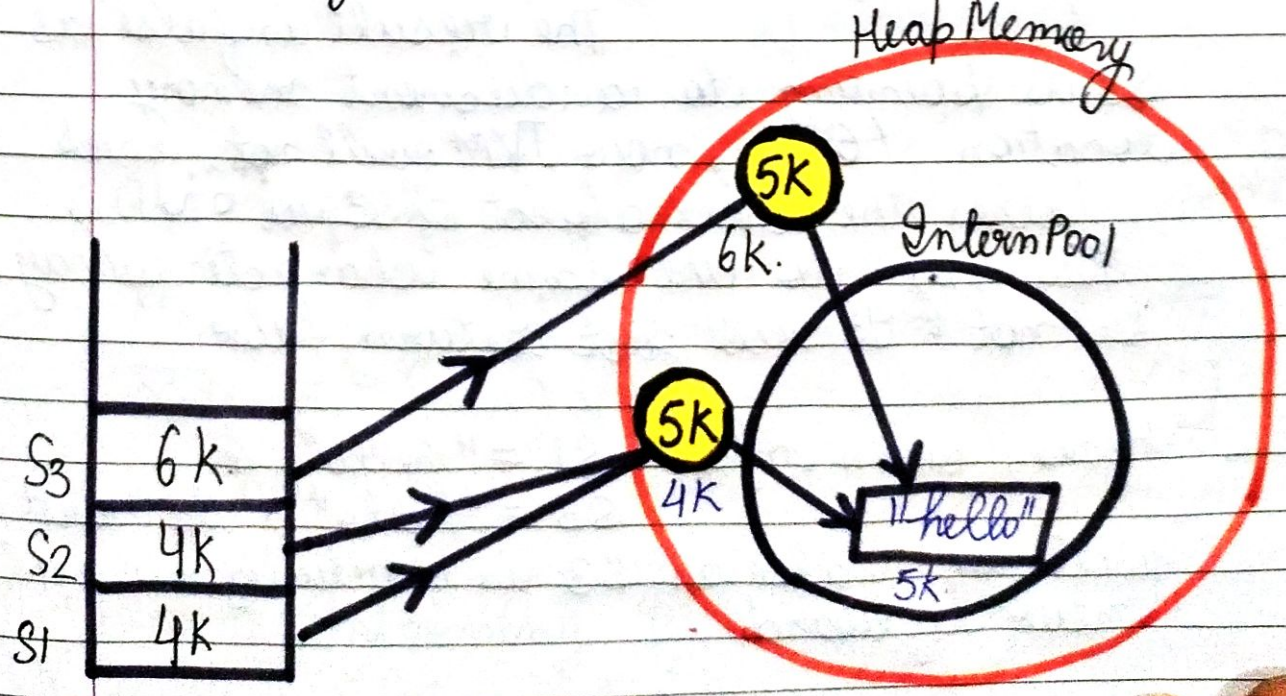
→ Now what will this line do?

* We have created the String using new keyword. This will create a new shell object outside of the Intern Pool (with a new address for the string.

* But there will be still a problem, if, there will be an identical character array, then this shell object will also point to the same character array.

*

→ For this line, a new shell object will be created at location, say 6K, but in this case also, the shell object will point to the same character array at location 5 K, since, it contains the string ("hello").



IMPLICATIONS OF INTERNING OF STRING

1) हमे अब .equals() का use करना पड़ेगा

Comparision Operator (==) vs equals() method

We should avoid using Comparision Operator (==) for comparing strings

(==) → इसको Comparision operator बोलते हैं।

==Operator

Because if we check S1 == S2 True

The result is True, Because In Stack Memory location, both S1 & S3 are pointing to 9K memory location which is same for both so the JVM will return True.

But if we check S1 == S3 False

The result is false as S3 is pointing to a different memory location (6K), now JVM will not check whether the shell object of S3 at 6K is pointing to the same character array or not. It will just return false.

Hence, even after S1 = "hello" & S3 = "hello", the result will be false as S3 is initialized using new keyword.

.equals()method

To compare strings in Java, we should make a habit to always use .equals() method.

* S1.equals(S2) will directly return true as both point to the same (4K) memory location.

Now, S1.equals(S3) will not return false if both point to different memory location, instead they will now be checked whether they are equal character-by-character or not.

Since both are ("hello"), it will also be True.

S1==S2 True

S1==S3 False

S1.equals(S2) True

S1.equals(S3) True

तो ये first Implication थी Interning की कि Strings को compare करने के लिए (मतलब ये फर्क पड़ा)
अब हमें (==) की जगह .equals() method का use करना पड़ेगा।

तो Java में String Interning करने से ये एक फर्क पड़ा
2) Strings Immutable बनगयी

ये Second Implication है String Interning की।

2. STRINGS ARE IMMUTABLE

Change नहीं कर सकते

JAVA HEAP MEMORY

STRING POOL/ INTERN POOL

String str = "ab"; → "ab"

str.concat("cd"); → "abcd"

↑ ये भी O(n) है।

String is immutable in JAVA

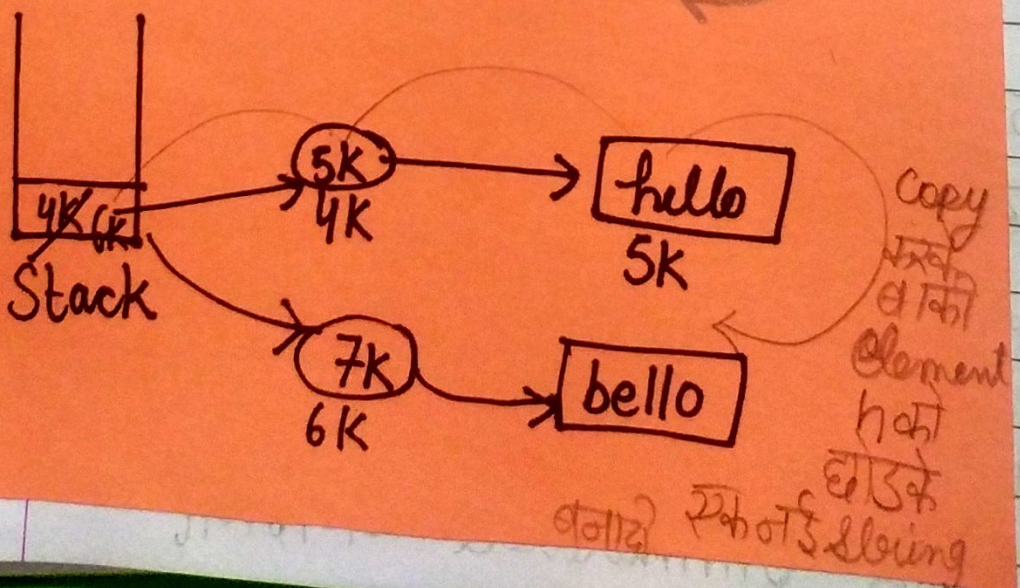
Replace function

Performance Issues

String S = "hello";

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

↑ ये O(n) है।



Page No.
Date

Ques.1
What is Immutability of Strings

→ Immutability का मतलब अगर हम English Dictionary में देखेंगे तो मिलेगा
'which cannot be changed or modified.

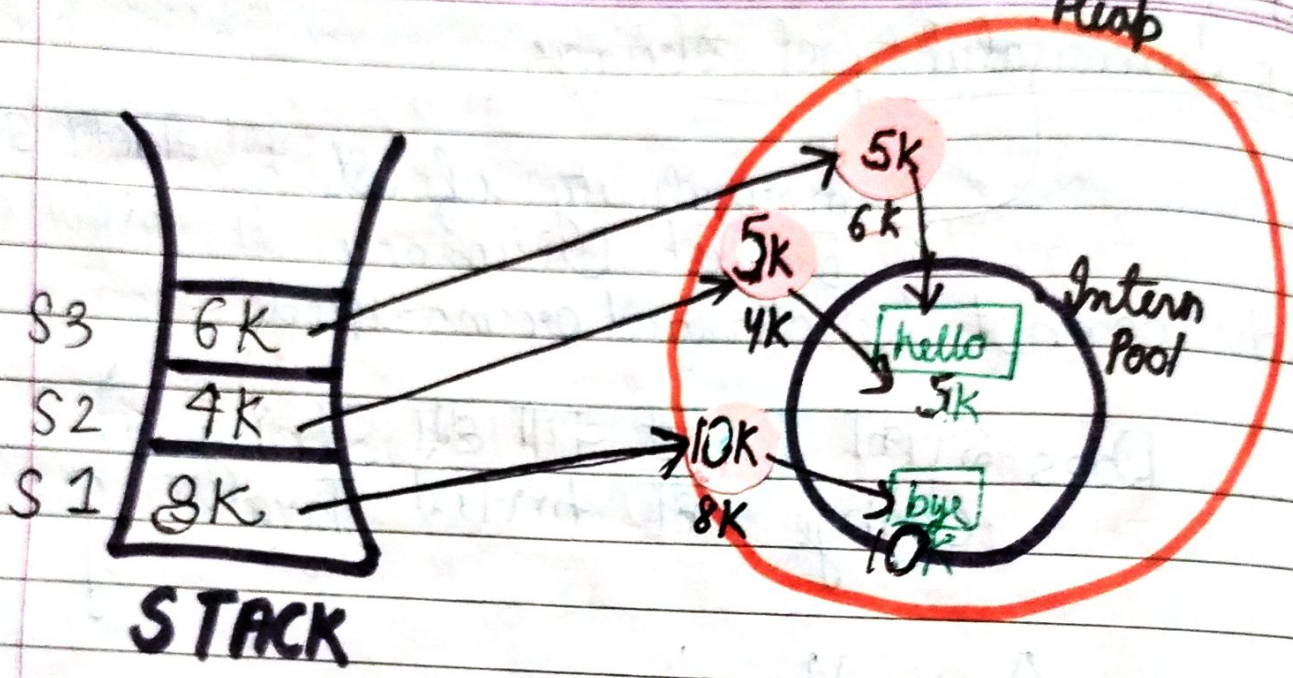
इसका मतलब क्या हम String को change नहीं कर सकते Java में ?

Actually Reference is Mutable, ↳ हो सकता है change
Instance is not.

एक बार अगर एक String Object बनगया, तो उस object का state या data हम change नहीं कर सकते लेकिन String can point to another character array

If we try to modify S1 as: S1 = "bye";
then what will happen? The data at 4K "hello" will not change
Now, the data at 4K will not change.
S2 will still point to 4K address le hello

Instead, a new object will be created with a character array {'b','y','e'} & S1 will now start pointing to new address location



Hence, we can say that

Reference of String (In Stack Memory) is Mutable.

Address Pointing To

can be changed

But, The Instance (Character Array in Intern Pool is Immutable.

Ques 2.

Page No.
Date

Why does Immutability occur in the first place?

Let's assume, if you are allowed to modify the instance of String (characterArray) ie modify the character array {'h','e','l','l','o'} in the intern pool, then what would have been the consequences?

Since, the character array at 4K location is pointed by more than 1 one String, modifying the character array itself would have led to the modification to all the strings which are pointing it.

for eg → If we would have changed the S1's character at 2nd index to say 'd', then character array would have been modified to {'h','e','d','l','o'}, hence the character at 2nd index for String S2 would have also been changed to 'd'.

* This can be a surprise to the developer who does not know about the existence of String S1, he/she, is working on a module which use String S2. He would wonder why S2 has now became "hedlo" like he himself initialized it to "hello".

∴ To avoid such such surprises, it is not allowed to modify the instances of Strings in Java.

STRINGS ARE IMMUTABLE IN JAVA AS AN IMMEDIATE CAUSE OF INTERNING

Date

Ques. 3. What is the impact of Immutability?

Immutability can lead to low performance issues, both in the term of Space & Time

↳ It means both Time & Space Complexity increase if we have the concept of Immutability.

Eg →

```
Scanner scn = new Scanner(System.in);
int n = scn.nextInt();
String s = "";
for(int i=0; i<n; i++)
{s+= i;
}
System.out.println(s);
```

What can be the time complexity of this code?

Will it be O(n) as there is 1 loop or as we are adding only 1 character n times

?

No, the time complexity of this code will be $O(n^2)$

This is because the Instance of String cannot be modified, hence adding a character to the end of string does not mean it will get appended in the end of character array.

Instead, even if a single character is to be appended, first JVM will create a **new** string, and then append the single character to the newly created string object. Hence, we are first copying the entire string which is $O(n)$ work even for appending a single character

* And since, we are appending since character for n times, the overall time complexity will turn out to be $O(n * n) = O(n^2)$

for eg

To add a single character 'd' to string s = "hello", we have to first make a new string object then we are first copying the entire char array {'h', 'e', 'l', 'l', 'o'} to the new string & append d to the new string.
Hence, we are not operating on 1 character but s.length() + 1 character everytime we adding a single char.

So, consider the current string is taking 2GiB space in memory, then adding a single character will lead to creation of a copy of the String of entire 2GiB & then appending 1 character of few bytes (s) to it.

→ So, this is not what we expect JAVA to do for us.

Is there any way to make String mutable in Java?

↳ Instance को भी change किया जा सके।

Yes, there is a concept of StringBuilder Class

↳ इसको हम आगे पढ़ेंगे।

अब से हमे जहाँ जहाँ String use करना होगा Strings are problematic, तो instead we will use StringBuilders.

C++ की Strings = Java में StringBuilders, वहाँ कोई issue नही होता
equally good

Interning

- **What**
  - Searching for an identical String literal in Intern Pool, if there, then share same memory
- **Why**
  - To save Space
- **Implication**
  - String comparision के लिए == use नहीं कर सकते
  - equal() method use करना चाहिए
- **Implication 2**
  - Strings are Immutable.

Immutability

- **What**
  - Reference are mutable
  - Instances are Immutable
- **Why**
  - Because of Interning, many string points to the same Instance, changing 1 would change many (to avoid that we are not allowed to change any instance
- **Implication 1**
  - Lack of functions ही String का data change करने के func नहीं मिलेंगे। या तो func नहीं मिलेगा या मिलेगा तो Performance Issues
- **Implication 2**
  - Performance Issue (Data change करने के लिए कोई भी func use करलो Time और Space Complexity बढ़ेगी।
- **Solve Implication 1 & Implication 2**
  - StringBuilder