★ STRINGS

* Character type arrays are called Strings.
* Strings are immutable i.e., if we have to change anything we have to make new string.
* We can create String by using :-
  - ⇒ String str = "abcd";
  - ⇒ String str2 = new String("xyz"); (or)
  - ⇒ char arr[] = {'a', 'b', 'c', 'd'}; (or)

* Input/output of Java

```
Scanner sc = new Scanner(System.in);
String name;
name = sc.next();  or  name = sc.nextLine();
// it prints only a word.   // it prints whole line.
System.out.println(name);
```

* String Length : ⇒ String-name.length();
  - * If also count space.
  - Ex ⇒ Syso (name.length());

* String Concatenation

When multiple strings combine & become one string.

```
⇒ String firstName = "Atrubhav";
  String lastName = "Singh";
  String fullName = firstName + " "
                    + lastName;
  Syso (fullName);
```

* Char At()

It means character at index ().

- It used to find single characters of string.

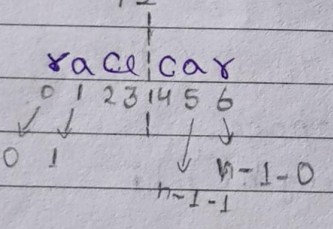
⇒
```
String firstName = "Anubhav";
String lastName = "Singh";
String fullName = firstName + " " + lastName;
Syso (fullName.charAt (1));
```

Output → n

OR

⇒
```
public static void printLetters (String str) {
    for (int i=0; i < str.length(); i++) {
        Syso (str.charAt (i) + ", ");
    }
    Syso();
}
```

Ques 1. Check if a String is a Palindrome.

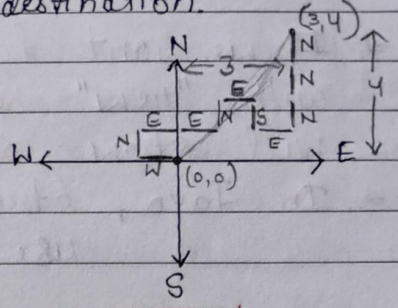


→ for (int i=0 to length/2)
compare str(i) == str(n-1-i)
⇒ O(n/2)

⇒
```
public static boolean isPalindrome (String str) {
    for (int i=0; i < str.length()/2; i++) {
        int n = str.length();
        if (str.charAt (i) != str.charAt (n-1-i)) {
            return false;
        }
    }
    return true;
}
```

Ques 2. Given a route containing 4 directions (E, W, N, S), find the shortest path to reach destination.
Displacement

"WNEENESENNN"



Shortest Path $= \sqrt{(x_2-x_1)^2 + (y_2-y_1)^2}$

$= \sqrt{(3-0)^2 + (4-0)^2}$

$\sqrt{9+16}$

$\sqrt{25}$

5

N ↑	y++
S ↓	y--
W ←	x--
E →	x++

⇒
```
public static float getShortestPath (string path) {
    int x=0, y=0;
    for (int i=0; i<path.length(); i++) {
        char dir = path.charAt(i);
        //south
        if (dir == 'S') {                    // Time Complexity
            y--;                             O(n).
        }
        //north
        else if (dir == 'N') {
            y++;
        }
        //West
        else if (dir == 'W') {
            x--;
        }
        //east
        else {
            x++;
        }
    }
    int x2 = x * x;
    int y2 = y * y;
    return (float) Math.sqrt(x2+y2);
}
```

* String Comparision

If we want to create new string then create with "new" keyword because else string will show old strings.

⇒ In Java, strings when compare they don't use (==).

Code:
```
if (s1.equals(s3)) {
    Syso ("Strings are Equal.");
} else {
    Syso ("Strings are not Equal.");
}
```

* String function - SUBSTRING

Substring are the specific part of the Original string.

Ex: "Hello World" ⇒ Substring (0,4)

0 1 2 3

↓

Hell

⇒
```
public static string substring (string str, int si, int ei) {
    string substr = "";
    for (int i = si; i < ei; i++) {
        substr += str.charAt(i);
    }
    return substr;
}
```

In Java, there is built-in Java substring function

i.e., str.substring(si, ei);

str → string name; si → start index; ei → end index

Ques. For a Given set of Strings, print the largest string. "apple", "mango", "banana".

* Lexicographic or Lexical order used for sequence of ordered symbols in dictionaries or elements of an ordered set.
* str1.compareTo(str2) → 'A' ≠ 'a'

  ↓ so, we use

  0 : equal
  < 0 : -ve ⇒ str1 < str2
  > 0 : +ve ⇒ str1 > str2.
* Compare To Ignore Case

  ↓

  'A' = 'a'

⇒
```
public static void main (String arg[]) {
    String fruits [] = {"apple", "mango", "banana"};

    String largest = fruits [0];
    for (int i=1; i < fruits.length; i++) {
        if (largest.compareTo (fruits [i]) <0) {
            largest = fruits [i];
        }
    }
    sysо (largest);
}
```

* Output: mango

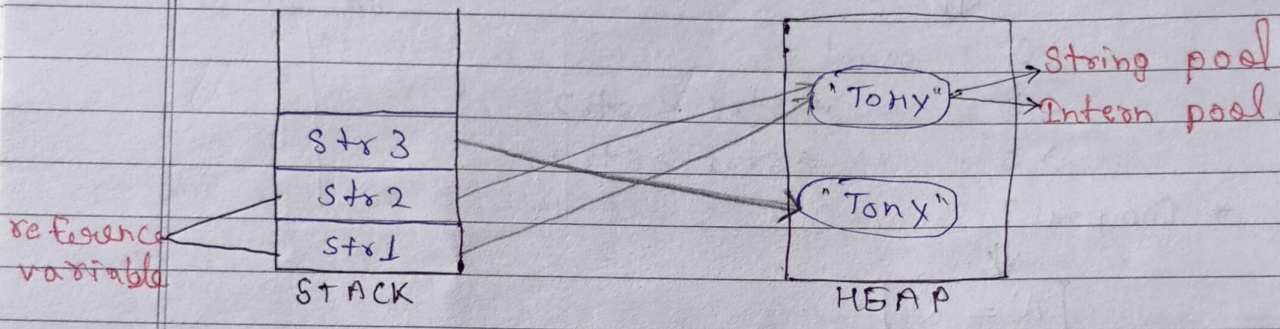
// Time Complexity, ⇒ O (n x m)

* WHY STRING ARE IMMUTABLE?

same → String str1 = "Tony";
String str2 = "Tony";
not same → String str3 = new String ("Tony");

concept → Interning

There two memory that are :-



* This concept is known as INTERNING.

* STRING BUILDER

- variable_name.toString() → it can change any object or datatype to string.
- StringBuilder variable_name = new StringBuilder (" "); it's a class.
- variable_name.append() → एक के एक फिर जुड़ते चले जाना

⇒
```
public static void main (String args[]) {
    StringBuilder sb = new StringBuilder ("");
    for (int char ch = 'a'; ch <= 'z'; ch++) {
        sb.append (ch);
    } //O(26)
    System.out.println (sb);
    System.out.println (sb.length());
}
```
अगर 2 से Empty नहीं होता तो // $O(n^2)$.

- We write all string program into StringBuilder becoz of memory efficiency & easy to use.

Ques. For a given String convert each the first letter of each word to uppercase.

"hi, i am monkey d luffy"

⇓

"Hi, I Am Monkey D Luffy."

- We will use an in-built function, character.toUpperCase();

⇒
```
public static String toUpperCase (String str) { //O(n)
    StringBuilder sb = new StringBuilder ("");
    char ch = Character.toUpperCase (str.charAt(0));
    sb.append (ch);
    for (int i=1; i<str.length(); i++) {
        if (str.charAt(i) == ' ' && i< str.length()-1) {
            sb.append (str.charAt(i));
            i++;
            sb.append (Character.toUpperCase (str.charAt(i));
        } else {
            sb.append (str.charAt(i));
        }
    return sb.toString(); }.
```

* STRING COMPRESSION

If any character is repeating again & again then compress it to small.

Ex: "aaabbccccdd" ⟹ "a3b2c4d2"

But, "abc" ⟹ "a1b1c1" ✗ Decompression.

it's called Decompress.

// Time Complexity, O(n)

⟹
```
public static String compress (String str) {

    String newStr = " ";
    for (int i=0; i < str.length(); i++) {
        Integer count = 1;
        while (i < str.length()-1 && str.charAt(i)
                == str.charAt(i+1)){

            count++;
            i++;
        }
        newStr += str.charAt(i);
        if (count > 1) {
            newStr += count.toString();
        }
    }
    return newStr;
}
```

Q2.
```
public class Solution {
    public static void main (String args[]) {
        String str = "Shradha Didi";
        String str1 = "ApnaCollege";
        String str2 = "Shradha Didi";
        Syso (str.equals (str1) + " " + str.equals (str2));
    }
}
```

false true

Q3.
```
public class solution {
    public static void main (String args[]) {
        String str = "ApnaCollege".replace ("l", " ");
        Syso (str);
    }
}
```

* it replaces all the "l" letter with a space " ".

⇩

ApnaCo ege

Q4. Count how many times lowercase vowels occurred in a String entered by the user.

```
public static void vowels_checking (String str) {
    int count = 0;
    for (int i=0; i < str.length(); i++) {
        char ch = str.charAt (i);
        if (ch == 'a' || ch == 'e' || ch == 'i' || ch == 'o' ||
            ch == 'u') {
            count ++;
        }
    }
    Syso ("Count of vowels are: " + count);
}
```

Q 5. Determine if 2 Strings are anagrams of eachother.
Anagrams : If two strings contain the same characters but in a different order, they can be said to be anagrams. Ex: race & care.

```
public static void main (String[] args) {
    Scanner sc = new Scanner (System.in);
    Syso ("Enter Ist String: ");
    String str1 = sc.nextline();
    Syso ("Enter IInd String: ");
    String str2 = sc.nextline();
//convert to lowercase that we don't have to check separately.
    str1 = str1.toLowerCase();
    str2 = str2.toLowerCase();
//check if the length are same or not
    if (str1.length() == str2.length()) {
      //convert strings to char Array.
        char[] str1charArray = str1.toCharArray();
        char[] str2charArray = str2.toCharArray();
      //sort the char Array
        Arrays.sort (str1charArray);
        Arrays.sort (str2charArray);
//if sorted char array are same or identical then they are anagram
        boolean result = Arrays.equals (str1charArray, str2charArray)
        if (result) {
            Syso (str1 + "&" + str2 + " are anagrams.");
        }
    } else {
        Syso (str1 + "&" + str2 + " are not anagrams.");
    }
}
```