javapoint.com

Java String

29 SATURDAY
OCTOBER
303-063 • WK 44

string → literal meaning
"seq of chars"

In Java, string is basically an object that represents sequence of characters.

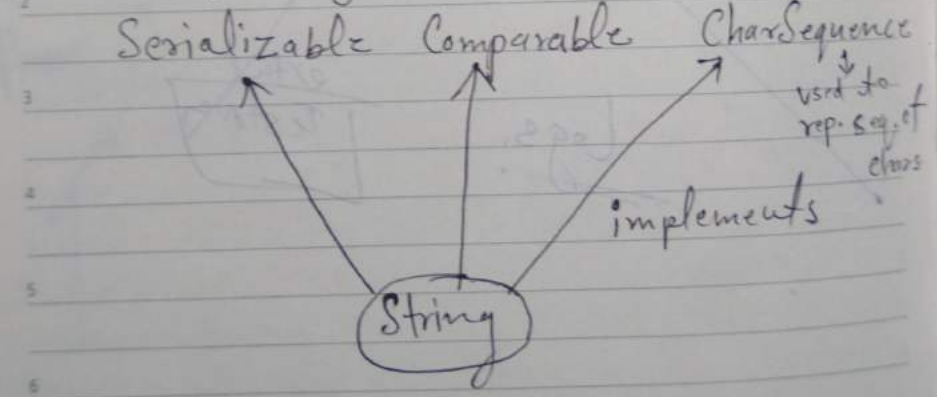
```
char[] ch = {'j', 'a', 'v', 'a'};
String s = new String(ch);
```

⇓ is same as

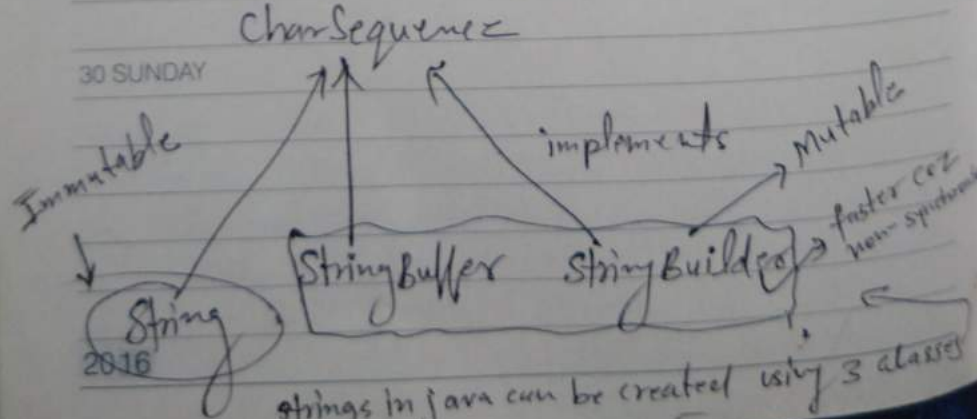
```
String s = "java";
```

x = 4
↓
x is a name given to the memory loc^n where 4 is stored

java.lang.String implements 3 interfaces



30 SUNDAY



2016

strings in java can be created using 3 classes

All wrapper Classes are Immutable

MONDAY OCTOBER 31

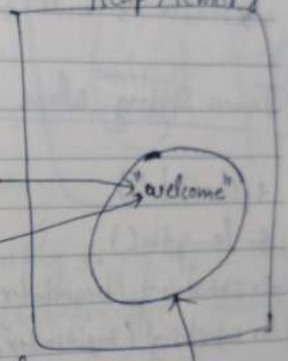
Immutable String → Unmodifiable or Unchangeable

Its data or state can not be changed, if we try to change it a new instance is created.

→ There are two way to create String Object.

1. By String Literal → created using dobble quotes " "
2. By new Keyword

① String s1 = "welcome";
String s2 = "welcome";



same object created only once.

whenever you create a string literal JVM checks its presence in SCPool.

i) if string dzn't exist in the pool, a new string obj. is created & placed in the pool.

ii) if string already exist in the pool, a reference of the pooled instance is returned.

Why java uses concept of string literals?
To make java more memory efficient. (coz no new obj is created if it exists already in string constant pool)

String Constant Pool → a special memory area in heap which stores string objects.

Why String Pool → increases reusability (of existing string obj)

synchronised } ⇒ Applicable for only Mutable objects
non-synchronised } not for Immutable.

01 TUESDAY NOVEMBER
306-060 • WK 45

(2) String s = new String ("welcome");
// creates two object & one reference variable

By JVM →
- a new obj. in (non-pool) heap memory (welcome)
- literal "welcome" will be placed in SC pool.

variable s will refer to the object in heap.

Java String class methods

s.charAt(i)
s.length()
s.substring(beginindex) → inclusive
s.substring(beginindex, endindex) → exclusive
T/F s.contains("abc")
T/F s.equals(s2) / equalsIgnoreCase(s2) → can be used for SB, SB
T/F s.isEmpty() → returns true only if length is 0
s.concat(s2) → char, charSequence, "abc", "xyz"
s.replace(old, new) → replace all occurrences
s.equalsIgnoreCase(s2)
s.split(regex) → string → returns array of splitted strings
s.split(regex, int limit) → limit for no. of string in array
s.intern()
s.indexOf(ch) → char, substring → int (index of 1st occurrance)
s.indexOf(ch, fromIndex)
s.toLowerCase()
s.toUpperCase()
s.trim() → removes spaces from beginning & ending of string
s.valueOf(123) → int → string (static String) — can be any type: int, float, ... char array etc.

as per matching regex:
- 1 → only one string
- 2 → splitted in 2 string
- 0 → All strings

s.valueOf(true), s.valueOf('c'), s.valueOf(6.6); double, Object

s.replaceAll(regex, new) ⇒ only for String, not for SB
s.replaceFirst(regex, new) → only 1st SB

2016

s.startsWith("ab");
s.endsWith("e");
s.contains("achi");
} T/F

s.matches(String regex) T/F
whether string matches given regex
regular exp

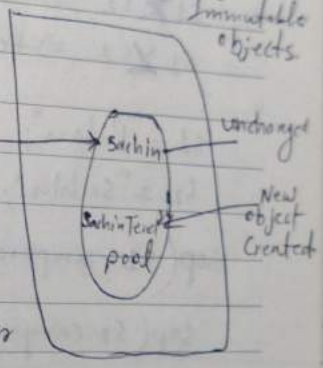
s1.compareTo(s2)
s1.compareToIgnoreCase(s2)

WEDNESDAY NOVEMBER 02
WK 45 • 307-059

String s = "sachin";
char ch[] = s.toCharArray();

s.concat("Tendulkar");
sop(s); // "Sachin" is printed coz strings are Immutable objects

s reference var still pts to "sachin"



s = s.concat("Tendulkar");
sop(s); // Sachin Tendulkar

// if assigned explicitly to ref var s then it will refer to SachinTendulkar

Java String Compare:

Java string can be compared on the basis of content & reference.

Uses:
- Content Authentication → by equals()
- Sorting → by compareTo()
- reference Matching → by == operator

3 ways to compare

String s1 = "sachin";
String s2 = "sachin";
String s3 = new String("sachin");

s1.equals(s2) → true
s1.equals(s3) → true

sop(s1==s2) → true
s1==s3 → false

2016

cap & L ASCI value less
A (65) a (97)

A → 1
B → 2
C → 3
D → 4
Z → 26

03 THURSDAY NOVEMBER
308-058 • WK 45

s1 = "Ram"; s2 = "ram";
s1.compareTo(s2) => -17
" Ignore case => 0

compareTo(): compares content lexicographically
returns an integer value → Dictionary order

s1 == s2 => 0 s1 precedes s2 in dict.
s1 > s2 => -ve value => s1 comes first in dictionary order as compared to s2
s1 < s2 => +ve value => s2 precedes s1

Left side minus Right side

s1 = "Ratan";
s2 = "sachin";

sop(s1.compareTo(s2)); => -40 ← random integer.

sop(s2.compareTo(s1)); => +39

Concatenation → By '+' opr
→ By concat() method

String s = "sachin" + "Tendulkar";
sop(s); ↑ Sachin Tendulkar

String s = 50+30+"sachin"+40+40;
sop(s); => 80 sachin4040

Note: After a string literal all + will be treated as a string concatn opr.

String s1 = "sachin";
String s2 = "Tendr";
String s3 = s1.concat(s2);
sop(s3); Sachin Tendr

2016

FRIDAY NOVEMBER 04

Substring → a part of string

```
String s = "Sachin Tendulkar";
           012345 6
sop(s.substring(6)); ⇒ Tendulkar
                ↳ begin index
sop(s.substring(0,6); ⇒ Sachin
```

startsWith() & endsWith()

```
sop(s.startsWith("sa")); → true
sop(s.endsWith("h")); → true.
```

StringBuffer → Synchronised

→ same as string but mutable

Impot constructor → StringBuilder() ⇒ same 3 consts in StringBuilder

- StringBuffer() → creates an empty string buffer with initial capacity of 16
- StringBuffer(String s) → creates SB with specified String
- StringBuffer(int capacity) → creates empty string buffer with specified capacity as length.

Impot methods: → like concat in string: → same methods in StringBuilder.

s.append(s2)	length()
insert	charAt()
delete	substring()
replace	
reverse	
capacity	

can be any data type (int, boolean, float, string)
But string rep^n of that type is appended.
concat(+) only string

2016

"String" → Immutable : so there is no sense of (synchronised or non-synchronised)
↳ coz it can not be modified by single thread or multiple thread.

05 SATURDAY NOVEMBER 310-056 • WK 45

StringBuffer s = new StringBuffer("Hello");

s.append("java");

sop(s); → Hello java

StringBuilder → same as SB class but it is non-synchronised.

StringBuilder s = new StringBuilder("Hello");

s.insert(1, "java"); => Hjavaello

or

s.delete(1,3); => Hlo

or

s.reverse() => olleH

sop(s.capacity()); => 16

if no of char increases from its current capacity. cap. is increased by (old cap. x2)+2

06 SUNDAY

if s.append("Deepak"); (is my favorite buddy in school)

sop(s.capacity()) => (16 x 2) + 2 => 34

=> befor append sop(s.hashCode());

After append sop(s.hashcode());

Both will be same in SB => efficient

2016 But in case string it would be diff

StringBuffer is Synchronised, StringBuilder is not which makes StringBuilder faster than StringBuffer.

String	StringBuffer	StringBuilder
Immutable	Mutable	Mutable
Thread-Safe	Thread-Safe	not thread-safe
slow & consumes more memory when concat of too many strings coz every time a new instance is created.	fast & consumes less Memory	Fastest
2 way to create string	By new op only	By new op only
Neither synchro nor non-synchro	Synchronised i.e. thread-safe i.e. two threads can't call the methods of StringBuffer simultaneously. less efficient	non-Synch more eff

```
String s1 = "Java";
String s2 = s1.concat("J2EE");
sop(s1==s2);
        ↓ false
```

+ op is used only for strings to concat

s2 = s1 + "J2EE"; ✓

```
StringBuffer s1 = new SB("Java");
SB s2 = s1.append("J2EE");
sop(s1==s2);
        ↓
       true
```

→ Not for SB, SB has its own method concat

s2 = s1 + "J2EE"; → compile time error "cannot convert String to StringBuffer"

08 TUESDAY
NOVEMBER
313-053 • WK 46

→ All the wrapper classes are immutable. (so final class)
String, Boolean, Integer, Long, Float etc.

→ we can create an Immutable class by making it final class that hv final data.

toString() → returns the string rep^n of an obj.
To represent any object as a string.

if you print any object java compiler internally invokes tostring() method on the obj.

So overriding the toString() method returns the desired o/p.

Advantage: By overriding toString() of obj class we can return values of the obj; so we don't need to write much code.

2016

WEDNESDAY
NOVEMBER 09

Understand problem without toString();

```
class Student
{   int rollno;
    String name;
    String city;

Student(int rollno, String name, String city)
{   this.rollno = rollno
}
psvm(s.a)
{   Student s = new Student(101, "Raj", Lucknow);
    sop(s); // compiler writes here s.toString()
}
```

→ o/p: student@1fee6fc → hashcode values of obj (Not the value of obj) ⇓ so to bring override toString()

if added

```
public String toString()
{
  return rollno+" "+name+" "+city;
}
```

sop(s); ⇒ o/p: 101 Raj Lucknow

2016

10 THURSDAY NOVEMBER 315-051 • WK 46

→ Depricated Now.

StringTokenizer: → used to break the string into tokens.
(or on the basis of delimiter)

→ similar to split method but dzn't support regr exprn.

indexOf()

String s = "this is index of example";
0123

sop(s.indexOf("is")); // 2
"index" // 8

s.indexOf("is", 4) // 5
From 4th index (inclusive)

intern() → returns string from pool memory.
→ returns interned string

String s1 = new String("Hello");
String s2 = "hello";
String s3 = s1.intern(); // returns string from pool, now it will be same as s2

sop(s1==s2) // false
sop(s2==s3) // true.

isEmpty()

String s1 = "";
sop(s1.length()); // 0
sop(s1.isEmpty()); // true
s3.isEmpty() // false.

String s2 = null
sop(s2.length());
→ NullPointerExcep

s3 = "Hi";

[abc] → a, b, or c

[^abc] → Any char except a, b, c

\s → space

\d → digit (0-9)

\D → Non-digit [^0-9]

StringUtils & String both has join metho.

FRIDAY NOVEMBER 11

join() → returns a string joined with given delimiter.

```
String s = String.join("-", "welcome", "to", "java");
                       ↑ delimiter (Actually String)
Utils
sop(s); // welcome-to-java
```

list of String

any no. of

```
String s = String.join("-", array)
```

split() → returns array of strings as per given regular expression

```
String s = " I love my India";
String[] words = s.split("\\s"); // split based on whitespace
for (String w : words)
    sop(w);
```

I
love
my
India

or s.split("\\s", 0)

default value

I love my india if s.split("\\s", 1)

No pt to split

I
love my India } if s.split("\\s", 2)

valueOf()

```
int value = 30;
String s1 = String.valueOf(value);
sop(s1+10); // 3010
```

→ To split with space `s.split(" ");`

comma `s.split(",");`

dot `s.split("\\.");`

↓ \ is a reserved word/symbol in java regex.

So `s.split(".");` ✗

2016

7) [a-d[m-p]] → a thru d or m thru p; [a-dm-p] → union

12 SATURDAY NOVEMBER
317-049 • WK 46

Java Regular Expressions

↳ define pattern for searching or manipulating strings.

↳ widely used to define constrain on strings such as password email validations.

"java.util.regex" Pkg → provides 3 interfaces & 3 classes

Pattern.matches()

MatchResult → interface
Matcher
Pattern
PatternSyntaxException } class

(1) Regex char classes:

1) [abc] → a, b or c (any single char)
2) [^abc] → Any char except a, b, c (Any combination)
3) [a-zA-Z] → a through z or A through Z (inclusive)
4) [a-z&&[^bc]] → a through z except for b & c [subtraction]
[a-z&&[def]] → d, e or f (intersection)
5) [a-z&&[^m-p]] → a through z and not m through p.

(2) Regex Quantifiers → specifies occurances of a char.

Applied on char classes.

- X? X occurs once or not at all → only once
- X+ once or more times → atleast once.
- X* zero or more times
- X{n} n times only
- X{n,} n or more times
- X{y,z} at least y times but less than z times

2016

[amn] → String can hv only char a or m or n

↳ [amn]{1}

"a" ✓

"abcd" ✗

"ammna" ✗

Length By default 1

MONDAY NOVEMBER 14

WK 47 • 319-047

Regex Metachar → works as short codes

or predefined char Classes:

dot → Any single char.

\d → any digits [0-9]

\D → Any non-digit, short for [^0-9]

\S → Any whitespace char, short for [\t\n\x0B\f\r]

\s → Any non-whitespace char

\w → A word character [a-zA-Z_0-9]

\W → A non-word char.

Sop(Pattern.matches("\\d", "abc")) → false

"1" → true

Regex, charSequence

Sop(Pattern.matches("[amn]", "a")) → true

"aaa" → false

"am" → false

"aau" → true

"aazzta" → false

2, if not matching pattern.

• (dot) represents Any single char. "[amn]+", "aau")

Pattern.matches(".s", "as"); // true

2nd char should be "s"

only two char

"mk" → f

mst → f ⇒ 2 ≠ 2

("..s", "mas") → true

Mobile No. format valid?

Sop(Pattern.matches("[789][1][0-9]{9}", "9867774816")); → true.

↳ Total 10 char.

starting with 7, or 8 or 9 only.

Sop(Pattern.matches("[a-zA-Z0-9]{6}", "arun32")) → T

aun12 → F

*, $

2016

psvm main (String[] args)

sop();

↓

Sysout ⇒ Ctrl+space

Javac filename.java

java filename

↓ passed as command line argument

String s = "Sachin";

String s2 = " sachin ";

Used in Assertion

1. sop(s.toUpperCase()); // SACHIN
2. sop(s.toLowerCase()); // sachin
3. sop(s.trim()); // sachin // Remove the added spaces at starting & ending
4. sop(s.startsWith("Sa")); // true
   ma ⟶ false
5. sop(s.endsWith("in"); // true
6. sop(s.charAt(1)); // a
7. sop(s.length()); // 6
8. sop(s.replace("ach", "ich"); // Sachin

15 TUESDAY NOVEMBER
320-046 • WK 47

String Interview Q

store words as key & its occurrence as value in Hashmap.

1) WAP to find duplicate words & their no. of occurences in a string

```
psvm(s.....)
{
    String s = "Super Man Bat man. spider man";
    String [] words = s.split(" ");
                      s.toLowerCase().split(" ");
    HashMap<String, Integer> hm = new HM<S,I>();
    for (String w : words)
    {
        if (hm.containsKey(w.toLowerCase()))
            hm.put(w.toLowercase(), hm.get(w.tLcase())+1);
        else
            hm.put(w.tolowercase(), 1); // 1st time inserted.
    }
Set<String> keys = hm.keySet();
    for (String K : Kegs)
    {
        if (hm.get(K) > 1)
            sop(K + " : " + hm.get(K));
    }
}
```

o/p man : 3

2016

hm. containsKey (Key) → T/F
• get (Key) → value of key
• KeySet () → set of strings

WEDNESDAY NOVEMBER 16

2) WAP to count the no of words in a string?

String s1 = " I love my India"

String [] s = s1.trim().split(" ");

sop (s.length);

3) WAP to count the total no. of occurrences of a given char in a string (without using any loop)

String s = "Java is java again";

Char ch = 'a';

int count = s.length() - s.replace('a', '').length();

sop(" No. of occurence of a is: " + count);

4) To Reverse a string

(A)

StringBuffer sb = new SB(" My Java");

sop(sb.reverse());

(B)

String s = "My Java"

char [] ch = s.toCharArray();

for (int i = ch.length-1; i>=0; i--)

soprint(ch[i]);

for (int i = s.length-1; i>=0; i--)

result += s.charAt(i)

sop(result);

String result = "";

17 THURSDAY NOVEMBER
322-044 • WK 47

hm.containsKey(c)
hm.get(c)
⇒ or duplicate char.

5) WAP to count occurences of each char in a string

```
String s = "super man bat man spider man";
char [] ch = s.toCharArray();
HashMap<Character, Integer> hm = new HM<>();
for (char c : ch)
{
```

store each char as key & its occurence as value in a Hashmap

```
if (hm.containsKey(c))
    hm.put(c, hm.get(c)+1);
else
    hm.put(c, 1);
}
sop(hm);
```

o/p: { s=2, a=4, r=2, u=1 .... }

⇒ Above program is case sensitive, it treats 'A' & 'a' as two diff char.

⇒ better convert input string to LC. or UCase

```
char [] ch = s.toLowerCase().toCharArray();
```

7) // In above to find duplicate char (occurance > 1)

```
Set<Character> keys = hm.keySet();
for (Character c : keys)
    if (hm.get(c) > 1) → sop(c + ": " + hm.get(c));
```

FRIDAY NOVEMBER 18

WK 47 • 322 (43)

replace() vs replaceAll()

→ uses regex

6) To remove all spaces from a string.

```
String st = s.replaceAll("\\s", "");
        or
if (ch[i] != ' ' && ch[i] != 'H')
    {
        sb.append(ch[i])
    }
```

8) Check whether one string is rotation of Another

S1 = "ABCD"

rotated versions

S2 = BCDA, CDAB, DABC, BACD ✗

```
if (s1.length() != s2.length())
    sop ("s2 is Not rotated version of s1");
else
{
    String s3 = s1 + s1;
    if (s3.contains(s2))
        sop ("rotated version");
    else
        sop ("Not");
}
```

2016

S:2
9:4

19 SATURDAY
NOVEMBER
324-042 • WK 47

→ length of both string should be equal
→ same char

9) To check wheter two strings are anagram.

→ Two strings are called anagrams if they contain same set of char, but in different order.

e.g Keep - peeK
SchoolMaster - the classroom
<space>
Dormitory - DirtyRoom
God → Dog ✓
God → Dogg ✗

```
String s1 = "Mother in Law";
String s2 = "Hitler Woman";
char[] ch1 = s1.replaceAll("\\s", "").toLowerCase()
                .toCharArray();
Char[] ch2 = s2.replaceAll("\\s", "").toLowerCase().toCharArray();
if (ch1.length != ch2.length)
    sop("strings are Not anagrams");
else{
    Arrays.sort(ch1);
    Arrays.sort(ch2);
    if (Arrays.equals(ch1, ch2))
        sop("strings are Anagrams");
    else
        sop("Not anagrams");
}
```

20 SUNDAY

2016

MONDAY
NOVEMBER 21

10) Reverse string with preserving the position of spaces

```
String s = "I Am Not String";

Char [] ch = s.toCharArray();
char[] result = new char[ch.length];

for(int i = 0; i < ch.length; i++
  { if (ch[i] == ' ')
        result[i] = ' ';
  }
```

→ split with space
→ reverse each string
→ concat with spaces

⇒ copy space characters

```
  int j = result.length - 1;

for(int i = 0; i < ch.length; i++)
  {
    if (ch[i] != ' ')
    {
      if (result[j] == ' ')
        { j--;
        }
      result[j] = ch[i];
      j--;
  } }

sop( ch + " → " + String.valueOf(result));
```

O/P: I Am Not String → I ma toN gnirtS

2016

22 TUESDAY NOVEMBER
327-039 • WK 48

11) Reverse each word of string

→ Split with space
→ reverse string
→ concat all string

```
String s = "what a bad day";

String[] words = s.split(" "); or s.split("\\s");

    String result = "";

for (String w : words)
  {
      String temp = "";

      for(int i = w.length()-1; i>=0; i--)
        { temp = temp + w.charAt[i];
        }

      result = result + temp + " ";
  }

sop( result.trim());

o/p: tahw a dab yad
```

12) String → Integer Conversion

```
String s = "2015";

int i = Integer.parseInt(s); // returns primitive int

int i = Integer.valueOf(s); // returns object Integer
```

↓ Autoboxing

obj. Integer is stored here in int primitive

WEDNESDAY
NOVEMBER
23

→ Integer to String

```
int i = 2015;
String s = Integer.toString(i);
String s = String.valueOf(i);
```

13) Swap two string variable without using 3rd variable

```
String s1 = "xyz";
        s2 = "ABC";
s1 = s1 + s2;
s2 = s1.substring(0, s1.length()-s2.length());
s1 = s1.substring(s2.length());
```

Sop(s1) → ABC
Sop(s2) → xyz

2016

a character → can be a { Letter, digit, symbol } → *, space, @, $ etc

24 THURSDAY NOVEMBER
329-037 • WK 48

14) To find %age of Uppercase letters, Lowercase letters, Digits & special characters in string?

"java.lang.Character" class

3 impot methods:

Character.isUpperCase(ch)
" .isLowerCase(ch)
" .isDigit(ch) → 0-9
.isLetter(ch) → a-z
.isSpace(ch) ⇒ space
.isLetterOrDigit(ch) ⇒ [0-9] & [a-z]

String s = "Tiger Runs @ the Speed Of 100 Km/hr";

```
int totalchar = s.length();
int uc=0;
 " lc=0;
 " digits=0;
 " others=0;

for(int i=0; i<s.length(); i++)
{
    char ch = s.charAt(i);
if(Character.isUpperCase(ch))
        uc++;
else if(Character.isLowerCase(ch))
        lc++;
else if(Character.isDigit(ch))
        digits++;
else
        others++;
```

FRIDAY NOVEMBER 25

```
double UCP = (UC * 100.0) / totalchar;
"   LCP = (LC * 100.0) / totalchar;
"   digitsP = (digits * 100.0) / totalchar;
"   othersP = (others * 100.0) / totalchar;

DecimalFormat formatter = new DecimalFormat("##.##");

sop("UC are :" + formatter.format(UCP) + "%");
```

LC ——— LCP ———
digits ——— digitsP ———
others ——— othersP ———

O/P: UC are : 13.16%
LC ——— 52.63%
Digits ——→ 7.89%
other char ——→ 26.32%

Sort a given string: O/P: -✓-✓-✓-f-L-T-e-f-h-n-o-t

```
String s = "L & T Infotech"
char ch[] = s.toCharArray();
// Arrays.sort(ch);
char temp;
for (int i=0; i<s.length(); i++)
{
    for(int j=i+1; j<s.length(); j++)
        if (ch[i] > ch[j])
        {
            temp = ch[i];
            ch[i] = ch[j];
            ch[j] = temp;
        }
}
```

or

```
for(char c: ch)
    sop("-"+ch);
```

↓ print.

2016

26 SATURDAY
NOVEMBER
331-035 • WK 48

Unicode values of character:

A → 65 a → 97
B → 66 b → 98
C → 67 C → 99
space → 32
@ → 64

To print

```
sop("A→" + (int)'A');
sop("space" + (int)' ');
sop("@:" + (int)'@');
```

Print the numbers, letters & symbols (special char) separately from a string.

```
String s = "Hi Subramanian @ 123 Good Morning 456 $";
char[] a = s.toCharArray();
String result = "";
for (int i=0; i<a.length; i++)
{ if (Character.isDigit(a[i]))
  {
    result = result + a[i];
  }
}
sop("Numbers are : " + result);
```

space comes under special char

char

isLetter(a[i])

⇒ 1234567

2016

!(Character.isLetterOrDigit(a[i])) ⇒ Hi Subramanian Good Morning

⇒ space sp @ sp sp sp sp sp $

MONDAY
NOVEMBER 28
WK 49 • 333 033

using Reg^x exp^n: String a[] = s.split(" ");

if (Pattern.matches("[0-9]*", a[i]) → string (actually CharSequence)

↓ To get words containing Numbers only.

[a-zA-Z]* → Letters only.

s = Hi 9 am 123Deepak s = Hi;9; Deepak

s.split("[0-9]"); s.split(";");

String s = abc.com/ebm-mobil/anp

String part[] = s.split(".com");

part[1] → /ebm-mobile/anp

part[0] → abc

Java Regex :...

29 TUESDAY NOVEMBER
334-032 • WK 49

works only for a single char
Here by default, by default Quantifier is [1]

Lowercase alphabetic char — [a-z]
Uppercase — [A-Z]

Alphabetic char — [a-zA-Z]
Digit — [0-9]
Alphanumeric [a-zA-Z0-9]

punctuation char ⇒ \p{Punct}

e.g. ! " # $ % & ' ( ) * + , - . / : ; < = > ? @ [\ ] ^ _ ` { | } ~ ⇒ 32

escape char \ req here. punctuation characters same on K.B.
quantifier

s.matches("\\p{Punct}*");

TIF → if given string contains punctuation symbol

Boundary Matchers:

^ → beginning of a line
$ → the end of a line

precedence of char-class operator (Highest to lowest)
(char classes may appear within other char classes - composed)

1. Literal escape \x
2. Grouping [...]
3. Range a-z → present atleast in one class.
4. Union [a-e][i-u]
5. Intersection [a-z && [aeiou]]

* denotes a class that contains every char that is in both of its operand classes

2016