**Strings**

**Day-01**

**27-03-2025**

**==================**

How Strings are important?

-------------------------------

Scanner class:

----------------

-> dynamic variable

Ex: i want to take an integer:

import java.util.Scanner;

Scanner scan = new Scanner(System.in);

Sop("Enter the value for a:");

int a = scan.nextInt();

nextLine() + parseInt()

S.o.pln("Enter the value for a:");

String a = scan.nextLine();

int a = Integer.parseInt();

-> While defining the dynamic variable, the object of the scanner class can take the value as String by default.

-> Before going to store in the heap memory, that string value internally can convert into required primitive type. This process is called as "Wrapping".

wrapping:

------------

converting a string into a primitive datatype is called as "wrapping".

-> to perform the wrapping, we have wrapper classes.

-> for each primitive datatype, there is one wrapper class:

int ==> Integer

float ==> Float

byte ==> Byte

short ==> Short

long ==> Long

double ==> Double

boolean ==> Boolean

char ==> Char

Type conversion:

-------------------

-> also called as "Type casting".

-> the data from one primitive type can be converted into another primitive type is called as "Type conversion".

Boxing:

---------

-> The data from primitive type to reference type can convert means this is called as "Boxing".

Unboxing:

-----------

When the reference data can convert into primitive type is called as "Unboxing".

Note:

----

Java has automatic boxing and unboxing technique.

So, no need to implement boxing and unboxing explicitly.

**String Definition?**

**---------------------**

String is a collection of characters which must be enclosed with double quotes.

Ex: "java", "java123", "123","java@123", "!\*#$" etc.

**How to create the string data?**

**------------------------------------**

JVM memory has two sections for handling/storing the string:

1) Heap memory

2) SCP (String Constant Pool)

When the string can store into the SCP?

and When the string can store into heap?

-> Strings can be defined in two forms:

1) String literal form

2) String Object form

-> When the string is defined in "string literal form", that value can store into SCP.

-> When we can define the string in the string object form", that can be stored into heap.

-> Java library is called as "Java API".

-> API ==> Application Program Interface.

-> To make execute the java application end to end successfully, the Java API can place the required programs from its packages as per the definition.

-> JAVA API is combination of packages:

1) Util Package

2) Lang package

3) IO package etc.

-> Java strings are pre-defined in Java.lang package.

-> String is one of the class from java.lang package.

-> the java.lang package is default in Java API so, no need to import the java.lang into our applications. It can be automatically happened by the JVM.

1) String Literal Form:

--------------------------

Syntax:

String variable-name = value-within-double-quotes;

2) String Object form:

-------------------------

Syntax:

String variable-name = new String("value");

**Internal working of Strings**

**==================**

Note:

=====

Equal operators:

----------------

== and !=

return: true/false

Ex: 10 == 10 ==> true

10 != 10 ==> false

when we can define the equal operators with primitive type of data, these can compare the values directly.

public class EqualOperatorWithPrimitives {

public static void main(String[] args) {

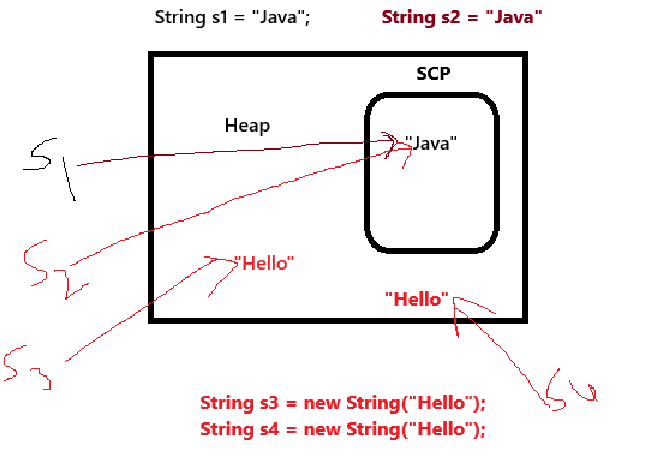
System.out.println(10 == 10);

System.out.println(10 != 10);

}

}

But, when we can define the equal operators on the strings, those are can compare the address locations of the strings.



Example-01:

--------------

public class EqualOperatorsOnStrings {

public static void main(String[] args) {

// String literal form

String s1 = "Java";

String s2 = "Java";

System.out.println(s1 == s2);

// String object form

String s3 = new String("Hello");

String s4 = new String("Hello");

System.out.println(s3 == s4);

}

}

Example-02:

---------------

public class EqualOperatorsOnStrings {

public static void main(String[] args) {

String s1 = "Hello";

String s2 = s1;

System.out.println(s1 == s2);

String s3 = new String("Hello");

String s4 = s3; // same address location

System.out.println(s3 == s4);

System.out.println(s1 == s3);

}

}

Note:

-----

-> The SCP cannot allow the duplication.

Ex:

String s1 = "Hello";

String s2 = "Hello"

-> In this case, s1 and s2 can point to the same memory location. Because SCP can store "Hello" for one time only.

-> The Heap memory does allow the duplication.

Ex:

String s1 = new String("Hello");

String s2 = new String("hello");

-> heap memory can reserve two memory blocks for the same data. One is pointed by "s1" and another is pointed by "s2".

**Day-02**

**28-03-2025**

**=======================**

**Q: When we are creating the dynamic memory for variable,**

**why we can use the syntax like below:**

**double d = scan.nextDouble()**

**int a = scan.nextInt()**

**?**

Ans:

----

Scanner is not able to read all the data

it can able to read any data in the string format.

-> to make convert the string formatted data into required (primitive type) format we can follow the above syntaxes.

scan.nextDouble()

String d = scan.nextLine();

double d = Double.parseDouble();

How to compare the string data?

-------------------------------

-> there are some pre-defined methods:

1) equals()

2) equalsIgnoreCase()

3) compareTo()

1) equals():

------------

-> when two string's data/content are same then, equals() can return "true" otherwise it can return "false".

Syntax:

string1.equals(string2);

Example program for equals():

----------------------------------

public class EqualsMethod {

public static void main(String[] args) {

String s1,s2;

s1 = "Hello";

s2 = "Java";

String s3,s4;

s3 = new String("Java");

s4 = new String("Hello");

System.out.println(s1 == s4);

System.out.println(s1.equals(s4));

System.out.println(s1.equals(s3));

}

}

**Q: What is the difference between '== operator' and equals() method:**

when we want to compare the two strings with the respect to the address, then we can use "==" operator.

When we want to compare two strings based on the data, we can use "equals()".

2) equalsIgnoreCase()

---------------------

Syntax:

string1.equalsIgnoreCase(string2);

-> this method is exactly same as equals() method

but this method can ignore the case of the text while the comparison.

Ex:

s1 = "abc";

s2 = "ABC"

s1.equals(s2) ==> false

s1.equalsIgnoreCase(s2) ==> true

Example Program for equalsIgnoreCase():

-----------------------------------------------

/\*

\* assume

\* username = "admin"

\* password = "admin123"

\*/

import java.util.Scanner;

import java.lang.String;

public class LoginSystem {

public static void main(String[] args) {

Scanner scan = new Scanner(System.in);

System.out.println("Enter the user name:");

String userName = scan.next();

System.out.println("Enter the password:");

String password = scan.next();

if(userName.equalsIgnoreCase("admin") && password.equalsIgnoreCase("admin123")) {

System.out.println("Login Success.");

}

else {

System.out.println("Login Fail.");

}

}

}

**Q: difference between next() and nextLine():**

**---------------------------------------------------**

next() method can read the data of the string until the white space only.

nextLine() can read the data of the string until the next line which including the white spaces also.

import java.util.Scanner;

public class NextVsNextLine {

public static void main(String[] args) {

Scanner s = new Scanner(System.in);

System.out.println("Enter user name:");

String user = s.next();

String user1 = s.nextLine();

System.out.println("Enter your location:");

String location = s.nextLine();

System.out.println("The user = "+user);

System.out.println("The user = "+user1);

System.out.println("The Location = "+location);

}

}

3) compareTo()

--------------

Syntax:

string1.compareTo(string2);

-> When two strings are with same Unicode values,

compareTo() can return '0'

-> When Unicode value string1 > Unicode value of string2,

compareTo() can return "+eve".

-> when the Unicode value of string1 is less than the string2,

compareTo() can return '-eve'

Note:

-----

compareTo() can work on individual characters from both the strings.

Example program for compareTo():

----------------------------------------

public class CompareToMethod {

public static void main(String[] args) {

String s1 = "Hello";

String s2 = new String("Hello");

System.out.println(s1.compareTo(s2));

String s3 = "java";

String s4 = "Java";

// lower case a to z ==> 97 to 122

// A to Z ==> 65 to 90

System.out.println(s3.compareTo(s4));

String s5 = new String("Python");

String s6 = new String("hello");

System.out.println(s5.compareTo(s6));

}

}

**How to access characters of strings:**

**------------------------------------**

charAt():

---------

-> this can be used to access the individual characters of the string based on the index value.

Syntax:

string-data.charAt(index);

Example program for charAt():

----------------------------------

public class StringAccessing {

public static void main(String[] args) {

String s1 = "Java";

// System.out.println(s1[0]);

System.out.println("The First Character = "+s1.charAt(0));

System.out.println("The Second Character = "+s1.charAt(1));

System.out.println("The Third Character = "+s1.charAt(2));

System.out.println("The Last charater = "+s1.charAt(3));

}

}

**How to traverse on the strings:**

**-------------------------------**

1) for loop

-----------

Example program for traversing using for loop:

----------------------------------------------------

public class StringAccessing {

public static void main(String[] args) {

String s1 = "Java";

for(int i = 0;i < s1.length();i++) {

System.out.println("The character at "+i+" is = "+s1.charAt(i));

}

}

}

2) for each loop

-----------------

-> for each loop on strings not applicable to use.

strictly for each loop can define with arrays only or other collections.

**How to acquire/access the part of the string?**

**---------------------------------------------**

substring():

------------

syntax-1:

string-data.substring(start-Index);

substring() can return new string from "start-index" to till the last of the given string.

Strings Vs Immutability:

------------------------

Once we have defined the string there is no possibility to modify within the same object location. Hence we can say the strings are "immutable".

Syntax-2:

---------

string-data.substring(begin, end);

Example Program for substring():

--------------------------------------

public class StringAccessing {

public static void main(String[] args) {

String s1 = "Java Programming Language";

System.out.println(s1);

String s2 = s1.substring(5);

// s1.substring(5);

System.out.println(s2);

String s3 = s1.substring(5,11);

System.out.println(s3);

}

}

Assignment:

----------

How to access the string characters using while loop.

**Day-03**

**29-03-2025**

**====================**

**String Concatenation:**

**===============**

-> Joining of two or more strings into one string is called as "String Concatenation".

-> String Concatenation ==> '+'

Syntax:

str1 + str2 + str3 + ...

Ex: "java" + "is" + "Object" + "Oriented" ==> "javaisObjectOriented"

Note:

-----

For string concatenation in java, there is at least one literal should be string.

Ex: "java" + 8 ==> "java8"

8 + "java" ==> "8java"

"8" + "java" ==> "8java"

public class StringConcatenation {

public static void main(String[] args) {

String s1 = "java";

int i1 = 8;

String r1 = s1 + i1;

System.out.println(r1);

System.out.println("java "+"Langauge");

System.out.println(i1 + s1 + i1 + 97+"Language");// 8java897

System.out.println(i1+i1+97+s1+79+" Language");

}

}

concat():

-----------

Syntax:

str1.concat(str2)

Note:

-----

While the string concatenation using concat() both inputs must be string type.

public class concatMethod {

public static void main(String[] args) {

String s1 = "Java";

String s2 = " Language";

Integer i1 = 9977;

String s3 = s1.concat(s2);

System.out.println(s3);

// String s4 = s1.concat(i1);

}

}

**String Replacement:**

**==============**

"Java is difficult"

replace "difficult" with "easy"

-> for this, we have methods:

1) replace()

2) replaceFirst()

3) replaceAll()

1) replace():

-------------

Syntax:

str-data.replace("old-sub-string", "new-sub-string");

2) replaceFirst():

------------------

When we want to replace the sub-string in the first occurrence of the string, we can use "replaceFirst()".

Syntax:

string-data.replaceFirst(old, new);

3) replaceAll():

---------------

Syntax:

string-data.replaceAll(old, new);

public class StringReplacement {

public static void main(String[] args) {

String s1 = "The cat sat on the wall.";

System.out.println("The String before replace operation = "+s1);

s1 = s1.replace("cat", "dog");

System.out.println("The String after replace operation = "+s1);

String s2 = "The cat sat on the wall.And from the wall the cat jump into another wall.";

s2 = s2.replaceFirst("cat", "dog");

System.out.println(s2);

String s3 = s2.replaceAll("cat", "dog");

System.out.println(s3);

}

}

Q: How to convert the string into character array?

--------------------------------------------------

s1 = "Java";

char[] ch = new char[s1.length()];

for(i = 0;i < s1.length();i++)

{

ch[i] = s[i];

}

toCharArray():

---------------

-> when we want to convert a string into the character array, we can use "toCharArray()".

Syntax:

char[] ch = string-data.toCharArray();

public class StringToCharacterArray {

public static void main(String[] args) {

String s1 = "Java Programs";

char[] ch = s1.toCharArray();

for(char c:ch) {

System.out.print(c+"\t");

}

System.out.println();

}

}

Q: How to convert the string into an array?

-------------------------------------------

split():

--------

Syntax:

string-data.split("separator")

import java.util.Arrays;

public class StringToArray {

public static void main(String[] args) {

String s1 = "29-03-2025";

String[] s2 = s1.split("-");

System.out.println("The Array = "+Arrays.toString(s2));

String s3 = "Java is Object Oriented Langauge";

String[] s4 = s3.split(" ");

System.out.println(Arrays.toString(s4));

}

}

Q: How to convert the array into the string?

--------------------------------------------

join():

-------

-> Static method.

Syntax:

String str = String.join("separator", Array-data);

public class ArrayToString {

public static void main(String[] args) {

String[] s1 = {"29","03","2025"};

String s2 = String.join("-", s1);

String s3 = String.join(".", s1);

String s4 = String.join("/", s1);

System.out.println(s2);

System.out.println(s3);

System.out.println(s4);

}

}

**Day-04**

**01-04-2025**

**============================**

**How to find the sub-strings in a given string?**

**============================**

"Cat sat on the wall. Cat jump to another wall"

substring(1) ==> substring from index-1 to till last

substring(1,10) ==> from index-1 to index-9 all characters

"cat" ==> index - 0

-> two string methods:

1) indexOf()

2) lastIndexOf()

Note:

-----

That every method naming convention must be with camel case representation.

Camel case ==> from the second word that first letter must be capital.

substring:

---------

the part of string with single or more than one character from the string.

Ex: "java"

"j", "ja", "jav" etc.

indexOf():

----------

-> can accept a sub-string.

-> and can return the first occurrence (index) of the given sub-string.

Ex: str = "Java"

1 and 3

first index ==> 1

indexOf() ==> 1

Syntax:

string-data/object.indexOf("sub-string");

lastIndexOf():

--------------

-> same as indexOf()

-> by accepting the substring the lastIndexOf() can return the last occurrence of the given substring.

Syntax:

string-data.lastIndexOf("sub-string");

Note:

-----

if the sub-string is not present in the given, indexOf() and lastIndexOf() can return '-1'.

public class FindingSubstrings {

public static void main(String[] args) {

String s1 = "Java is High-level and Java is Object Oriented programming Language";

int i1 = s1.indexOf('a');

int i2 = s1.indexOf("Java");

System.out.println("The First Occurrence = "+i1);

System.out.println("The First Occurrence = "+i2);

int i3 = s1.lastIndexOf('a');

int i4 = s1.lastIndexOf("Java");

System.out.println("The Last Occurrence = "+i3);

System.out.println("The Last Occurrence = "+i4);

int i5 = s1.indexOf("Python");

int i6 = s1.lastIndexOf("C++");

System.out.print(i5+" "+i6);

}

}

**How to remove leading spaces from the string?**

**==============================**

leading spaces ==> spaces at beginning and/or at ending

Ex: "java programs"

when the string with the spaces in between characters we can't remove.

Only we can allowed to remove the leading spaces only.

trim():

-------

method can remove the leading spaces.

Syntax:

string-data.trim()

public class RemovalOfLeadingSpaces {

public static void main(String[] args) {

String s1 = "Java Programs";

String s2 = " Java";

String s3 = "Java ";

String s4 = " Java ";

String s5 = " Java Programs ";

System.out.println("Before the trim operation = ");

System.out.println(s1);

System.out.println(s2);

System.out.println(s3);

System.out.println(s4);

System.out.println(s5);

s1 = s1.trim();

s2 = s2.trim();

s3 = s3.trim();

s4 = s4.trim();

s5 = s5.trim();

System.out.println("After the trim operation = ");

System.out.println(s1);

System.out.println(s2);

System.out.println(s3);

System.out.println(s4);

System.out.println(s5);

}

}

**Q: Write a java program to accept a string as an input. And find the count of vowels and consonants in a string.**

String s1 = "Java8 is Object Oriented";

1) before to implement the count for vowels and consonants, we should convert the total string into one case (lower case or to upper case)

toLowerCase():

--------------

-> can convert the string into lower case.

Syntax:

string-data.toLowerCase()

toUpperCase():

--------------

-> can convert the string into upper case.

Syntax:

string-data.toUpperCase()

2) Need to traverse on the string.

3) While traversing, we need to find only alphabets.

4) After the finding of alphabets, we should separate vowels and consonants

5) finally count number of vowels and consonants.

import java.util.Scanner;

public class CountVowelsAndConsonants {

public static void main(String[] args) {

Scanner scan = new Scanner(System.in);

System.out.println("Enter some string:");

String s1 = scan.nextLine();

s1 = s1.toLowerCase();

int vowels = 0,consonants = 0;

int index = 0;

while(index < s1.length()) {

char ch = s1.charAt(index);

if(ch >= 'a' && ch <= 'z') {

if(ch == 'a' || ch == 'e' || ch == 'i' || ch == 'o' || ch == 'u') {

vowels++;

}

else {

consonants++;

}

}

index++;

}

System.out.println("The number of vowels = "+vowels);

System.out.println("The number of consonants = "+consonants);

}

}

**How to find the Unicode values of the characters of the string?**

**======================================**

codePointAt():

--------------

-> can accept an index of the string

-> return a Unicode value of the specified index (character's).

Syntax:

string-data.codePointAt(index);

public class OtherMethods {

public static void main(String[] args) {

String s1 = "Java";

System.out.println(s1.codePointAt(0));

System.out.println(s1.codePointAt(1));

}

}

**String Validations:**

**--------------------**

startsWith():

-------------

Syntax:

boolean result = string-data.startsWith("sub-string");

endsWith():

------------

Syntax:

boolean result = string-data.endsWith("Sub-string");

Ex:

String url = "https://www.ashokit.in";

String mail = "ravivraoinfs@gmail.com";

Sop(url.startsWith("https://"));

Sop(mail.endsWith("@gmail.com"));

**Do we able to write java program without creation of java file?**

java 1.9 has introduced a feature "jshell".

using jshell, we can create java program without creation of java file.

How to use?

-----------

1) start command propmpt

and enter the below command:

$ jshell

-> to get any help:

$ /help

-> to exit from the jshell:

$ /exit

-> to list out all the practice on the jshell:

$ /list

-> /edit our practice:

$ /edit

Why jshell:

------------

1) to test the code faster.

2) no need to create any class and main()

3) for better practice with java