**STRINGS: refer “StringMethod.java”**

String Objects are created in two ways:

1. Using new Operator

2. Assignment

//first way

String firstName= **new** String("Raisha");

//first way

String lasttName;

lasttName= **new** String("Begum");

//third way

String address="Kodagu";

To understand better look at the below program:

Program:

public class Demo {

public static void main(String[] args) {

String s1 = **new** String("Java");

String s3 = **new** String("Java");

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

String s2 = "Java";

String s4 = "Lava";

System.***out***.println(s2 == s4);

}

}

**Note:**

The String Object which is created using assignment Operator is stored in **String constant pool**

The String Object which is created using new Keyword is stored in **non-constant pool**

**Note:**

In the above program, first s2 object is created in string constant pool, next s1 object is created in non-constant pool, and after that compiler executing s4 it sees content of s4 and s1 are same, so does not create new string in string constant pool rather it assign it to same object created using s2. So now **s2 and s4 are pointing towards same object** because content is same, this happens only to the strings in string constant pool

Now, when compiler executing s3,though the s1 and s3 content are equal, irrespective of the content it will create one more object and s3 is pointing towards new object, in non-constant pool irrespective of the content new objects are created every time when new keyword is used.

That is the reason why s2==s4 is true as they are pointing towards same object

That is the reason why s1==s3 is false ,as they are pointing towards different Object

**Note:**

String class Object created without new operator will be created in **String constant pool(This happens only for string class)**

If we have one more object with same value without new operator then the variable will point towards the already existed object(In constant pool you can have only constant objects).

**No duplicates are allowed in String constant pool.**

**Print statements**

1. **Print**
2. **Println**
3. **Printf**(print format specifier just like c and c++

%d – int

%f- float

%c for char

%s for string

**Int i=10;**

**Float f=6.345664**

System.out.println**(“ the value of** %d and %f are”, i,f);

Output: **the value of** i and f are”, 10,6.345664);

System.out.println**(“ the value of** %d and %.2f are”, i,f);

Note: .2 mean print upto 2 decimal value

Output: **the value of** i and f are, 10,6.34

System.out.println**(“ the value of** %d and %8.2f are”, i,f);

Note: 8.2 mean print upto 8 spaces including value and .2 upto 2 decimal value

Leave some spaces print value like “ 6.34”- total 8charcters including spoaces

Output: **the value of** i and f are, 10,6.34

1. **Format – same as format specifier**

**Methods():**

**length()**:this method counts the length of the string and returns int value[Ex: s1.length()]

**concat(String)**:this method adds String to the already existing string assume[Ex: String s1="hello"; now Sop(s1.concat("developer")); then o/p is hello developer

**isEmpty()**:this method returns boolean, it will check whether string is empty or full and returns true or false[Ex: String s1="hello"; now SOP(s1.isEmpty()); as s1 is not free, the output will false]

**contains(String)**:this method returns boolean and checks whether the given string contains a particular set of words[Ex: String s1="hello";and SOP(s1.contains("deve")); checks whether s1 has word “deve”, as s1 is “hello” and does not have “deve” so the output will be false.

**Equals**: compare 2 string

**Important methods:**

**equalsIgnoreCase()**: return type:boolean

suppose we have

String s1="hello";

System.out.println("-----------");

System.out.println(s1.equals("Hello"));

Output will be false because here in passed object H is capital

In this case we choose,ignorecase method which checks irrespective of upper case and lower case letters ,checks only content

System.out.println(s1.equalsIgnoreCase("Hello"));

O/P : true

|  |
| --- |
| H |
| E |
| L |
| L |
| O |

0

1

2

3

4

**indexOf()**: return type :int

suppose we have

String s1="hello";

Indexof will refer to the char in that index

System.out.println(s1.indexOf('o'));

When you look at the table o is present at the 4 index

O/P will be 4

We can give string also as the argument

System.out.println(s1.indexOf("ll"));

Then it will look for the first character in “ll” which is l , so first l is there in the 2 index

O/P will be 2

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| i |  | L | o | V | e |  | j | A | V | A |  | a | n | d |  | s | e | l | e | n | i | u | m |

0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23

**lastIndexOf()**: this method returns int.

In the above string character ‘e’ is at 5,17,19 position, if we want to know the index of last ‘e’ , then we use this method

System.out.println(s1.lastIndexOf('e'));

O/P will be 19, which is the index of last ‘e’

If we give System.out.println(s1.indexOf('e');

Then the O/P will be 5 which is the index of first ‘e’

But there are many ‘e’ in the string, if we want to know the index of second ‘e’ then we need to use

System.out.println(s1.indexOf('e', 6));

Then this will search for e from 6 index and the O/p will be 17

If we want to know the index of third ‘e’

System.out.println(s1.indexOf('e', 18));

Then this will search for e from 18 index and the O/p will be 19

**replace(char,char)**;

this method will replace the char which is in first argument with the second argument

Ex: s1="i love java and selenium";

System.out.println(s1.replace('e','E'));

It wil replace the char ‘e’ with ‘E’

i lovE java and sElEnium

similarly we can use….

replaceAll(String,String),

where we can give strings in the arguments

System.out.println(s1.replaceAll("l","L"));

O/p i Love java and seLenium.

**charAt(int)**:

this method returns a char, number in the parameter depicts the index, and it will return the value at that position

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

O/P:

I

**endsWith(String)**:

this method returns boolean, we need to give the string in the parameter, this will check whether the orignal string ends with the

given string or not, and returns true or false

Ex: s1="i love java and selenium";

System.out.println(s1.endsWith("per"));

This will check whether s1 string is ending with string “per” or not

O/P is false

**startsWith(String)**

This method returns boolean, we need to give the string in the parameter, this will check whether the orignal string starts with the

given string or not, and returns true or false

Ex: s1="i love java and selenium";

System.out.println(s1.startsWith("java"));

This will check whether s1 string is starting with string “java” or not

O/P is false

String.compareTo(String)

**compareTo()** method does Unicode comparison and returns the difference of the Unicode

System.out.println("DA".compareTo("CA"));

This will take the difference of Unicode of D to C and difference of Unicode of A-A which is 67-66 and 65-65=1 and 0

O/P is 1

**toUpperCase()**

This will take the string and prints the string in uppercase

s1="i love java and selenium";

System.out.println(s1.toUpperCase());

This will take the s1 and prints s1 in uppercase

O/P: I LOVE JAVA AND SELENIUM

**toLowerCase()**

This will take the string and prints the string in lower case

s1="i love java and selenium";

System.out.println(s1.toLowerCase());

This will take the s1 and prints s1 in lowercase

O/P: i love java and selenium

Q. Print the given String in vertical fashion using for loop

Program:

public class Demo101 {

public static void main(String[] args) {

String s1="JackAndJill";

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

{

System.out.println(s1.charAt(i));

}

}

}

O/p:

J

a

c

k

A

n

d

J

i

l

l

Q. Write the program to print the reverse of given string

Program

public class Demo101 {

public static void main(String[] args) {

String s1="JackAndJill";

for(int i=s1.length()-1;i>=0;i--)

{

System.out.println(s1.charAt(i));

}

}

}

O/P:

l

l

i

J

d

n

A

k

c

a

J

**trim()**

This trim(); method removes the spaces if any, present in extreme right or extreme left corner

Of the string but not the spaces in between the string and it returns the new Object but not change the original string.

s1.trim(); and SOP(s1); it will print the original but not the modified one, because it will return the new object

s1.trim();

System.out.println(s1);

It will print the original value

For that reason

s1=s1.trim();

System.out.println(s1);

System.out.println(s1.trim().length());--This is called method chaining..

**13.substring(int)**

**Substring start index: from start index till last index**

Program:

public class Demo102 {

public static void main(String[] args) {

String s1="jackandjill";

System.out.println(s1.substring(4));

System.out.println(s1.substring(4,9));

}

}

O/P:

andjill

andji

substring(int) method, if we mention index in the int parameter from that index it will print the string

suppose

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| j | A | c | K | A | N | d | j | i | l | l |

0 1 2 3 4 5 6 7 8 9 10

If we give

System.out.println(s1.substring(4))

Index value of 4 is ‘a’ from there it will print the string

O/P:

Andjill

**substring(int,int)**;

This is also same like above method but in above we are giving the index value from where it should print but we are not specifying till where it should print, so now here we are

Giving till where it should print in the second parameter,second parameter depicts the position.

System.out.println(s1.substring(4,9));

o/p:

andji

**split(String):** string used to split original string will be omitted in the output

**example:** “ Malayalam”, here split string by “a”, the output will contain string array ignoring a

This method splits the sentence into word, In the parameters we have to give the where we want to split the sentence ,look at the below example

public class Splitmethod {

public static void main(String[] args) {

String s1="i love java";

String arr1[]=s1.split(" ");

for(int i=arr1.length-1;i>=0;i--)

{

System.out.println(arr1[i]+" ");

}

}

}

O/P:

java

love

i

Note:

String arr1[]=s1.split(" "); splits the string at the space and stores it in array

|  |
| --- |
| I |
| love |
| java |

0

1

2

**Escape sequence character**

A character with a backslash (\) just before it is an escape sequence or escape character.

|  |  |
| --- | --- |
| **Escape Characters** | **Description** |
| **\t** | It is used to insert a **tab** in the text at this point. |
| **\'** | It is used to insert a **single quote** character in the text at this point. |
| **\"** | It is used to insert a **double quote** character in the text at this point. |
| **\r** | It is used to insert a **carriage return** in the text at this point. |
| **\\** | It is used to insert a **backslash character** in the text at this point. |
| **\n** | It is used to insert a **new line** in the text at this point. |
| **\f** | It is used to insert a **form feed** in the text at this point. |
| **\b** | It is used to insert a **backspace** in the text at this point. |

1. //it inserts a Tab Space
2. String str = "Andrew\tGarfield";
3. System.out.println(str);
4. //it inserts a New Line
5. String str1 = "the best way\nto communicate \nan idea \nis to act it out";
6. System.out.println(str1);
7. //it insert a backslash
8. String str2 = "And\\Or";
9. System.out.println(str2);
10. //it insert a Carriage
11. String str3 = "Carriage\rReturn";
12. System.out.println(str3);
13. //it prints a single quote
14. String str4 = "Wall Street\'s";
15. System.out.println(str4);
16. //it prints double quote
17. //String str5 = "New\'Twilight'Line";
18. String str5 = "'JavaTpoint'";
19. System.out.println(str5);

Andrew Garfield

the best way

to communicate

an idea

is to act it out

And\Or

Carriage

Return

Wall Street's

'JavaTpoint'