**STRINGS:**

String Objects are created in two ways:

1. Using new Operator

2. Assignment

To understand better look at the below program:

Program:

package com.qsp.pack2;

public class Demo97 {

public static void main(String[] args) {

// TODO Auto-generated method stub

String s1="hello";

String s2=new String("banglore");

System.out.println(s1+" "+s2);

}

}

O/P

hello banglore

**Note:**

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

Ex:

String s1="hello";

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

Ex:

String s2=new String("banglore");

String constant pool

S1 s2

hello

banglor

non constant pool

Program:

package com.qsp.pack2;

public class Demo98 {

public static void main(String[] args) {

// TODO Auto-generated method stub

String s1="java";

String s2=new String("Developer");

String s3="java";

String s4=new String("Developer");

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

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

String s5=new String("java");

System.out.println(s1==s5);

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

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

System.out.println(s2.equals(s4));

System.out.println(s1.equals(s5));

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

}

}

O/P:

true

false

false

-----------

true

true

true

false

**Note:**

In the above program, first s1 object is created in string constant pool, next s2 object is created in non-constant pool, and after that compiler executing s3 it sees content of s3 and s1 are same, so does not create new string in string constant pool rather it assign it to same object created using s1. So now s1 and s3 are pointing towards same object because content is same, this happens only to the strings in string constant pool

Now, when compiler executing s4,though the s2 and s4 content are equal, irrespective of the content it will create one more object and s4 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 s1==s3 is true as they are pointing towards same object

That is the reason why s2==s4 is false ,as they are pointing towards different Object

Even though s1 and s5 content is same, but address is different, because s1 object is created in string constant pool and s5 object is created in non-constant pool, so both are different objects, with different address

That is the reason why s1==s5 is false

String constant pool

java

s5

s1

java

s3

developer

s4

developer

non constant pool

s2

**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.

**Note:**

In String class, equals method is automatically over ridden to compare the values

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

as both content is equal so its true

System.out.println(s2.equals(s4));

Even though both are different objects, the content is same, so its true

System.out.println(s1.equals(s5));

Even though both are different objects,one is in string constant pool and another is non constant pool,But the content is same, so its true

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

Content is different so its false

Program:

package com.qsp.pack2;

public class Demo99 {

public static void main(String[] args) {

String s1="java";//stored in string constant pool

String s2="Developer";////stored in string constant pool

String s3=s1+s2;// //stored in non constant pool

System.out.println(s3);

String s4="javadeveloper";////stored in string constant pool

System.out.println(s4);

System.out.println(s3==s4);//false, two address are different,one is in string constant pool and other is in non const

String s5="java"+"developer";//stored in string constant pool

System.out.println(s3==s5);// false, two address are different,one is in string constant pool and other is in non const

System.out.println(s4==s5);true,s4 is also in string constant pool and s5 is also in string constant pool

String s6=s1+"developer";//stored in non- constant pool

System.out.println(s3==s6);false, even though both are in non-constant pool, but both are different objects.

System.out.println(s4==s6); false, two address are different, one is in string constant pool and other is in non-constant pool

System.out.println(s4.equals(s6));//true, it will check only the content irrespective of address

}

}

O/P:

javaDeveloper

javadeveloper

false

false

true

false

false

true

Note: when we concatenate any reference variable with another reference variable, it will be stored in non-constant pool

Ex:

String s1="java";//stored in string constant pool

String s2="Developer";////stored in string constant pool

String s3=s1+s2;// //stored in non-constant pool

Even though s1 and s2 belongs to string constant pool, when they concatenate s3 will be stored in non-constant pool.

But, when you concatenate using two Strings, the resultant will be stored in string constant pool

String s5="java"+"developer";

S5 is stored in String constant pool

When you concatenate one reference variable and string the resultant will be stored in non-constant pool

String s6=s1+"developer";//

S6 will be stored in non-constant pool, whenever reference variable used, it will be stored in non-constant pool

String constant pool

developer

java

s1

Java developer

s3

Java developer

s2

s4

Java developer

s5

s6

Non constant pool

Program:

package com.qsp.pack2;

public class Demo100 {

public static void main(String[] args) {

String s1="hello";

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

System.out.println(s1.concat("developer"));

System.out.println(s1);

String s2=new String();

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

s2=s2.concat("banglore");

System.out.println(s1.contains("deve"));

s1.equals(null);

s1.equals(s2);

s1.equals("hello");

s1.equals(new String("hello"));

"hello".equals(s2);

"hello".equals(new String("hello"));

"hello".equals("hello");

}

O/p:

5

hellodeveloper

hello

false

false

**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.

Different ways of using equals method

String s1="hello";

String s2=”hello”;

1st way:

s1.equals(s2);

2nd way:

s1.equals("hello");

3rd way

s1.equals(new String("hello"));

4th way:

"hello".equals(s2);

5th way:

"hello".equals(new String("hello"));

6th way:

"hello".equals("hello");

Program:

public class Demo100 {

public static void main(String[] args) {

String s1="hello";

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

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

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

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

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

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

s1="i love java and selenium";

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

System.out.println(s1.indexOf('e', 7));

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

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

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

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

System.out.println(s1.charAt(s1.length()-1));

char ch=s1.charAt(0);

System.out.println(ch);

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

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

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

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

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

System.out.println("Sum of 4 and 5"+4+5);

System.out.println("sum of 4 and 5"+(4+5));

System.out.println(4+5+"is the sum of 4 and 5");

}

}

O/P:

-----------

false

true

4

2

4

19

17

19

i lovE java and sElEnium

i Love java and seLenium

i

m

i

false

false

1

i love java and selenium

I LOVE JAVA AND SELENIUM

Sum of 4 and 545

sum of 4 and 59

9is the sum of 4 and 5

**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:

package com.qsp.pack2;

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

package com.qsp.pack2;

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

Q.Write the program to print string using tocharArray method

Program:

package com.qsp.pack2;

public class Demo101 {

public static void main(String[] args) {

String s1="JackAndJill";

char arr[]=s1.toCharArray();

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

{

System.out.print(arr[i]+"\t");

}

}

}

O/P:

J a c k A n d J i l l

Note:

The difference between charAt(int) and tocharArray(), the method charAt(int) will take the value at the index and throws it, whereas tocharArray(),it takes the string and puts in the char array from there it will throws the values

Q.Write the program to print the unicode values of the characters in the given string

Program

package com.qsp.pack2;

public class Demo101 {

public static void main(String[] args) {

String s1="JackAndJill";

char arr[]=s1.toCharArray();

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

{

System.out.print((int)arr[i]+"\t");

}

}

}

O/P:

74 97 99 107 65 110 100 74 105 108 108

Note: here arr[i] prints the character, now if we cast it using int , i.e, (int)arr[i] it will print unicode values

Q. Write the program to print the next char of the present char I,e if A, then prints B, if J, print K

A. package com.qsp.pack2;

public class Demo101 {

public static void main(String[] args) {

String s1="JackAndJill";

char arr[]=s1.toCharArray();

//1st way

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

{

System.out.print((char)(arr[i]+1)+"\t");

}

//2nd way

//System.out.print((++arr[i]));

}

}

O/P:

K b d l B o e K j m m

Note: here, (arr[i]+1)prints the next unicode value of the present character, when we cast it with char then it will print character

Note:

Different Scenarios to find the length() of the String

Ex: 1.String s1=”hello”;

2.System.out.println(s1.length());

3.System.out.println(new String(“Hello”).length());

4.System.out.println(“Banglore”.length());

Program:

package com.qsp.pack2;

public class Demo102 {

public static void main(String[] args) {

String s1=" Hello ";

//1st way

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

s1=s1.trim();

System.out.println(s1);

//2nd way

System.out.println(s1.trim().length());

}

}

O/P:

11

Hello

5

**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..

Program:

package com.qsp.pack2;

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

13.substring(int)

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)**

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

package com.qsp.pack2;

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