DAY-25

-------

STRINGS

Strings is a collection of characters or a array of characters.It is a non-primitive datatype.In java Strings are treated as object becuase the memory

is allocated in heap segment.

Example: Sagar,StudyOnline,Java,September.......etc

Based on creation of the creation of the strings it is classified into two types:

1.IMMUTABLE STRINGS

2.MUTABLE STRINGS

IMMUTABLE STRING:

Immutable strings are such strings once it is created it can not be altered or changed.

Mulitiple ways in which immutable strings can be created.

1.String s = new String("sagar");

2.String s = "sagar";

3.char []a ={a,b,c,d};

4.byte[] a={97,98,99,100};

refer the diagram

-->Whenever we create a string using new keyword then the strings is created in heap segment[non-constant pool] and here duplicates are not allowed.

-->Whenever we create a string without using new keyword then the strings is created in heap segment[constant pool] and here duplicates are allowed.

COMPARING THE STRINGS USING STRING CLASS:

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

1. Using the ==

2. using the equals() method

3. using the equalsIgnorecase() method

4. using the compareTo() method

EXMAPLE:1

// comparing the two strings using the == symbol.

class Demo

{

public static void main(String[] args)

{

String s1 = "sagar";

String s2 = "sagar";

if(s1 == s2)

{

System.out.println("reference are equal");

}

else

{

System.out.println("reference are not equal");

}

}

}

OUTPUT:

-------

reference are equal

NOTE: == symbol will always compare the memory address of the object.

EXAMPLE:2

---------

// comparing the two strings using the equals() method .

class Demo

{

public static void main(String[] args)

{

String s1 = "sagar";

String s2 = "sagar";

if(s1.equals(s2))

{

System.out.println("Strings are equal");

}

else

{

System.out.println("Strings are not equal");

}

}

}

OUTPUT:

-------

Strings are equal.

NOTE: equals method will compare the data present inside the two string objects

EXAMPLE:3

---------

// comparing the two strings using the == symbol .

class Demo

{

public static void main(String[] args)

{

String s1 = new String("sagar");

String s2 = new String("sagar");

if(s1 == s2)

{

System.out.println("reference are equal");

}

else

{

System.out.println("refrence are not equal");

}

}

}

OUTPUT:

-------

refrence are not equal

EXAMPLE:4

---------

// comparing the two strings using the equals method .

class Demo

{

public static void main(String[] args)

{

String s1 = new String("sagar");

String s2 = new String("sagar");

if(s1.equals(s2))

{

System.out.println("String are equal");

}

else

{

System.out.println("String are not equal");

}

}

}

OUTPUT:

-------

String are equal

EXAMPLE:5

---------

// comparing the two strings using the equalsIgnoreCase() method .

class Demo

{

public static void main(String[] args)

{

String s1 = new String("sagar");

String s2 = new String("SAGAR");

if(s1.equalsIgnoreCase(s2))

{

System.out.println("String are equal");

}

else

{

System.out.println("String are not equal");

}

}

}

OUTPUT:

-------

String are equal

NOTE: equalsIgnoreCase() method will compare the data present in the string object by ignoring its case.This is method is used because java is

case sensitive.

EXAMPLE:6

---------

class Demo1

{

public static void main(String[] args)

{

String s1 = "sagar";

System.out.println(s1);

}

}

NOTE: java.lang.\* package(default). Whenever we try to use the String class in the program it will be by default imported from the jvm.

Whenever we try to print the reference (memory address) of a string, value is displayed instead of address.

EXAMPLE:7

---------

class Demo1

{

public static void main(String[] args)

{

String s1 =new String("sagar");

String s2 =new String("sagar");

String s3 ="sagar";

String s4 ="sagar";

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

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

}

}

OUTPUT:

-------

false

false