STRING

Without String we can’t code a single class.

1.String is a class and its non-primitive datatype. (All classes in java are non-primitive datatype.)

* Package- java. Lang (no import statement is needed).
* String class can also initialize the obj without new keyword. (e.g., String s=”abc” ;)
* Two ways to create String objects,
* Using new Keyword,

String s=**new** String ();

Whatever, the objects are created using new keyword in classes, are stored in heap memory.

* Without using new Keyword,

String s1="ABC" ;

Only in String we have the special approach, that we can create object without using new keywords, these objects are stored in String Constant pool(scp) special memory.

SCP:

It is the special memory Area in heap memory, to store string literal.

Java String class maintain SCP secretly in heap.

Creating more string objects in heap memory leads to increase in memory and decrease the performance.

To reduce this, jvm performs some steps to initialize string literal to increase performance and reduce memory loads.

Strings are immutable (not changeable) in nature.

When the string literal are created in ,first check in scp whether this literal present in scp ,if not then it creates new string literal.

Example:

**public** **class** StringDemo {

**public** **static** **void** main(String[] args) {

String s = **new** String("Hello");

System.***out***.println("The String in s with new keyword- "+s);

String s1 = "ABC";

System.***out***.println("The first string literal in scp for s1"+s1);

s1 = "pqr";

String s2 = "pqr";

String s3 = "pqr";

System.***out***.println("The string Literals changed for s1- "+s1);

System.***out***.println("The String object point to the literals present in scp s2- "+s2);

System.***out***.println("The String object point to the literals present in scp s3- "+s3);

s3="efg";

System.***out***.println("After reusing the literals in scp now new literals is created now its pointed s3- "+s3);

}

}

Output:

The String in s with new keyword- Hello

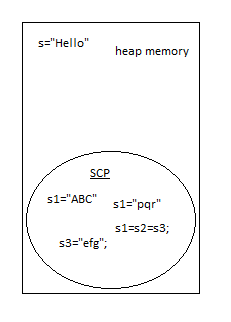
The first-string literal in scp for s1- ABC

The string Literals changed for s1- pqr

The String object point to the literals present in scp s2- pqr

The String object point to the literals present in scp s3- pqr

After reusing the literals in scp now new literals is created now its pointed s3- efg



**String-Methods** :

1.***toString()*** : whenever print the object toString will call internally

Whenever the object value is printed toString method calls internally,and its return the hashcode-hexadecimal values.

Example:

**package** string\_program;

**public** **class** String\_methods {

**public** String\_methods(String string) {

}

**public** **static** **void** main(String[] args) {

String\_methods sm = **new** String\_methods("Hi");

System.***out***.println(sm);

}

}

Output:

string\_program.String\_methods@340f438e

* toString method can be overriden for object value
* toString 🡪java.lang.object package

Example:

**package** string\_program;

**public** **class** String\_methods {

**public** String\_methods(String string) {

}

@Override

**public** String toString() {

**return** "Hello";

}

**public** **static** **void** main(String[] args) {

String\_methods sm = **new** String\_methods("Hi");

System.***out***.println(sm);

}

}

Output: Hello

* In String the toString method override the return value and print the string values.

Example:

**package** string\_program;

**public** **class** String\_methods {

**public** **static** **void** main(String[] args) {

String s = **new** String("Good");

System.***out***.println(s);

}

}

Output:Good

* Object and String return values:

Example:

**package** string\_program;

**public** **class** String\_methods {

**public** String\_methods(String string) {

}

@Override

**public** String toString() {

**return** "Hello";

}

**public** **static** **void** main(String[] args) {

String\_methods sm = **new** String\_methods("Hi");

String s = **new** String("Good");

System.***out***.println(sm);

System.***out***.println(s);

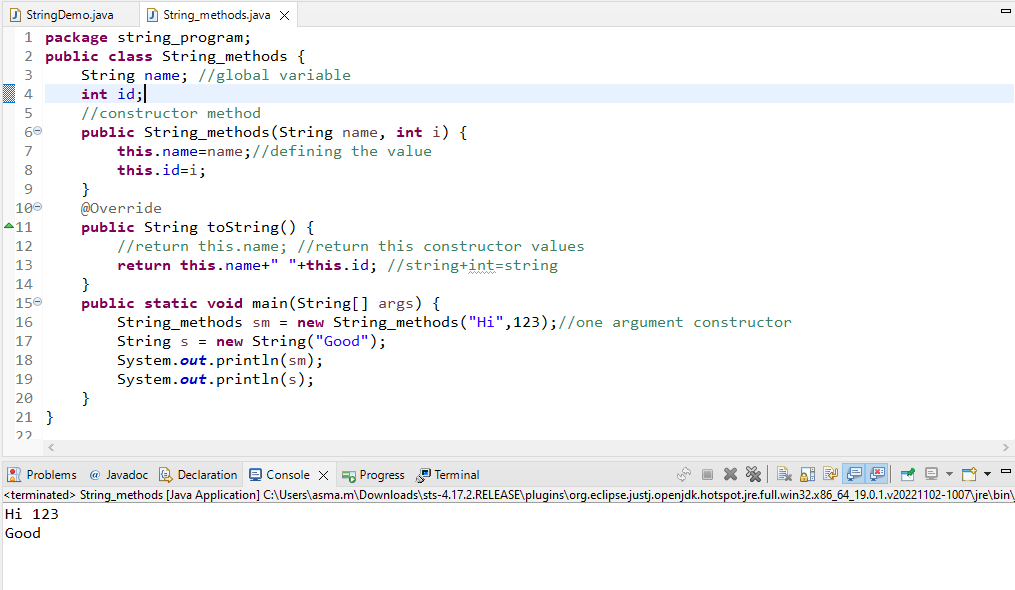
}

}

Output:

Hello

Good



2.***String-length()*** :

This method is used to find the length of String.

Example:

String s = **new** String("Good people");

System.***out***.println(s.length());

Output: 11

3.***CharAt ()*** :

This method is used to print the characters in specific index of string.

Syntax: charAt(int index)

Example: This example shows that string length, character present in string, same character present in string.

**package** string\_program;

**public** **class** String\_methods {

**public** **static** **void** main(String[] args) {

String s = **new** String("Good people");

System.***out***.println(s);

System.***out***.println(s.length());

**int** i=0; **int** count=0;

**while**(i<s.length())

{

**if**(s.charAt(i)=='o') //used to find how much same character present string

{

count++;

}

System.***out***.print("\n"+"The String characters : "+s.charAt(i));//used to print the character in string.

i++;

}

System.***out***.println("\n"+"The same character 'o' present in string is : "+count);

}

}

Output:

Length of s : 11

The String characters : G

The String characters : o

The String characters : o

The String characters : d

The String characters :

The String characters : p

The String characters : e

The String characters : o

The String characters : p

The String characters : l

The String characters : e

The same character 'o' present in string is : 3

3.***contains()*** : this method is used to check whether the given character present in the string or not.

Returns Boolean value(truw/false).

Example:

**public** **static** **void** main(String[] args) {

String m="java developer";

System.***out***.println("The String contains a? : "+m.contains("a"));

System.***out***.println("The String contains a? : "+m.contains("i"));

}

Output:

The String contains a? : true

The String contains a? : false

4.***String-Concat()*** : this method will concat (join two strings) or add two strings.

Example:

**public** **static** **void** main(String[] args) {

String m="java developer ";

String n="world!";

System.***out***.println("String Concatenation : "+m.concat(n));

}

Output:

String Concatenation : java developer world!

5.***String-endsWith()*** :this method will check that given string is ends with defined string or character.

Example:

**public** **static** **void** main(String[] args) {

String m="java developer";

System.***out***.println("String Endwith er ? : "+m.endsWith("r"));

}

Output:

String Endwith er ? : true

5.***String-startsWith()***:this method will check that given string is starts with defined string or character.

Example:

**public** **static** **void** main(String[] args) {

String m="java developer";

System.***out***.println("String Startswith j ? : "+m.startsWith("j"));

}

Output:

String Startswith j ? : true

6.***String-trim()*** : this method will remove the whitespace from both ends in the String. Does not change original String.

Example:

**public** **static** **void** main(String[] args) {

String m=" java developer ";

String p=m.trim();

System.***out***.println(m.length()+"\n"+p+"\n"+p.length());

}

Output:

16

java developer

14

7.