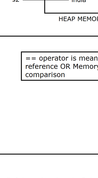
****What is the difference between the following two statements ?

Case 1 :

```
String s1 = "india";

String s2 = new String("Java");

* String s1 = "india";
* In this statement only one object will be
  created in the SCP area and the same
  Object is refered by s1 reference variable.

* String s2 = new String("Java");
* In this statement two Java objects will be
  created one Java object will be created inside
  Heap Area (non SCP Area) and it is refered
  by s2 reference variable, whereas one more
  java object (due to " ") will be placed inside
  SCP Area and it is an un-referenced object
  meant for Re-usability.
```

Case 2 :

```
String s1 = "india";

String s2 = new String("india");
```



Note :- IF WE ARE CREATING THE STRING BY USING LITERAL OR BY USING NEW KEYWORD, STRING OBJECTS ARE NOT ELIGIBLE 4 GC AS WELL NOT ELIGIBLE FOR MODIFICATION.

//Program for String comparison :

```
package com.ravi.String_handling;

public class StringComparison
{
    public static void main(String[] args)
    {
        String s1 = "java";
        String s2 = new String("java");

        System.out.println(s1==s2);//false
    }
}
```

== operator is meant for reference OR Memory Address comparison

---

public String intern() :

* It is a predefined non static method of String class.
* It is used to canonical representation of String Object. Actually It is used to place the String object in the SCP area but if SCP area is already containing the String object then it will return the address of that String object which is known as canonical representation of String Object.

Case 1 :

```
String s1 = "Java";
String s2 = new String("Java");

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

s2 = s2.intern();

System.out.println(s1==s2); //true
```

//Program :

```
package com.ravi.String_handling;

public class StringComparison
{
    public static void main(String[] args)
    {
        String s1 = "Java";
        String s2 = new String("Java");

        System.out.println(s1==s2); //false
        s2 = s1.intern();
        System.out.println(s1==s2); //true
    }
}
```

Case 2 :

```
String s1 = "Hyd";
String s2 = new String("Hyd");

String s3 = s2.intern();
```


```
package com.ravi.String_handling;

public class StringComparison
{
    public static void main(String[] args)
    {
        String s1 = "Hyd";
        String s2 = new String("Hyd");

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

        String s3 = s2.intern();

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

        System.out.println(s1==s3); //true
    }
}
```

Working with methods of String :-

String class has provided number of predefined methods to work with String which are as follows :

1) public char charAt(int indexPosition) : Used to retrieve a particular character from the given String.

```
package com.ravi.String_handling;

public class CharAtDemo
{
    public static void main(String[] args)
    {
        String x = "Hello Hyderabad";

        char ch1 =  x.charAt(6); //H
        System.out.println(ch1);

        ch1 = x.charAt(4);
        System.out.println(ch1); //o

        ch1 = x.charAt(9);
        System.out.println(ch1); //e

    }

}
```

---

public String concat(String str) : It is used to append OR concat two Strings. Concatenation is also possible by using '+' operator.

```
package com.ravi.String_handling;

public class ConcatDemo {

    public static void main(String[] args)
    {
        String s1 = "Data";
        String s2 = "base";
        String s3 = s1.concat(s2);
        System.out.println("String after concatenation :"+s3);


        String s4 = "Tata";
        String s5 = "Nagar";
        String s6 = s4+s5;
        System.out.println("String after concatenation :"+s6);

        String s7 = "Naresh";
        System.out.println(s7.concat(" Technology"));
    }

}
```

---

public boolean equals(Object obj) :

* Used to compare two Strings based on the content regardless of memory address.
* It is case sensitive method
* It takes Object as a parameter because It is an Overridden method.

```
package com.ravi.String_handling;

public class StringEquality
{

    public static void main(String[] args)
    {
        String s1 = "India";
        String s2 = "India";
        String s3 = "India";

        System.out.println(s1.equals(s2));
        System.out.println(s1.equals(s3));

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

        String s4 = "Java";
        String s5 = new String("Java");
        System.out.println(s4==s5);
        System.out.println(s4.equals(s5));
    }

}
```

---

public boolean equalsIgnoreCase(String str) :

* Used to compare two Strings by ignoring the case.

```
package com.ravi.String_handling;

public class StringEqualityByIgnoringTheCase
{
    public static void main(String[] args)
    {
        String s1 = "HyDeRaBAD";
        String s2 = "HYDERABAD";

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

    }

}
```

IQ :

What is the difference between == operator and equals(Object obj) method of String class while comparing the String Object?

== Operator is meant for reference OR memory address comparison, on the other hand the overridden equals(Object obj) method is used for content comparison.

```
package com.ravi.String_handling;

public class StringEquality {

    public static void main(String[] args)
    {
        String s4 = "Java";
        String s5 = new String("Java");
        System.out.println(s4==s5); //false
        System.out.println(s4.equals(s5)); //true

    }

}
```

---

public String replace(char old, char new) :
public String replace(CharSequence old, CharSequence new) :

* Used to replace a particular character OR a particular String from the given String.

```
package com.ravi.String_handling;

public class StringReplacement
{
    public static void main(String[] args)
    {
        String s1 = "ababababab";
        System.out.println("Original String :"+s1);
        System.out.println("After Replacement :"+s1.replace('b', 'B'));

        System.out.println("...................");
        String s2 = "Manager";
        System.out.println("Original String :"+s2);
        System.out.println("After Replacement :"+s2.replace("Man","Dam"));
    }
}
```