java.lang.String
==========
String it refers to an Object in java present in package called java.lang.String(C)
String refers to collection of characters.

```
eg:: String s= "sachin";
       System.out.println(s);//sachin

      String s =new String("sachin");
      System.out.println(s);//sachin
```

In java String object is by default immutable,meaning once the object is created we cannot change the value of the
object, if we try to change then those changes will be reflected on the new object not on the existing object.

```
case 1::
         String s= "sachin";
         s.concat("tendulkar");//(new object got created with modification so
immutable)
         System.out.println(s);

         output::sachin

                  vs

         StringBuilder sb=new StringBuilder("sachin");
         sb.append("tendulkar");//(on the same object modification so mutable)
         System.out.println(sb);

         output:: sachintendulkar

case 2:: String s1 = new String("sachin");
             String s2 = new String("sachin");
             System.out.println(s1==s2); //false
             System.out.println(s1.equals(s2));//true
           => String class .equals method will compare the content of the object if
same return true otherwise return false

                                        vs

            StringBuilder sb1 = new StringBuilder("sachin");
            StringBuilder sb2 = new StringBuilder("sachin");
            System.out.println(sb1==sb2); //false
            System.out.println(sb1.equals(sb2));//false
            => StringBuilder class .equals method for reference comparison
                 if differnt object returns false,even if the contents are same.


case 3:: String s =new String("sachin");
         In this case 2 objects will be created one in the heap and the other one
in
         the String Constant Pool, the reference will always point to Heap.
                              vs

        String s ="sachin";
          In this case only one object will be created in the SCP and it will be
refered
```

by our reference.