DAY-48

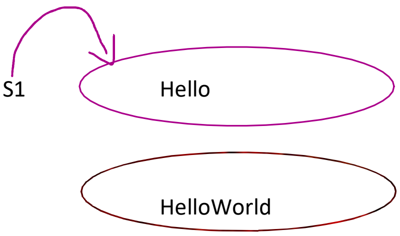
31 August 2023 15:35

IMMUTABLITY

WHAT IS IMMUTABLITY ?

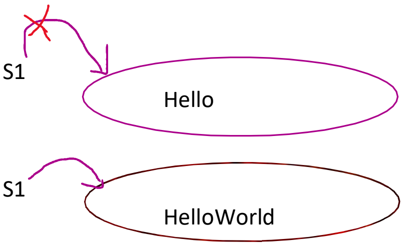
- WHENEVER WE CRATE AN OBJECT OF STRING CLASS, WE CAN NOT CHANGE CONTENT OF THAT OBJECT. IF WE TRY TO CHANGE THE CONTENT OF THE OBJECT , WITH THE CHANGES , A NEW OBJECT WILL BE CREATED.
- THIS UNCHANGEABLE BEHAVIOUR IS KNOWN AS IMMUTABILITY.
- STRING IS IMMUTABLE. I.E, WHENVER WE TRY TO PERFORM ANY CHANGES ON STRING OBJECT , WITH THOSE CHNAGES A NEW OBJECT WILL BE CREATED.

```
String s1 = "Hello";
  s1.concat("World");
Sop(s1);//Hello
```



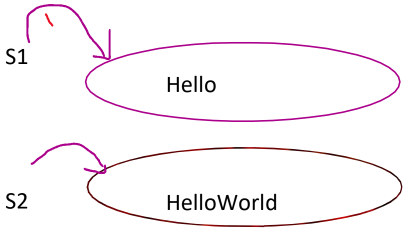
With new changes , a new object got created

```
String s1 = "Hello";
 s1 = s1.concat("World");
Sop(s1);//HelloWorld
```



With new changes, new object got created and s1 starts referring to new object

```
String s1 = "Hello";
String s2 = s1.concat("World");
Sop(s1);//Hello
```



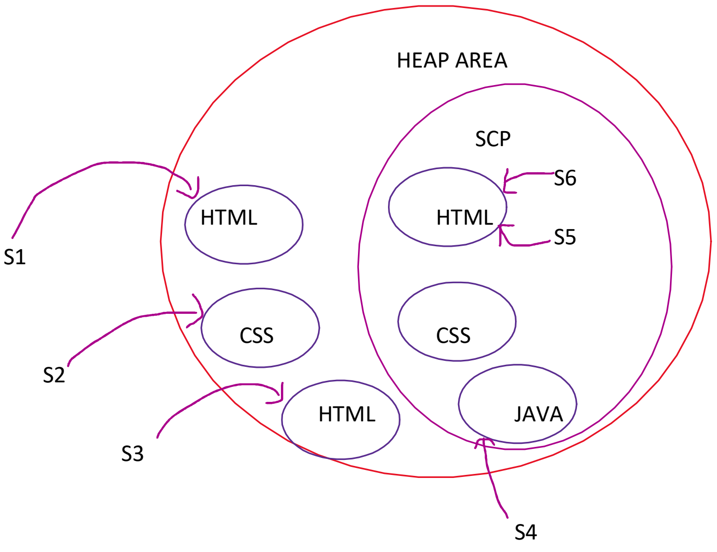
With changes, new objects got created and s2 reference referring to new object , s1 still referring to old object

STRING CONSTANT POOL AND HEAP AREA

- WHENVER WE CREATE STRING OBJECT USING NEW KEYWORD, THE OBJECT WILL BE STORED IN HEAP AREA AND A COPY OF THAT OBJECT WILL BE CREATED IN STRING CONSTANT POOL AND WILL BE REFERED BY JVM REFERENCE.
- WHENVER WE CREATE A STRING OBJECT WITHOUT USING NEW KEYWORD, OBJECT WILL BE CREATED IN STRING CONSTANT POOL AND NO COPY OF THAT OBJECT WILL BE GIVEN TO HEAP AREA.

```
String s1 = new String("Html");//2 OBJECTS
String s2 = new String("CSS");//2 OBJECTS
String s3 = new String("HTML");// 1 OBJECT


String s4 = "JAVA";//1 OBJECT
String s5 = "HTML";//0 OBJECT
String s6 ="HTML";//0 OBJECT
```



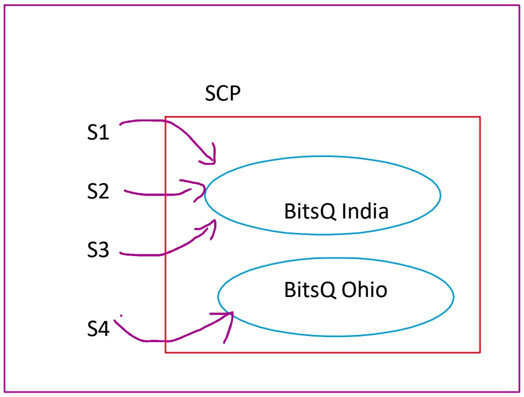
- IF ALREADY THERE IS AN OBJECT WITH SAME CONTENT IS PRESENT IN **SCP** , AGAIN NEW OBJECT WILL NOT BE CREAED BUT THE REFERENCE VARIABLE WILL START REFERING TO SAME EXISTING OBJECT WITH SAME CONTENT.

WHY IS STRING CLASS IMMUTABLE ?

- THERE IS CHANCE OF MULTIPLE REFERENCES REFERRING TO SAME STRING OBJECT, IF WE CHANGE THE CONTENT OF OBJECT USING ANY ONE OF REFERENCES, IT WILL AFFECT THE CONTENT OF ALL THE OTHER REFERENCES.
- TO AVOID THIS , STRING CLASS OBJECT IS IMMUTABLE.

```
String s1 = "BitsQ India";
String s2 = "BitsQ India";
String s3 = "BitsQ India";


String s4 = "BitsQ Ohio";
```



- ALL STUDENTS S1, S2 AND S3 BELONGS TO BitsQ India BRANCH.
- IF S3 WANTS TO CHANGE TO BitsQ Ohio, WE NEED TO CHANGE ONLY S3 TO Ohio. NOT S1 AND S2
- THIS IS POSSIBLE ONLY IF STRING CLASS IS IMMUTABLE.

<u>PROBLEM OF IMMUTABLE AND HOW TO OVERCOME IT</u>

- SINCE, STRING CLASS IS IMMUTABLE, WHENEVER WE TRY TO PERFORM ANY CHANGES TO PROPERTIES OF STRING OBJECT, WE END UP CREATING A NEW OBJECT EVERY TIME WITH CHANGES WHICH LEADS TO MEMORY LEAKAGE PROBLEM.
- TO ADDRESS THIS MEMORY LEAK PROBLEM , JAVA INTRODUCED 2 CLASSES RELATED TO STRING CLASS :
    1. StringBuffer
    2. StringBuilder
- THESSE CLASSES ARE MUTABLE i.e, EACH TIME WE PEFROM CHANGES TO CONTENT OF OBJECT , IT WILL ACCEPT CHANGES RATHER THAN CREATING NEW OBJECT.

- THESE CLAASES ARE ALSO FINAL.
- WE CAN CREATE OBJECTS TO STRING BUFFER AND STRING BUILDER ONLY WITH THE HELP OF NEW KEYWORD.

```
package string;

public class StringMethods {

        public static void main(String[] args)
        {
                StringBuffer s1 = new StringBuffer("java");
                s1.append(" class");
                System.out.println("StirngBuffer object "+s1);

                StringBuilder s2 = new StringBuilder("java");
                s2.append(" class");
                System.out.println("StirngBuilder object "+s2);

                String s3 = new String("java");
                s3.concat(" class");
                System.out.println("String object "+s3);

        }

}
```