DAY 5

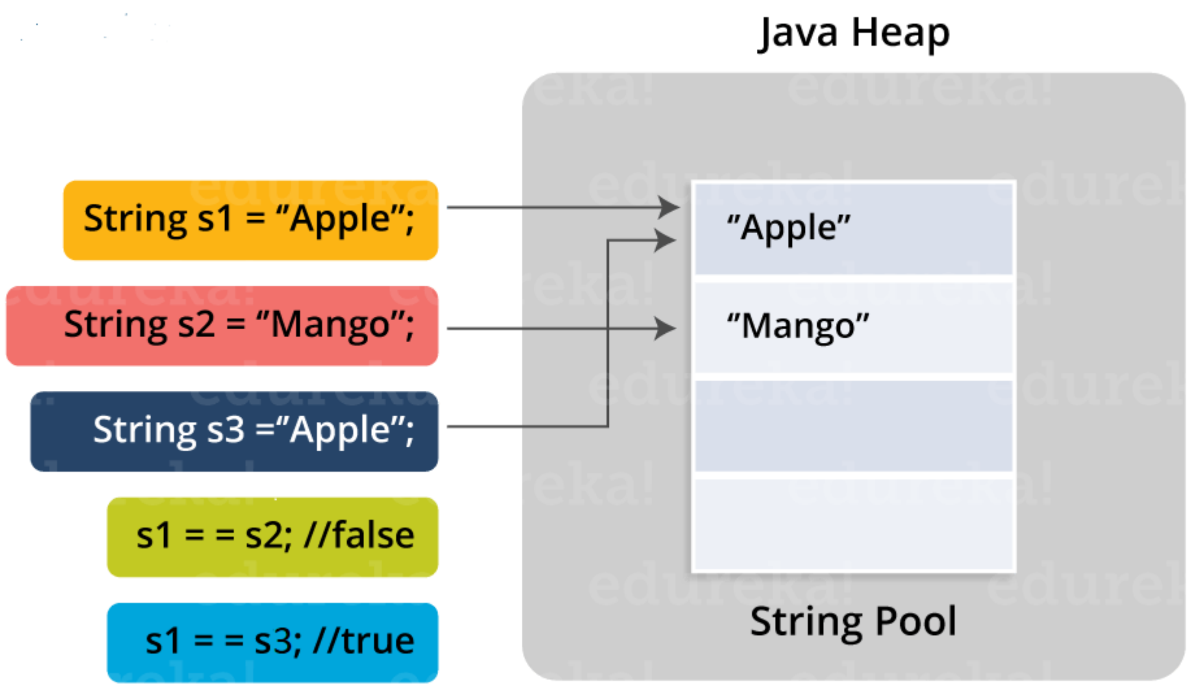
STRING METHODS

Java, a string is an object that represents a sequence of characters or char values. The *java.lang.String* class is used to create a Java string object.

There are two ways to create a String object:

1. **By string literal** : Java String literal is created by using double quotes.
   For Example: String s="Welcome";
2. **By new keyword** : Java String is created by using a keyword "new".
   For example: String s=new String("Welcome");
   It creates two objects (in String pool and in heap) and one reference variable where the variable 's' will refer to the object in the heap.

Java String Pool: Java String pool refers to collection of Strings which are stored in heap memory. In this, whenever a new object is created, String pool first checks whether the object is already present in the pool or not. If it is present, then same reference is returned to the variable else new object will be created in the String pool and the respective reference will be returned. Refer to the diagrammatic representation for better understanding:



in the above image, two Strings are created using literal i.e "Apple" and "Mango". Now, when third String is created with the value "Apple", instead of creating a new object, the already present object reference is returned. That's the reason Java String pool came into the picture.

Before we go ahead, One key point I would like to add that unlike other data types in Java, Strings are immutable. By immutable, we mean that Strings are constant, their values cannot be changed after they are created. Because String objects are immutable, they can be shared. For example:

```
  String str =”abc”;
```

is equivalent to:

```
char data[] = {‘a’, ‘b’, ‘c’};
   String str = new String(data);
```

Let us now look at some of the inbuilt methods in String class.

Java String Methods

- **Java String length()**: The Java String length() method tells the length of the string. It returns count of total number of characters present in the String. For example:
- Here, String length() function will return the length 5 for s1 and 7 for s2 respectively.

```
public class Example{
public static void main(String args[]{
String s1="hello";
String s2="whatsup";
System.out.println("string length is: "+s1.length());
System.out.println("string length is: "+s2.length());
}}
```

Java String compareTo(): The Java String compareTo() method compares the given string with current string. It is a method of *'Comparable'* interface which is implemented by String class. Don't worry, we will be learning about String interfaces later. It either returns positive number, negative number or 0. For example:

```
public class CompareToExample{
public static void main(String args[]){
String s1="hello";
String s2="hello";
String s3="hemlo";
String s4="flag";
System.out.println(s1.compareTo(s2)); // 0 because both are e
System.out.println(s1.compareTo(s3)); //-1 because "l" is onl
lower than "m"
System.out.println(s1.compareTo(s4)); // 2 because "h" is 2 ti
than "f"
}}
```

This program shows the comparison between the various string. It is noticed that
if s1 > s2, it returns a positive number
if s1 < s2, it returns a negative number
if s1 == s2, it returns 0

- **Java String concat() :** The Java String concat() method combines a specific string at the end of another string and ultimately returns a combined string. It is like appending another string. For example:

```
public class ConcatExample{
public static void main(String args[]){
String s1="hello";
s1=s1.concat("how are you");
System.out.println(s1);
}}
```

- The above code returns “hellohow are you”.

- **Java String IsEmpty()** : This method checks whether the String contains anything or not. If the java String is Empty, it returns true else false. For example:

```
public class IsEmptyExample{
public static void main(String args[]){
String s1="";
String s2="hello";
System.out.println(s1.isEmpty());      // true
System.out.println(s2.isEmpty());      // false
```

```
    }}
```

- **Java String Trim()** : The java string trim() method removes the leading and trailing spaces. It checks the unicode value of space character (‘u0020’) before and after the string. If it exists, then removes the spaces and return the omitted string. For example:

```
public class StringTrimExample{
public static void main(String args[]){
String s1="  hello   ";
System.out.println(s1+"how are you");        // without
trim()
System.out.println(s1.trim()+"how are you"); // with
trim()
}}
```

- In the above code, the first print statement will print “hello how are you” while the second statement will print “hellohow are you” using the trim() function.
- **Java String toLowerCase()** : The java string toLowerCase() method converts all the characters of the String to lower case. For example:

```
    public class StringLowerExample{
    public static void main(String
    args[]){
    String s1="HELLO HOW Are You?”;
    String s1lower=s1.toLowerCase();
    System.out.println(s1lower);}
    }
```

- The above code will return “hello how are you”.
- **Java String toUpper()** : The Java String toUpperCase() method converts all the characters of the String to upper case. For example:

```
    public class StringUpperExample{
    public static void main(String args[]){
    String s1="hello how are you";
```

```
String s1upper=s1.toUpperCase();
System.out.println(s1upper);
}}
```

- The above code will return “HELLO HOW ARE YOU”.

- **Java String ValueOf()**: This method converts different types of values into string.Using this method, you can convert int to string, long to string, Boolean to string, character to string, float to string, double to string, object to string and char array to string. The signature or syntax of string valueOf() method is given below:
  public static String valueOf(boolean b)
  public static String valueOf(char c)
  public static String valueOf(char[] c)
  public static String valueOf(int i)
  public static String valueOf(long l)
  public static String valueOf(float f)
  public static String valueOf(double d)
  public static String valueOf(Object o)