String

1. *Difference between .equals() &==.?*
2. *What is immutable class?*
3. *Give example in java and how can create it.*
4. *What is mutable?*
5. *Difference between mutable and Immutable class.*
6. *Why Java Strings are immutable in nature?*
7. *Difference between String, String Buffer and String Builder.*
8. *Difference logical question on String(.equals() and ==)*
9. *How is the creation of a String using new() difference from that of a Literal?*

Program Question

1. *WAP to Reverse the String?*
2. *Palindrome String*
3. *Find the Length of given String.*
4. *Occurrence of character in a given String.*
5. *Remove Duplicate characters from String.*
6. *Give a String in Alpha Numeric Chart, Filter out only Alphabets "A".*
7. *"RAM SHYAM" Remove space and print.*
8. *"RAM SHYAM" Print Both words in Different line.*

String Class

String is a class in java which consists of several build in method using which we can manipulator string data easily.

- **Creating String:** There are two ways in which a String object can be created in Java:

Using a string literal: String literal in Java can be created using double-quotes.

Example: `String s= "Hello World!";`

Using the new keyword: Java String can be created by using the keyword "new".

Example: `String s=new String ("Hello World!");`

Mutable Class

1. *Mutable an object are one whose values keep changing.*

```
package mutable;
//MUTABLE CLASS WHOSE VALUE CHANGING 10 TO 20
public class B {
int i=10;
public static void main(String[] args) {
	A a=new A();
	a.i=20;
}
}
```

SETTER & GETTER METHOD

Setter and getter both methods which help us to initialized variables and methods that help me to read the variable of value.

```
package mutable;

public class A {
private int id;
private String name;
public int GetID() {
	return id;
}
public void SetID(int id) {
	this.id=id;// INITIALIZED THE VALUE
}
public String GetNAME() {
	return name;// READ THE VALUE
}
public void SetNAME(String name) {
	this.name=name;
}
```

```
public static void main(String[] args) {
	A a1=new A();
	a1.SetID(12);
	a1.SetNAME("xyz");
	System.out.println(a1.GetID());
	System.out.println(a1.GetNAME());
}
}
OUTPUT:
12
xyz
```

Immutable Class

1. Immutable an object are one whose values can never alter.

2. *String class is immutable class.*

   Rules to design Immutable class:

   1. *Make class final.*
   2. *Make variable final & private.*
   3. *Initialize variable through constructor.*
   4. *Use only getter & don't use setter.*

```
package p1;

final class A {
	}
	final private int i;
	final private int j;
	public int getJ() {
		return j;
	}
	public int getI() {
		return i;
	}
	A(int i, int j) {
		this.i=i;
		this.j=j;
		public static void main(String[] args) {
			A a1=new A(10,20);
			System.out.println(a1.getI());
			System.out.println(a1.getJ());
		}

}
```

Q Why Java Strings are immutable in nature?

In java, string objects are immutable in nature which simple means once the string object is created it state cannot be modified. Whenever you try to update the value of that object insert of updating the value of that particular object, java created a new string object.

Q. Difference between mutable and immutable class.

MUTABLE CLASS	IMMUTABLE CLASS
1. *Mutable class an object whose value keep changing.*	1. *Immutable an object are one whose values can never alter.*
2. *In mutable objects, no new object is formed.*	**2.** *In immutable objects, new object are formed when the values can never alter.*
3. *Its provide methods to the object.*	*3. It does not provide any methods to change the object.*
4. *Its support gets () and set () method to deal with object.*	*Only support get () method*

Q. Difference between .equals() &==.?

"=="

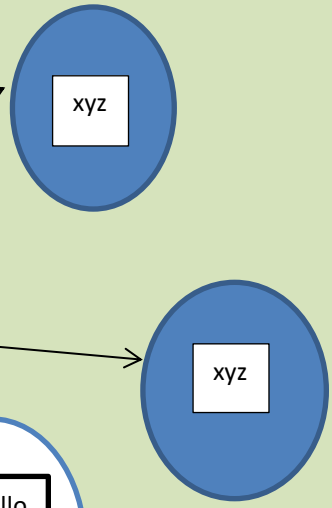
1. *Compare memory address of string objects.*
2. *It is a operator.*

".equals()"

1. Compares values of String.
2. It is a method.

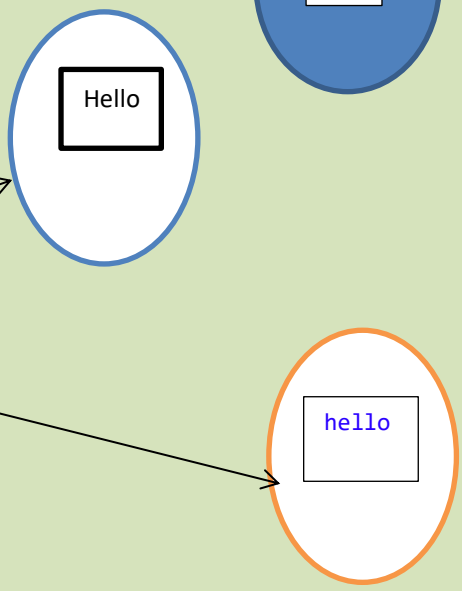
Example1:

```
package p2;

public class A {
public static void main(String[] args) {
	String s1=new String("xyz");
	String s2=new String("xyz");
	System.out.println(s1=s2);//false
	System.out.println(s1.equals(s2));//true
}
}
```



Example2:

```
package p2;

public class A {
public static void main(String[] args) {
	String s1=new String("Hello");
	String s2=new String("hello");
	System.out.println(s1=s2);//false
	System.out.println(s1.equals(s2));//false
}
}
```



Example3

```
package p2;

public class A {
public static void main(String[] args) {
	String s1="xyz";
```

```
        String s2="xyz";
        String s3="xxx";
        String s4="xyz";
        System.out.println(s1=s2);//true
        System.out.println(s2==s4);//true
        System.out.println(s1==s3);//false
}
}
```

Output:

```
xyz
true
false
```

```
package string;

public class Test {
        public static void main(String[] args)
        {
                Thread t1 = new Thread();
                Thread t2 = new Thread();
                Thread t3 = t1;

                String s1 = new String("GEEKS");
                String s2 = new String("GEEKS");

                System.out.println(t1 == t3);
                System.out.println(t1 == t2);
                System.out.println(s1 == s2);

                System.out.println(t1.equals(t2));
                System.out.println(s1.equals(s2));
        }
}
```

OUTPUT:

```
true
false
false
false
true
```

```
package string;

//Java program to illustrate
//== operator for incompatible data types

class Test {
        public static void main(String[] args)
        {
                Thread t = new Thread();
                Object o = new Object();
                String s = new String("GEEKS");

                System.out.println(t == o);
                System.out.println(o == s);

        // Uncomment to see error
        System.out.println(t==s);
        }

}
```

OUTPUT:

```
_false

false

// error: incomparable types: Thread and String


// Java program to illustrate
// == operator for compatible data
// types

class Test {
        public static void main(String[] args)
        {
                // integer-type
                System.out.println(10 == 20);

                // char-type
                System.out.println('a' == 'b');

                // char and double type
                System.out.println('a' == 97.0);

                // boolean type
                System.out.println(true == true);
        }
}
```

Output

```
false

false

true

true
```

INBUILD FUNCTION:

Methods of string functions in Java

Following are the different methods:

Method	Description
to Lowercase()	**It returns a string with all chars in to lowercase**
toUpperCase()	**It returns a string with all chars in uppercase**
trim()	**It removes the starting and ending whitespaces of this string.**
Split()	**Print in different line**

Int length()	**It returns the length of the string**
char charAt(int index)	**It returns the char value of the particular index as mentioned.**
Startswith() & endswith()	

1. To Uppercase() & To Lowercase()

```
package builtmethod;

public class D {

	public static void main(String[] args) {
		String s1= "PAnKaj";
		System.out.println(s1.toLowerCase());
		System.out.println(s1.toUpperCase());
	}
}
```

Output:

```
pankaj
PANKAJ
```

2. Trim()

```
package builtmethod;
//trim():used to removed white space from beginning and ending of string
public class RemoveWhiteSpace {
public static void main(String[] args) {
	String s1=" RAMSHYAM";
	System.out.println(s1);
	System.out.println(s1.trim());
}
}
```

Output:

```
 RAMSHYAM
RAMSHYAM
```

3. Split()

```
package builtmethod;
//split(): used to print different line
public class PrintDifferentLine {
public static void main(String[] args) {
String s1="RAM SHYAM";
String[] s2=s1.split(" ");
System.out.println(s2[0]);
System.out.println(s2[1]);
}
}
```

Output:

```
RAM
SHYAM
```

4. Length()

```
package builtmethod;

public class FindLengthOfString {
public static void main(String[] args) {
	String s1="I want to run";
	System.out.println(s1.length());
	String[] s2=s1.split(" ");
	System.out.println(s2[0].length());
	System.out.println(s2[1].length());
	System.out.println(s2[2].length());
	System.out.println(s2[3].length());
}
}
```

Output:

```
13
1
4
2
3
```

5. charAt()

```
package builtmethod;
//charAt()
public class A {
public static void main(String[] args) {
	String s1="ManiShanker";
	System.out.println(s1.charAt(0));
	System.out.println(s1.charAt(1));
	System.out.println(s1.charAt(2));
	System.out.println(s1.charAt(3));
	System.out.println(s1.charAt(4));
	System.out.println(s1.charAt(5));
}
}
Output:
M
a
n
i
S
H

package builtmethod;

public class B {
	public static void main(String[] args) {
		String s1="testing";
	System.out.println(s1.charAt(2));

}
}

Output: s
```

```java
package builtmethod;

public class C {
	public static void main(String[] args) {
		String s1="testing";

		for(int i=0;i<s1.length();i++) {
		System.out.print(s1.charAt(i));

}
	}
}
```

Output: testing

Q WAP TO REVERSE THE STRING

```java
package builtmethod;

import java.util.Scanner;

public class ReverseString {
public static void main(String[] args) {
	Scanner sc=new Scanner(System.in);
	System.out.println("Enter your String");
	String str=sc.next();
	String reverse="";
	for(int i=str.length()-1;i>=0;i--) {
		reverse=reverse+str.charAt(i);

	}
	System.out.println(reverse);
}
}
```

Output:

```
Enter your String
TESTING
GNITSET
```

Q PALINDROME STRING

```java
package builtmethod;

import java.util.Scanner;

public class PalindromString {
public static void main(String[] args) {
	Scanner sc=new Scanner(System.in);
	System.out.println("Enter your String");
	String str=sc.next();
	String org_str=str;
	String rev="";

	for(int i=str.length()-1;i>=0;i--) {
		rev=rev+str.charAt(i);
	}//.equals():compare the value of string
	if(org_str.equals(rev)) {
```

```
                System.out.println(org_str+"Palindrom String");
        } else {
                System.out.println(org_str+"Not Palindrom");

        }
}
}
```

```
OUTPUT:
Enter your String
MADAM
MADAMPalindrom String
```

Given a string and a character, the task is to make a function which count occurrence of the given character in the string.

Examples:

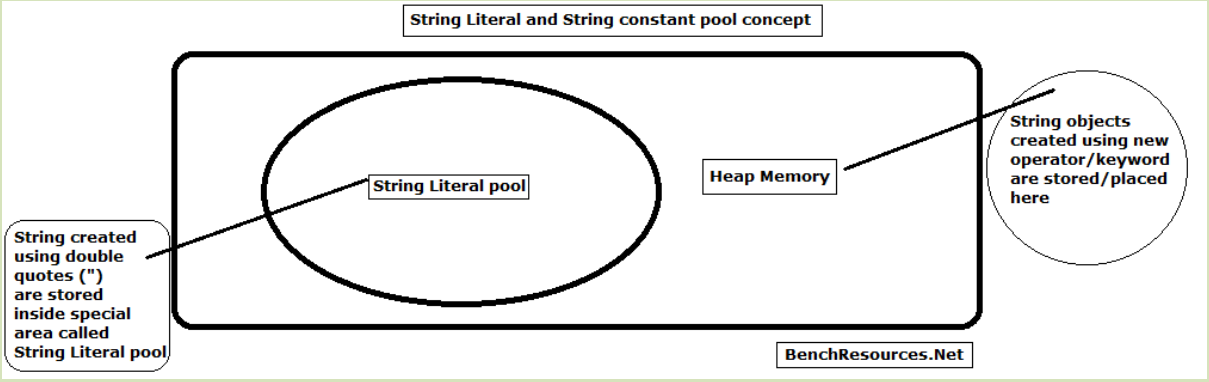
```
Input : str = "geeksforgeeks"
          c = 'e'
Output : 4
'e' appears four times in str.


Input : str = "abccdefgaa"
           c = 'a'
Output : 3
'a' appears three times in str.
```

```
package builtmethod;
//JAVA program to count occurrences
//of a character

class GFG
{
        // Method that return count of the given
        // character in the string
        public static int count(String s, char c)
        {
                int res = 0;

                for (int i=0; i<s.length(); i++)
                {
                        // checking character in string
                        if (s.charAt(i) == c)
                        res++;
                }
                return res;
        }

        // Driver method
        public static void main(String args[])
        {
                String str= "geeksforgeeks";
                char c = 'e';
                System.out.println(count(str, c));
        }
}
```

```
OUTPUT:4
```

Q. **string pool concept** : **String pool** is nothing but a storage area in Java heap where string literals stores. It is also known as **String Intern Pool** or **String Constant Pool**.

It is just like object allocation. By default, it is empty and privately maintained by the **Java String** class. Whenever we create a string the string object occupies some space in the heap memory. Creating a number of strings may increase the cost and memory too which may reduce the performance also.



As you can see from above figure, there are 2 areas to store strings in Java,

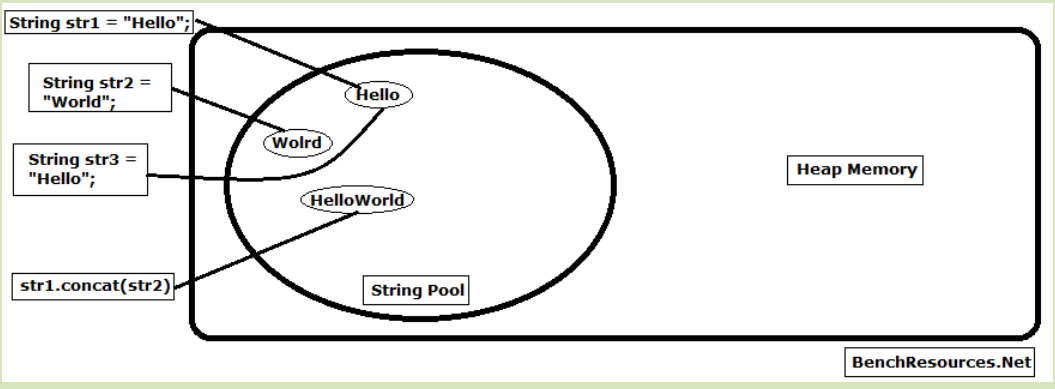
1. Heap memory –> area where other Java objects is stored
2. String pool –> to store string literals

Note: String literal pool area is actually a special area inside heap memory

Q) How String Literals are stored/placed inside String constant pool area ?

- We will create 3 string literals to understand working?

```
1  // declaration of string literals
2  String str1 = "Hello";
3  String str2 = "World";
4  String str3 = "Hello";
5
6  // string operation to concatenate 2 strings
7  str1.concat(str2);
```



Example1

```
public class StringPoolExample
1. {
2. public static void main(String[] args)
3. {
4. String s1 = "Java";
5. String s2 = "Java";
6. String s3 = new String("Java");
7. String s4 = new String("Java").intern();
8. System.out.println((s1 == s2)+", String are equal."); // true
9. System.out.println((s1 == s3)+", String are not equal."); // false
10. System.out.println((s1 == s4)+", String are equal."); // true
11. }
12. }
```

Example2:

```
public class StringPoolExperiment {

    public static void main(String[] args) {
        String s1 = "Rachel";
        String s2 = "Rachel";
        String s3 = new String("Rachel");
        String s4 = new String("Rachel").intern();

        System.out.println(s1 == s2); // true
        System.out.println(s1 == s3); // false
        System.out.println(s1 == s4); // true
    }
}
```

Example3

```
// Program 1: Comparing two references to objects

// created using literals.

import java.util.*;

class GFG {

public static void main(String[] args)

{

String s1 = "abc";

String s2 = "abc";

// Note that this == compares whether

// s1 and s2 refer to same object or not
```

```
                if (s1 == s2)
                System.out.println("Yes");//yes
                else
                System.out.println("No");
        }
}
```

Example4:

```
// Program 2: Comparing two references to objects
// created using new operator.
import java.util.*;

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

                // Note that this == compares whether
                // s1 and s2 refer to same object or not
                if (s1 == s2)
                System.out.println("Yes");
                else
                System.out.println("No");//no
        }
}
```