**String-2020**

**Example1:**

class Str

{

public static void main(String args[])

{

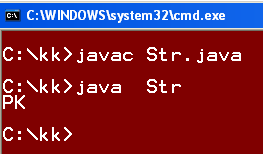
String s="KK";

s="PK";

System.out.println(s);

}

}

****

* **Because variable with S = "KK" , will not refer to KK as it lost its reference when we write S="PK".**

**Example2:**

class Str1

{

public static void main(String args[])

{

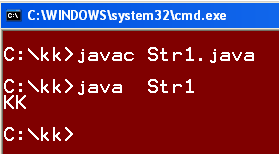
String s="KK";

s.concat(" PK");//concat() method appends the string at the end

System.out.println(s);//will print KK because strings are immutable objects.

}

}



* **It will print KK as because String are immutable Objects.**

**Example3:**

**//Program to demonstrate == operator.**

class Str2

{

public static void main(String args[])

{

String obj1 = new String("xyz");

String obj2 = new String("xyz");

if(obj1 == obj2) // two distinct reference

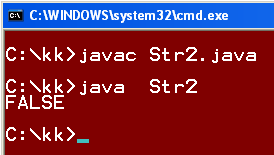
System.out.println("TRUE");

else

System.out.println("FALSE");

}

}



* **It will print FALSE value because Obj1 & Obj2 are referring to two different memory allocation although they have same content.**

**Example4:**

class Str21

{

public static void main(String args[])

{

String str1="KK";

String str2="KK";

if(str1==str2)

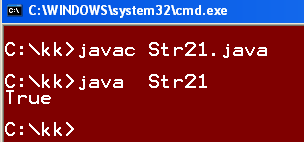
System.out.println("True");

else

System.out.println("False");

}

}



**Note: Both are referring to same memory space.**

**Example5:**

**//Program to demonstrate == operator.**

class Str3

{

public static void main(String args[])

{

String str = new String("xyz");

String str1 = str; // assignment

if(str== str1) // same reference

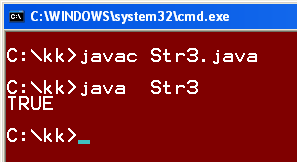
System.out.println("TRUE");

else

System.out.println("FALSE");

}

}



* **It will print TRUE value because Obj1 & Obj2 are referring to the same memory allocation.**

**Note: If we are comparing two Strings Objects with == operator , it will return TRUE only when both String are referring to Same memory allocation otherwise FALSE.**

**Example6:**

class Str31

{

public static void main(String args[])

{

String str = new String("xyz");

String str1 = str;

if(str.equals(str1)) // same reference

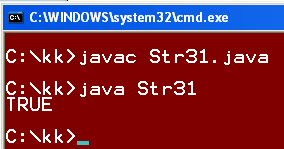
System.out.println("TRUE");

else

System.out.println("FALSE");

}

}

****

**Example7:**

**// Program to demonstrate equals() method of Object Class.**

class Str4

{

public static void main(String args[])

{

String str1 = new String("xyz");

String str2 = new String("xyz");

if(str1.equals(str2))

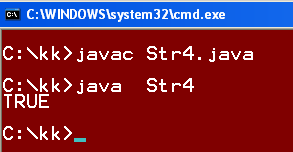
System.out.println("TRUE");

else

System.out.println("FALSE");

}

}



* **It will print TRUE if the contents are same of two Objects otherwise it will print FALSE.**

**Example8:**

**// Program to demonstrate equals() method of Object Class.**

class Str5

{

public static void main(String args[])

{

String str1 = new String("XYZ");

String str2 = new String("PQR");

if(str1.equals(str2))

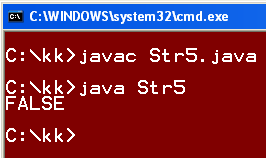
System.out.println("TRUE");

else

System.out.println("FALSE");

}

}



**Note: If we are comparing two Strings Objects with equals() method of Object Class, it will return TRUE only if the contents of both the String objects are Same otherwise it will return FALSE.**

**Example9:**

class Str6

{

public static void main(String args[])

{

String str = "Name";

String str1 = new String("Name");

if(str.equals(str1))

System.out.println("true");

else

System.out.println("false");

if(str==str1)

System.out.println("true");

else

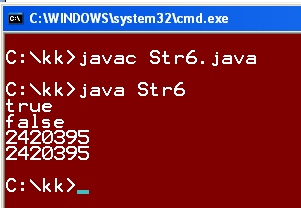
System.out.println("false");

System.out.println(str.hashCode());

System.out.println(str1.hashCode());

}

}



**Note: equals() method compares the String contents and "==" operator compares the reference. But in output the hashCode() method prints the reference number for both strings as same then why the "==" returns "false". They have same hasCode values as because they are referring to same memory allocation.**