Idris Ratlamwala
C31 - 2003145

Experiment 13: Program on String

Theory :

String is a sequence of characters, for e.g. “Hello” is a string of 5 characters. In java, string is an immutable object which means it is constant and can cannot be changed once it has been created.

Creating a String
There are two ways to create a String in Java

1. String literal
2. Using new keyword

String literal
In java, Strings can be created like this: Assigning a String literal to a String instance:
String str1 = "Welcome";
String str2 = "Welcome";

The problem with this approach: String is an object in Java. However we have not created any string object using new keyword above. The compiler does that task for us it creates a string object having the string literal (that we have provided , in this case it is “Welcome”) and assigns it to the provided string instances.

But if the object already exist in the memory it does not create a new Object rather it assigns the same old object to the new instance, that means even though we have two string instances above(str1 and str2) compiler only created on string object (having the value “Welcome”) and assigned the same to both the instances. For example there are 10 string instances that have same value, it means that in memory there is only one object having the value and all the 10 string instances would be pointing to the same object.

Using New Keyword
As we saw above that when we tried to assign the same string object to two different literals, compiler only created one object and made both of the literals to point the same object. To overcome that approach we can create strings like this:
String str1 = new String("Welcome");
String str2 = new String("Welcome");
In this case compiler would create two different object in memory having the same string.

StringBuffer class is used to create a **mutable** string object. It means, it can be changed after it is created. It represents growable and writable character sequence.

It is similar to String class in Java both are used to create string, but stringbuffer object can be changed.

So **StringBuffer** class is used when we have to make lot of modifications to our string. It is also thread safe i.e multiple threads cannot access it simultaneously. StringBuffer defines 4 constructors.

1. **StringBuffer**(): It creates an empty string buffer and reserves space for 16 characters.
2. **StringBuffer**(int size): It creates an empty string and takes an integer argument to set capacity of the buffer.
3. **StringBuffer**(String str): It creates a stringbuffer object from the specified string.
4. **StringBuffer**(charSequence []ch): It creates a stringbuffer object from the charsequence array.

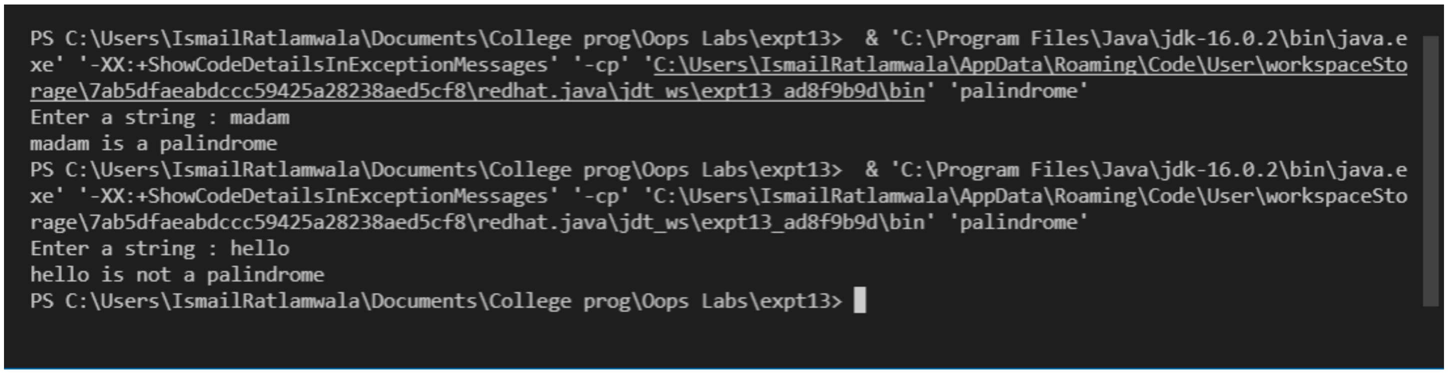
A.

Aim : WAP to check if a string is a palindrome .

Program :

```java
import java.util.Scanner;

public class palindrome{
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);

        System.out.print("Enter a string : ");
        String inp = sc.next();
        String reversedString = reverseString(inp);

        if (inp.equals(reversedString))  System.out.println(inp + " is a palindrome");
        else  System.out.println(inp + " is not a palindrome");
    }

    private static String reverseString(String inp) {
        StringBuffer buffer = new StringBuffer();
        buffer.append(inp);
        return buffer.reverse().toString();
    }
}
```

Output :



```
PS C:\Users\IsmailRatlamwala\Documents\College prog\Oops Labs\expt13>  & 'C:\Program Files\Java\jdk-16.0.2\bin\java.e
xe' '-XX:+ShowCodeDetailsInExceptionMessages' '-cp' 'C:\Users\IsmailRatlamwala\AppData\Roaming\Code\User\workspaceSto
rage\7ab5dfaeabdccc59425a28238aed5cf8\redhat.java\jdt_ws\expt13_ad8f9b9d\bin' 'palindrome'
Enter a string : madam
madam is a palindrome
PS C:\Users\IsmailRatlamwala\Documents\College prog\Oops Labs\expt13>  & 'C:\Program Files\Java\jdk-16.0.2\bin\java.e
xe' '-XX:+ShowCodeDetailsInExceptionMessages' '-cp' 'C:\Users\IsmailRatlamwala\AppData\Roaming\Code\User\workspaceSto
rage\7ab5dfaeabdccc59425a28238aed5cf8\redhat.java\jdt_ws\expt13_ad8f9b9d\bin' 'palindrome'
Enter a string : hello
hello is not a palindrome
PS C:\Users\IsmailRatlamwala\Documents\College prog\Oops Labs\expt13> 
```

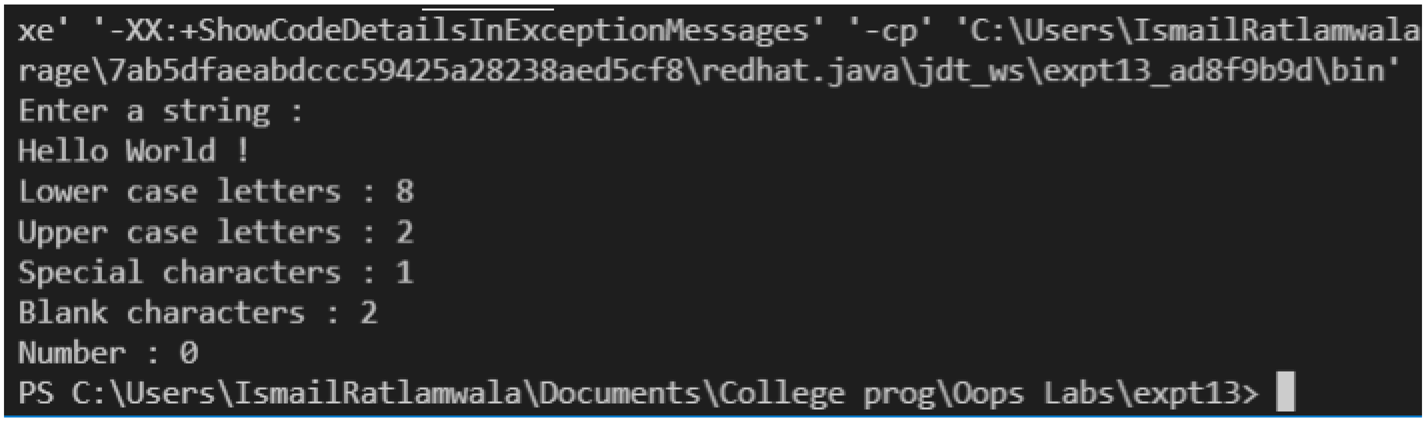
B.

Aim : WAP to accept a string from user and display the number of uppercase, lowercase, special characters, blank spaces & digits present in the accepted string.

Program :

```
import java.io.*;
import java.util.Scanner;

class Count
{
    public static void main(String args[])
    {
        Scanner sc = new Scanner(System.in);
        System.out.println("Enter a string :");
        String str = sc.nextLine();
        int upper = 0, lower = 0, number = 0, special = 0, blank = 0;

        for(int i = 0; i < str.length(); i++)
        {
            char ch = str.charAt(i);
            if (ch >= 'A' && ch <= 'Z')
                upper++;
            else if (ch >= 'a' && ch <= 'z')
                lower++;
            else if (ch==' ')
                blank++;
            else if (ch >= '0' && ch <= '9')
                number++;
            else
                special++;
        }

        System.out.println("Lower case letters : " + lower);
        System.out.println("Upper case letters : " + upper);
        System.out.println("Special characters : " + special);
        System.out.println("Blank characters : " + blank);
        System.out.println("Number : " + number);
    }
}
```

Output :



```
xe' '-XX:+ShowCodeDetailsInExceptionMessages' '-cp' 'C:\Users\IsmailRatlamwala
rage\7ab5dfaeabdccc59425a28238aed5cf8\redhat.java\jdt_ws\expt13_ad8f9b9d\bin'
Enter a string :
Hello World !
Lower case letters : 8
Upper case letters : 2
Special characters : 1
Blank characters : 2
Number : 0
PS C:\Users\IsmailRatlamwala\Documents\College prog\Oops Labs\expt13>
```