# PW SKILLS

## JAVA With DSA & System Design

### Assignment – Strings in Java (Part - 3)

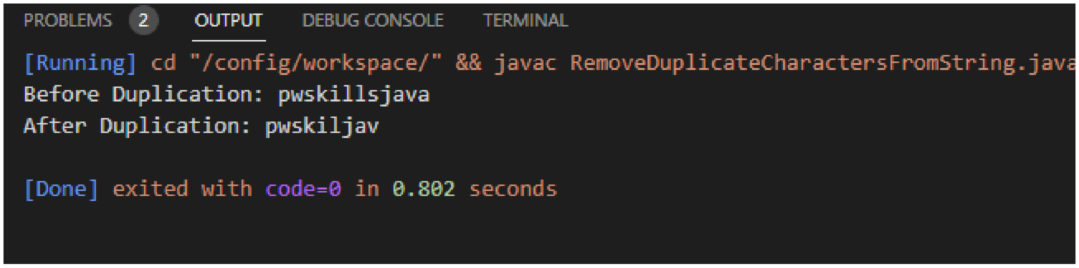
### Day - 16

1. **WAP (Write a program) to remove Duplicate from a String. (Take any String with Duplicates character).**

   **Ans: Program to remove Duplicate from a String**

```
class RemoveDuplicateCharactersFromString {
public static void main(String[] args) {
    String str1 ="Pw Skills Java";
    String  str = str1.replace(" ", "");
    str= str.toLowerCase();
    String  res =""+str.charAt(0);

        for(int i=0;i<str.length();i++)
        {
            if(!res.contains(String.valueOf(str.charAt(i))))
            {
            res = res+str.charAt(i);
            }
        }
    System.err.println("Before Duplication: " + str);
    System.out.println("After Duplication: " + res);
    }
}
```

**Output:**



```
[Running] cd "/config/workspace/" && javac RemoveDuplicateCharactersFromString.java
Before Duplication: pwskillsjava
After Duplication: pwskiljav

[Done] exited with code=0 in 0.802 seconds
```

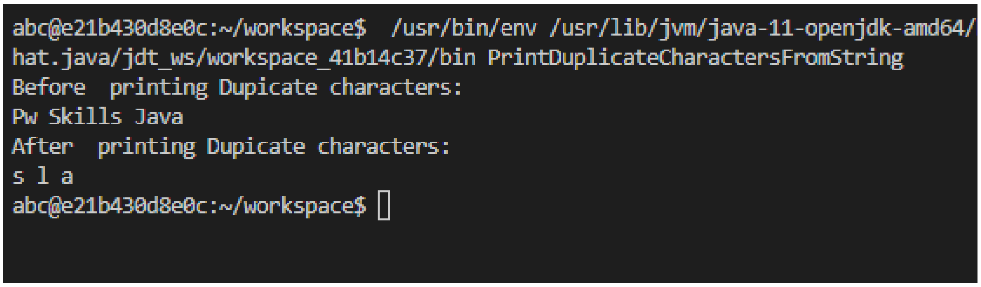
2. **WAP to print Duplicates characters from the String.**

   **Ans: Program to print duplicate characters from the string.**

```
import java.util.Scanner;
public class PrintDuplicateCharactersFromString {
    public static void main(String[] args) {
    System.out.println("Before  printing Dupicate characters: ");
    Scanner sc = new Scanner(System.in);
    String str1 = sc.nextLine();

    String  str = str1.replace(" ", "");
    str= str.toLowerCase();
    char arr[]=str.toCharArray();
    System.out.println("After  printing Dupicate characters: ");
    for(int i=0;i<arr.length;i++)
    {
```

```
                int count=1;
                for(int j=i+1;j<arr.length;j++)
                        {
                        if(arr[i]==arr[j] && arr[i]!=' ')
                        {
                        count++;
                        arr[j]='0';
                        }
        }
                if(count >1 && arr[i]!='0')
                {
                System.out.print(arr[i]+" ");
                }
                }
                        System.out.println();
                }
        }
```

**Output:**



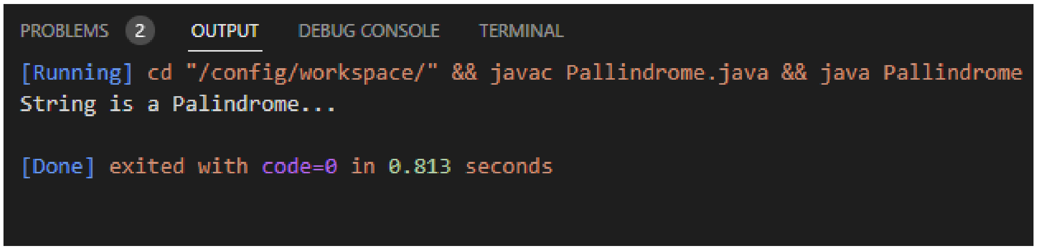
```
abc@e21b430d8e0c:~/workspace$  /usr/bin/env /usr/lib/jvm/java-11-openjdk-amd64/
hat.java/jdt_ws/workspace_41b14c37/bin PrintDuplicateCharactersFromString
Before  printing Dupicate characters:
Pw Skills Java
After  printing Dupicate characters:
s l a
abc@e21b430d8e0c:~/workspace$ 
```

3. **WAP to check if “2552” is palindrome or not.**
   **Ans: Program to check if “2552” is palindrome or not.**

```
        public class Pallindrome {
                public static void main(String[] args) {
                        String str1 = "2552";
                        String str2 = "";

                        for(int i=str1.length()-1;i>=0; i--)
                        {
                                str2=str2+str1.charAt(i);
                        }

                        if(str1.equals(str2)){
                                System.out.println("String is a Palindrome...");
                        }
                        else
                        {
```

```
                        System.out.println("String is not a Palindrome");
                }
        }
}
```

**Output:**



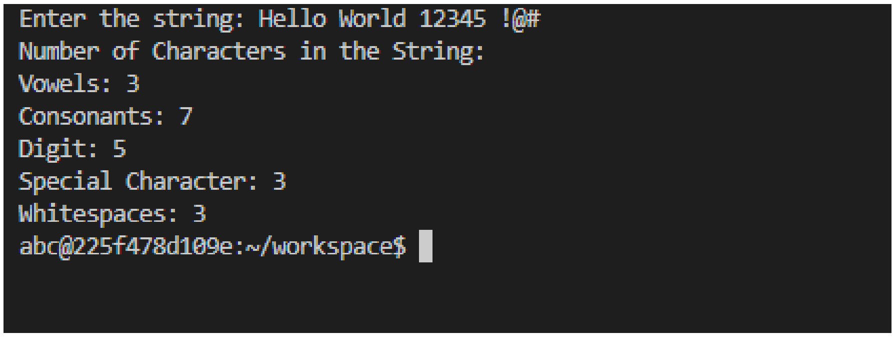
```
[Running] cd "/config/workspace/" && javac Pallindrome.java && java Pallindrome
String is a Palindrome...

[Done] exited with code=0 in 0.813 seconds
```

4. **WAP to count the number of consonants, vowels, special characters in a String.**
   **Ans: Program to count the number of consonants, vowels, special characters in a String:**

```
import java.io.*;
import java.util.Scanner;
public class WAP_TO_Count_NoOf_Consonants_Vowels_Special_Characters_InA_String {
   static void countCharacterType(String str)
   {
      //Declaration of variables, constants, digits, and special characters
      int vowels=0, constants=0, specialChar=0, digit=0, whitespace=0;
      for(int i=0; i<str.length(); i++)
      {
         char ch= str.charAt(i);
         if( (ch>='a'&& ch<='z') || (ch >= 'A' && ch <='Z'))
         {
            //to handle upper case letters
            ch = Character.toLowerCase(ch);
            if( ch == 'a' || ch == 'e' || ch=='i' || ch== 'o' || ch== 'u')
            {
               vowels++;
            }
            else
            {
               constants++;
            }

         }
         else if(ch>='0' && ch<='9')
            {
               digit++;
            }
         else if(ch==' ')
            {
              whitespace++;
```

```
                }
                else
                {
                   specialChar++;
                }
            }
            System.out.println("Vowels: "+vowels);
            System.out.println("Consonants: "+constants);
            System.out.println("Digit: "+digit);
            System.out.println("Special Character: "+specialChar);
            System.out.println("Whitespaces: "+whitespace);
        }
           public static void main(String[] args) {
            System.out.print("Enter the string: ");
            Scanner sc = new Scanner(System.in);
            String str = sc.nextLine();
            System.out.println("Number of Characters in the String: ");
            countCharacterType(str);
        }
    }
```

**Output:**

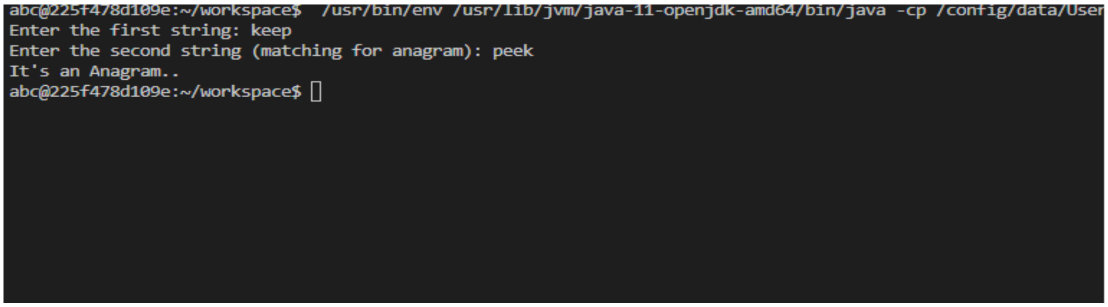


```
Enter the string: Hello World 12345 !@#
Number of Characters in the String:
Vowels: 3
Consonants: 7
Digit: 5
Special Character: 3
Whitespaces: 3
abc@225f478d109e:~/workspace$ 
```

5. **WAP to implement Anagram checking least inbuilt methods being used.**
   **Ans: Program to implement Anagram least inbuild method being used in it:**

```
import java.util.Arrays;
import java.util.Scanner;

public class Anagram {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter the first string: ");
        String str1= sc.nextLine();
        System.out.print("Enter the second string (matching for anagram): ");
        String str2=sc.nextLine();

        //To remove whitespace
        str1=str1.replace(" ","");
        str2=str2.replace(" ","");

        //TO convert into lower case
        str1=str1.toLowerCase();
        str2=str2.toLowerCase();

        //to convert into char array
        char arr1[]=str1.toCharArray();
        char arr2[]=str2.toCharArray();

        //to sort the arrays
        Arrays.sort(arr1);
        Arrays.sort(arr2);

        //checks if the given string are anagram or not
        if(Arrays.equals(arr1, arr2)){
            System.out.println("It's an Anagram..");
        }
        else{
            System.out.println("It's not a anagram..");
        }
    }
}
```

**Output:**



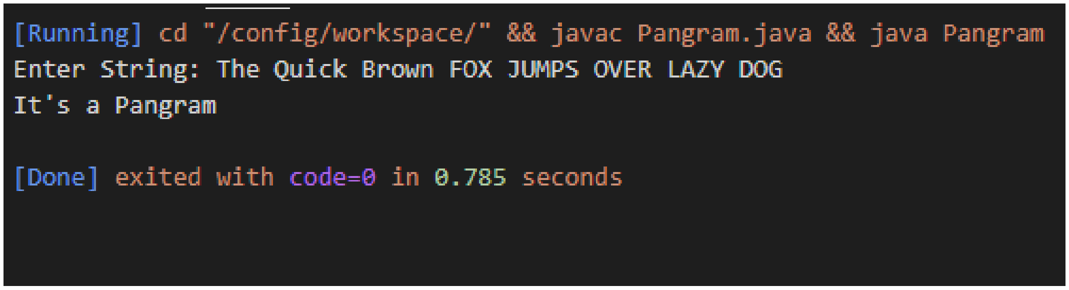
```
abc@225f478d109e:~/workspace$  /usr/bin/env /usr/lib/jvm/java-11-openjdk-amd64/bin/java -cp /config/data/User
Enter the first string: keep
Enter the second string (matching for anagram): peek
It's an Anagram..
abc@225f478d109e:~/workspace$ 
```

6. **WAP to implement Pangram checking with least inbuilt methods being used.**
   **Ans: Program to implement Pangram checking with least inbuilt methods being used in it:**

```
public class Pangram {
  public static void main(String[] args) {
    boolean flag = false;
    String str = "The Quick Brown FOX JUMPS OVER LAZY DOG";
    System.out.println("Enter String: "+str);
    str=str.replace(" ", "");

    str=str.toUpperCase();

    char ch[]= str.toCharArray();
```

```
        int arr[] = new int[26];
        for(int i=0; i<ch.length;i++)
        {
           arr[ch[i]-65]++;
        }

        for(int i=0;i<arr.length;i++)
        {
           if(arr[i]==0)
           {

              flag =true;
           }
        }

        if(flag==true)
        {
           System.out.println("It's not a Pangram");
        }
        else{
           System.out.println("It's a Pangram");
        }
    }
}
```

**Output:**



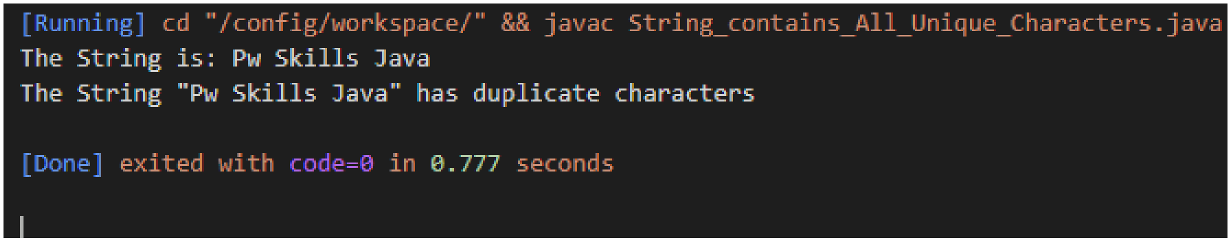
```
[Running] cd "/config/workspace/" && javac Pangram.java && java Pangram
Enter String: The Quick Brown FOX JUMPS OVER LAZY DOG
It's a Pangram

[Done] exited with code=0 in 0.785 seconds
```

7. **WAP to find if String contains all unique characters.**
   **Ans: Program to implement a string contains unique characters:**

```
public class String_contains_All_Unique_Characters {
public static void main(String[] args) {
String str = "Pw Skills Java";
boolean flag=true;

System.out.println("The String is: "+str);

for(int i=0;i<str.length();i++)
{
```

```
for(int j=i+1;j<str.length();j++)
{
if(str.charAt(i) == str.charAt(j))
{

flag = false;
}
}
}
if(flag==false)
{
System.out.println("The String " +'"' + str + '"'+ " has duplicate characters");
}
else
{
System.out.println("The String " + '"' + str + '"' + " has all unique characters");
}
}
}
```

**Output:**

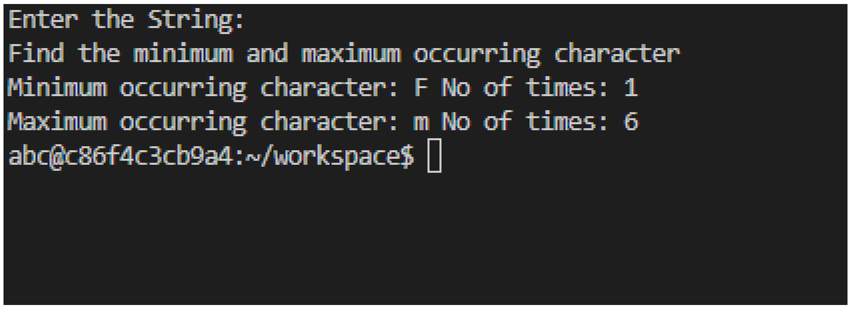


```
[Running] cd "/config/workspace/" && javac String_contains_All_Unique_Characters.java
The String is: Pw Skills Java
The String "Pw Skills Java" has duplicate characters

[Done] exited with code=0 in 0.777 seconds
```

8. **WAP to find the maximum occurring character in a String.**
   **Ans: Program to print maximum occurring character in a String:**

```
import java.util.Scanner;
public class Maximum_occurring_character_in_String {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.println("Enter the String: ");
        String str = sc.nextLine();

        //array of frequency
        int freq[]= new int[str.length()];

        //max and min char var
        int min,max;
        char minChar = str.charAt(0);
        char maxChar = str.charAt(0);

        //str to string array, i.e., char array
        char string[] = str.toCharArray();

        for(int i=0;i<str.length();i++)
        {
            freq[i]=1;
            for(int j=i+1;j<str.length();j++)
            {
                if(string[i] == string[j] && string[i] != ' ' && string[i] != '0')
                {
                    freq[i]++;
                    string[j]='0';
                }
            }
        }
        min= max =freq[0];
        for(int i=0; i<freq.length;i++)
        {
            if(min > freq[i] && freq[i] != '0')
            {
                min = freq[i];
                minChar = string[i];
            }

            if(max<freq[i])
            {
                max = freq[i];
                maxChar = string[i];
            }
        }
        System.out.println("Minimum occurring character: " + minChar + "  No of times: "+ min);
        System.out.println("Maximum occurring character: " + maxChar + "  No of times: "+ max);
    }
}
```

**Output:**



```
Enter the String:
Find the minimum and maximum occurring character
Minimum occurring character: F No of times: 1
Maximum occurring character: m No of times: 6
abc@c86f4c3cb9a4:~/workspace$ 
```