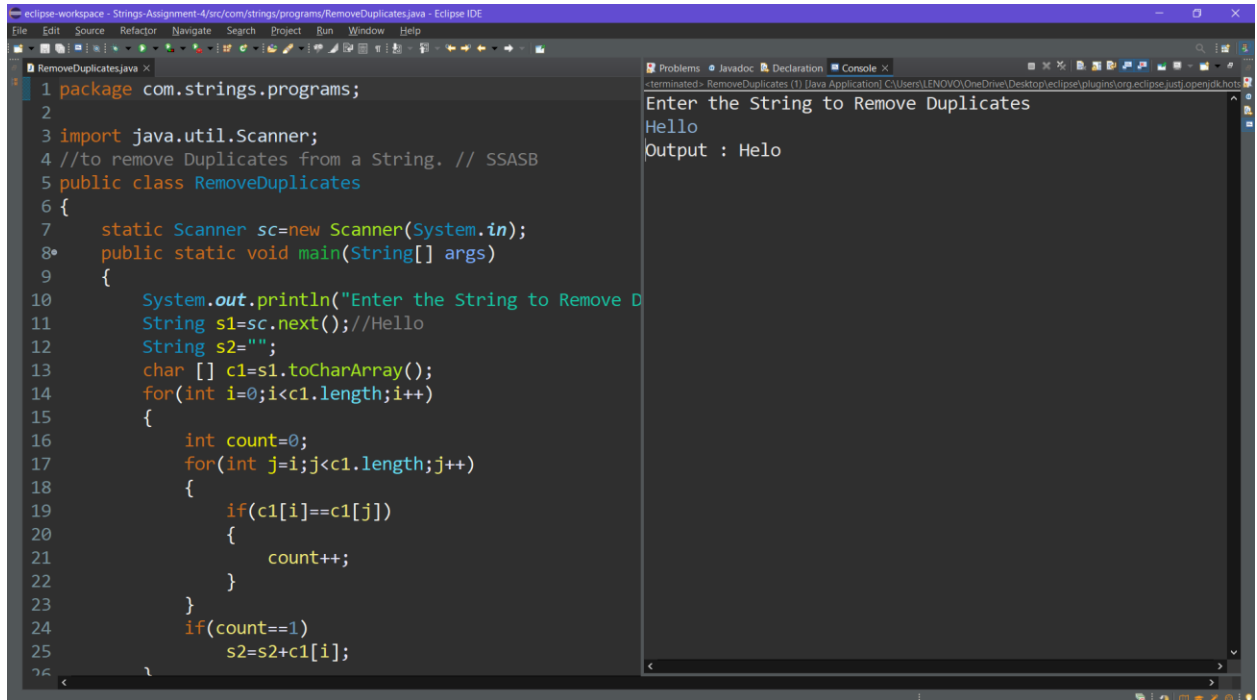


Assignment 4 Outputs

1. WAP to remove Duplicates from a String.



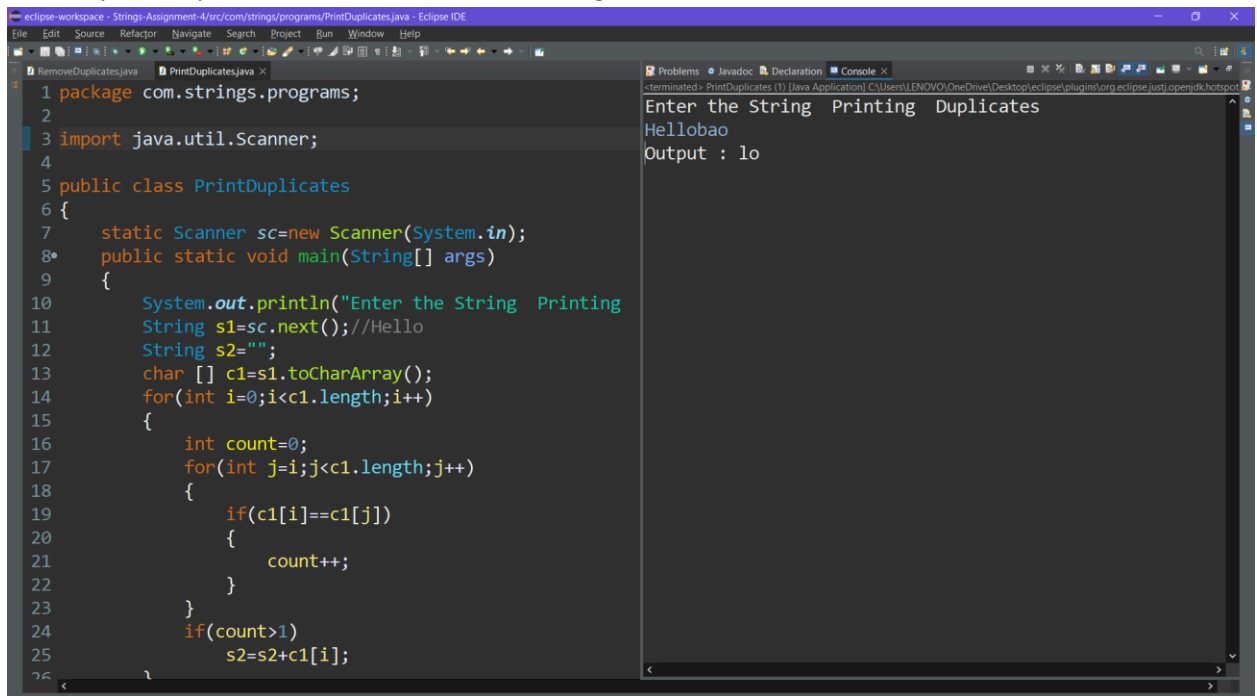
The screenshot shows the Eclipse IDE with a Java file named `RemoveDuplicates.java`. The code defines a class `RemoveDuplicates` with a `main` method that uses a `Scanner` to take input from the user. The input is "Hello", and the output is "Helo". The console output shows the prompt "Enter the String to Remove Duplicates", the input "Hello", and the output "Output : Helo".

```
1 package com.strings.programs;
2
3 import java.util.Scanner;
4 //to remove Duplicates from a String. // SSASB
5 public class RemoveDuplicates
6 {
7     static Scanner sc=new Scanner(System.in);
8     public static void main(String[] args)
9     {
10         System.out.println("Enter the String to Remove Duplicates");
11         String s1=sc.next();//Hello
12         String s2="";
13         char [] c1=s1.toCharArray();
14         for(int i=0;i<c1.length;i++)
15         {
16             int count=0;
17             for(int j=i;j<c1.length;j++)
18             {
19                 if(c1[i]==c1[j])
20                 {
21                     count++;
22                 }
23             }
24             if(count==1)
25                 s2=s2+c1[i];
26         }
27     }
28 }
```

Console Output:

```
Enter the String to Remove Duplicates
Hello
Output : Helo
```

2. WAP to print Duplicates characters from the String.



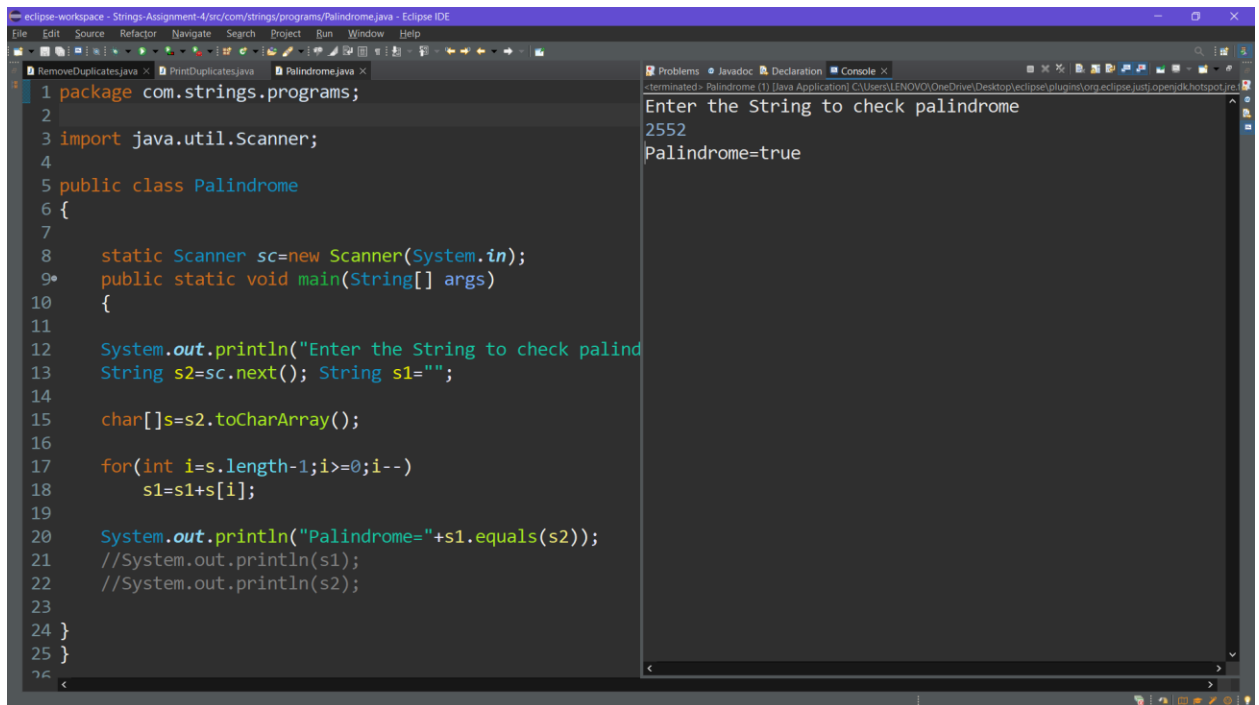
The screenshot shows the Eclipse IDE with a Java file named `PrintDuplicates.java`. The code defines a class `PrintDuplicates` with a `main` method that uses a `Scanner` to take input from the user. The input is "Hello", and the output is "Helo". The console output shows the prompt "Enter the String Printing Duplicates", the input "Hello", and the output "Output : 10".

```
1 package com.strings.programs;
2
3 import java.util.Scanner;
4
5 public class PrintDuplicates
6 {
7     static Scanner sc=new Scanner(System.in);
8     public static void main(String[] args)
9     {
10         System.out.println("Enter the String Printing Duplicates");
11         String s1=sc.next();//Hello
12         String s2="";
13         char [] c1=s1.toCharArray();
14         for(int i=0;i<c1.length;i++)
15         {
16             int count=0;
17             for(int j=i;j<c1.length;j++)
18             {
19                 if(c1[i]==c1[j])
20                 {
21                     count++;
22                 }
23             }
24             if(count>1)
25                 s2=s2+c1[i];
26         }
27     }
28 }
```

Console Output:

```
Enter the String Printing Duplicates
Hello
Output : 10
```

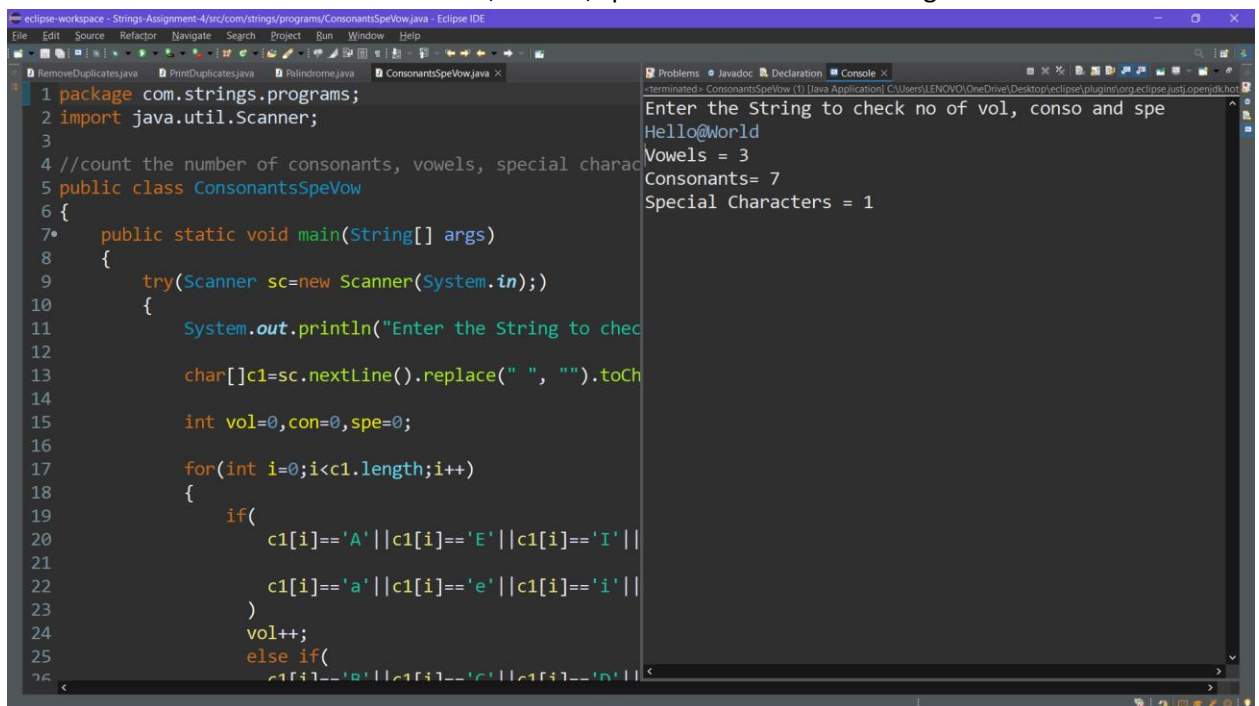
3. WAP to check if "2552" is palindrome or not.



```
1 package com.strings.programs;
2
3 import java.util.Scanner;
4
5 public class Palindrome
6 {
7
8     static Scanner sc=new Scanner(System.in);
9     public static void main(String[] args)
10    {
11
12        System.out.println("Enter the String to check palindrome");
13        String s2=sc.next(); String s1="";
14
15        char[]s=s2.toCharArray();
16
17        for(int i=s.length-1;i>=0;i--)
18            s1=s1+s[i];
19
20        System.out.println("Palindrome="+s1.equals(s2));
21        //System.out.println(s1);
22        //System.out.println(s2);
23
24    }
25 }
```

Enter the String to check palindrome
2552
Palindrome=true

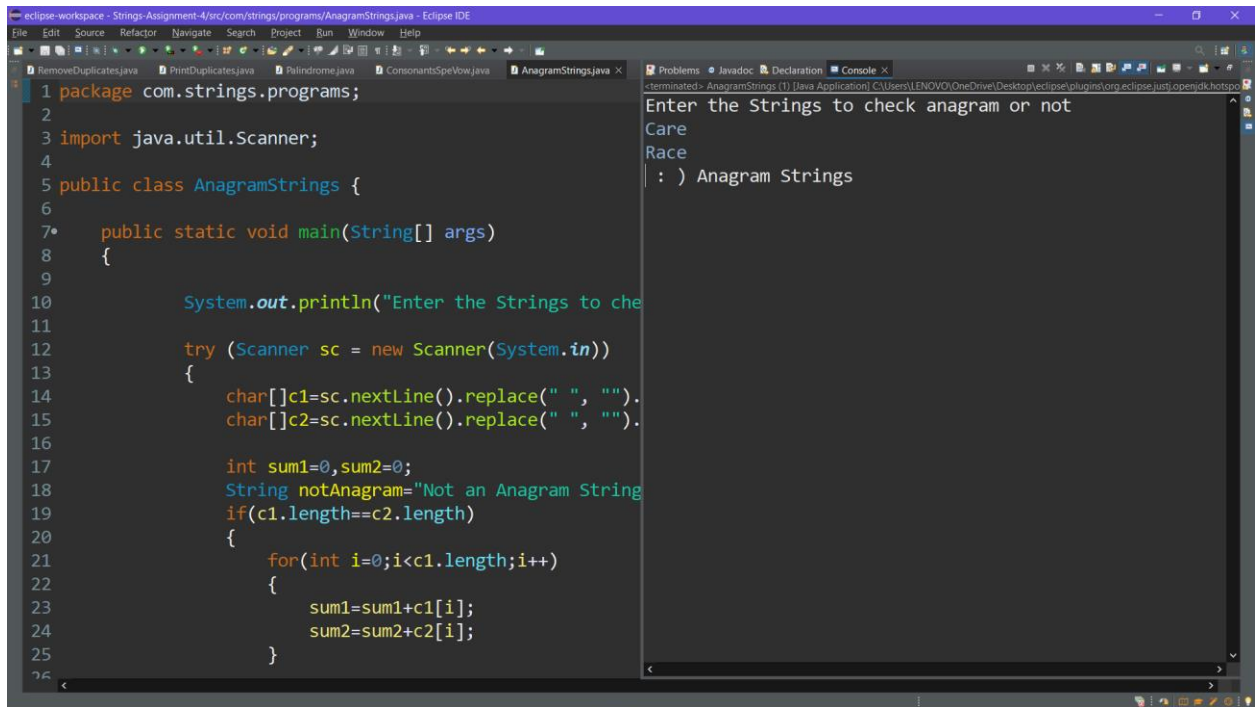
4. WAP to count the number of consonants, vowels, special characters in a String.



```
1 package com.strings.programs;
2 import java.util.Scanner;
3
4 //count the number of consonants, vowels, special characters
5 public class ConsonantsSpeVow
6 {
7     public static void main(String[] args)
8     {
9         try(Scanner sc=new Scanner(System.in));
10        {
11            System.out.println("Enter the String to check no of vol, conso and spe");
12
13            char[]c1=sc.nextLine().replace(" ", "").toLowerCase().toCharArray();
14
15            int vol=0,con=0,spe=0;
16
17            for(int i=0;i<c1.length;i++)
18            {
19                if(
20                    c1[i]=='A' || c1[i]=='E' || c1[i]=='I' ||
21                    c1[i]=='a' || c1[i]=='e' || c1[i]=='i' ||
22                )
23                    vol++;
24                else if(
25                    c1[i]=='B' || c1[i]=='C' || c1[i]=='D' ||
26                    c1[i]=='b' || c1[i]=='c' || c1[i]=='d' ||
27                    c1[i]=='F' || c1[i]=='G' || c1[i]=='H' ||
28                    c1[i]=='f' || c1[i]=='g' || c1[i]=='h' ||
29                    c1[i]=='J' || c1[i]=='K' || c1[i]=='L' ||
30                    c1[i]=='j' || c1[i]=='k' || c1[i]=='l' ||
31                    c1[i]=='M' || c1[i]=='N' || c1[i]=='O' ||
32                    c1[i]=='m' || c1[i]=='n' || c1[i]=='o' ||
33                    c1[i]=='P' || c1[i]=='Q' || c1[i]=='R' ||
34                    c1[i]=='p' || c1[i]=='q' || c1[i]=='r' ||
35                    c1[i]=='S' || c1[i]=='T' || c1[i]=='U' ||
36                    c1[i]=='s' || c1[i]=='t' || c1[i]=='u' ||
37                    c1[i]=='V' || c1[i]=='W' || c1[i]=='X' ||
38                    c1[i]=='v' || c1[i]=='w' || c1[i]=='x' ||
39                    c1[i]=='Y' || c1[i]=='Z' || c1[i]=='y' || c1[i]=='z' ||
40                    c1[i]=='_' || c1[i]=='-' || c1[i]=='_' ||
41                    c1[i]=='_' || c1[i]=='-' || c1[i]=='_' ||
42                )
43                    con++;
44                else
45                    spe++;
46            }
47        }
48    }
```

Enter the String to check no of vol, conso and spe
Hello@World
Vowels = 3
Consonants = 7
Special Characters = 1

5. WAP to implement Anagram Checking least inbuilt methods being used.

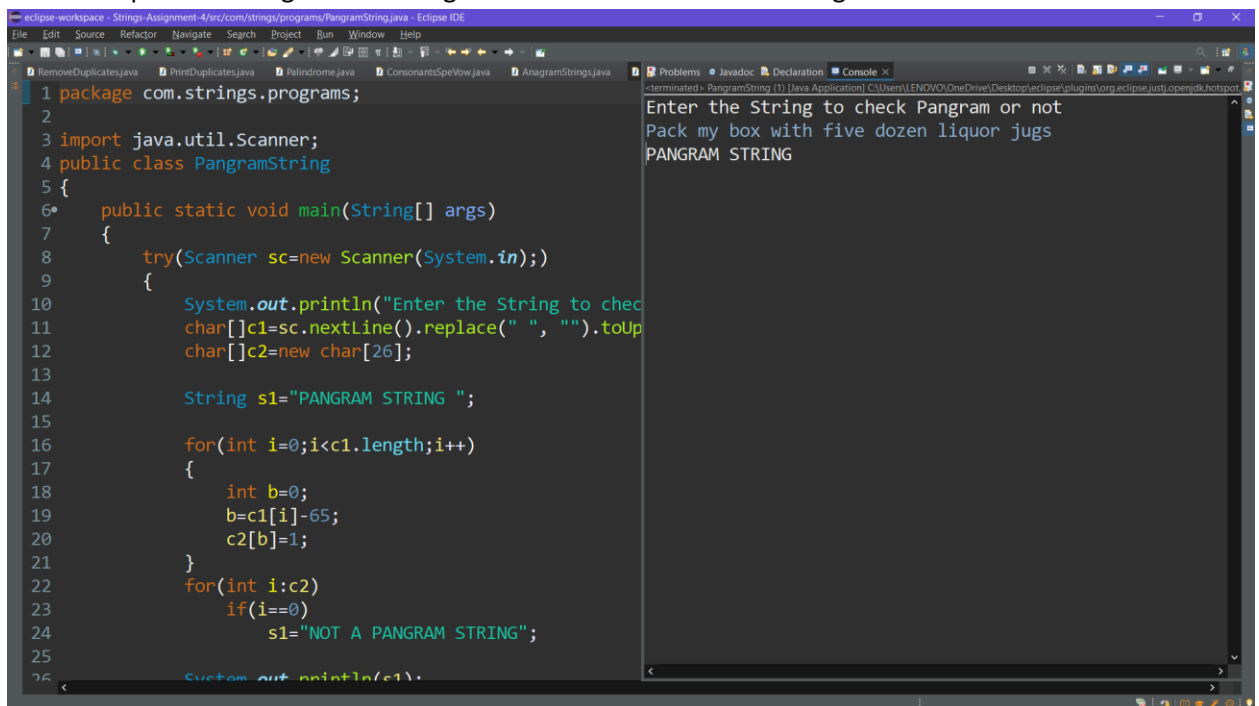


The screenshot shows the Eclipse IDE with a Java project named 'Strings-Assignment-4'. The main editor displays the file 'AnagramStrings.java'. The code defines a package 'com.strings.programs', imports 'java.util.Scanner', and defines a public class 'AnagramStrings' with a 'main' method. The 'main' method prompts the user to enter two strings, removes spaces, and checks if they are anagrams by comparing the sum of character ASCII values. The console on the right shows the program's execution, displaying the prompt 'Enter the Strings to check anagram or not' and the user input 'Care Race', followed by the output ':) Anagram Strings'.

```
1 package com.strings.programs;
2
3 import java.util.Scanner;
4
5 public class AnagramStrings {
6
7     public static void main(String[] args)
8     {
9
10        System.out.println("Enter the Strings to che
11
12        try (Scanner sc = new Scanner(System.in))
13        {
14            char[] c1=sc.nextLine().replace(" ", "");
15            char[] c2=sc.nextLine().replace(" ", "");
16
17            int sum1=0,sum2=0;
18            String notAnagram="Not an Anagram String
19            if(c1.length==c2.length)
20            {
21                for(int i=0;i<c1.length;i++)
22                {
23                    sum1=sum1+c1[i];
24                    sum2=sum2+c2[i];
25                }
26            }
27        }
28    }
29 }
```

Enter the Strings to check anagram or not
Care
Race
:) Anagram Strings

6. WAP to implement Pangram Checking with least inbuilt methods being used.



The screenshot shows the Eclipse IDE with a Java project named 'Strings-Assignment-4'. The main editor displays the file 'PangramString.java'. The code defines a package 'com.strings.programs', imports 'java.util.Scanner', and defines a public class 'PangramString' with a 'main' method. The 'main' method prompts the user to enter a string, removes spaces, and checks if it is a pangram by comparing the frequency of each character in the input string to a predefined string 'PANGRAM STRING'. The console on the right shows the program's execution, displaying the prompt 'Enter the String to check Pangram or not' and the user input 'Pack my box with five dozen liquor jugs', followed by the output 'PANGRAM STRING'.

```
1 package com.strings.programs;
2
3 import java.util.Scanner;
4 public class PangramString
5 {
6     public static void main(String[] args)
7     {
8         try(Scanner sc=new Scanner(System.in));)
9         {
10            System.out.println("Enter the String to check
11            char[] c1=sc.nextLine().replace(" ", "").toUp
12            char[] c2=new char[26];
13
14            String s1="PANGRAM STRING ";
15
16            for(int i=0;i<c1.length;i++)
17            {
18                int b=0;
19                b=c1[i]-65;
20                c2[b]=1;
21            }
22            for(int i:c2)
23                if(i==0)
24                    s1="NOT A PANGRAM STRING";
25
26            System.out.println(c1);
27        }
28    }
29 }
```

Enter the String to check Pangram or not
Pack my box with five dozen liquor jugs
PANGRAM STRING

7. WAP to find if String contains all unique characters.

The screenshot shows the Eclipse IDE with a Java file named `CheckStringContainUnique.java`. The code is as follows:

```
1 package com.strings.programs;
2
3 import java.util.Scanner;
4
5 public class CheckStringContainUnique
6 {
7     public static void main(String[] args)
8     {
9         try(Scanner sc=new Scanner(System.in));
10        {
11            System.out.println("Enter the Strings find all Unique Characters or not");
12
13            char[] s=sc.nextLine().replace(" ", "").toLowerCase();
14
15            String s1=":)Above String Contain All Unique Characters";
16
17            for(int i=0;i<s.length;i++)
18            {
19                int c=0;
20                for(int j=0;j<s.length;j++)
21                {
22                    if(s[i]==s[j])
23                        c=c+1;
24                }
25                if(c!=1)
26                    s1="(: Above String Not Contain All Unique Characters";
27            }
28        }
29    }
30 }
```

The console output shows the program execution with the input string "Heloabc" and the resulting message: " :)Above String Contain All Unique Characters".

The screenshot shows the Eclipse IDE with a Java file named `CheckStringContainUnique.java`. The code is as follows:

```
1 package Programs;
2
3 import java.util.Scanner;
4
5 public class CheckStringContainUnique
6 {
7     public static void main(String[] args)
8     {
9
10        System.out.println("Enter the Strings find all Unique Characters or not");
11
12        Scanner sc=new Scanner(System.in);
13        String s2=sc.next();
14        String s1=":)Above String Contain All Unique Characters";
15        char[] s=s2.toCharArray();
16        for(int i=0;i<s.length;i++)
17        {
18            int c=0;
19            for(int j=0;j<s.length;j++)
20            {
21                if(s[i]==s[j])
22                    c=c+1;
23            }
24            if(c!=1)
25                s1="(: Above String Not Contain All Unique Characters";
26        }
27    }
28 }
```

The console output shows the program execution with the input string "ABCDEF" and the resulting message: " :)Above String Contain All Unique Characters".

8. WAP to find the maximum occurring character in a String

```
1 package Programs;
2
3 import java.util.Scanner;
4
5 public class MaximumCharCount
6 {
7     public static void main(String[] args)
8     {
9
10        System.out.println("Enter the Strings find Maximum occurrence char");
11        Scanner sc=new Scanner(System.in);
12        String s2=sc.next();
13        char[] s=s2.toCharArray();int max=0;String s
14
15        for(int i=0;i<s.length;i++)
16        {
17            int c=0;
18            for(int j=0;j<s.length;j++)
19            {
20                if(s[i]==s[j])
21                    c=c+1;
22            }
23            if(c>max)
24            {
25                max=c;
26            }
27        }
28    }
29 }
```

Enter the Strings find Maximum occurrence char
HelloWorld
Character is :l
count is :3