# PRACTICAL-1

**Given a string and a non-negative int n, we'll say that the front of the string is the first 3 chars, or whatever is there if the string is less than length 3. Return n copies of the front; front\_times('Chocolate', 2) → 'ChoCho' front\_times('Chocolate', 3) → 'ChoChoCho' front\_times('Abc', 3) → 'AbcAbcAbc'**

**Program Code:**

class SP\_6

{

public static void main(String args[])

{

SP\_6 obj=new SP\_6();

String s1=obj.front\_lines("Chocolate",2);

String s2=obj.front\_lines("Chocolate",3);

String s3=obj.front\_lines("ABC",3);

System.out.println(s1);

System.out.println(s2); System.out.println(s3);

}

String front\_lines(String s, int no)

{

String s1=""; for(int i=1;i<=no;i++)

{

s1=s1+s.substring(0,3);

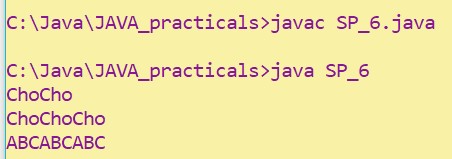
}

return s1;

}

}

**Output:**



# PRACTICAL-2

**Given an array of ints, return the number of 9's in the array.**

**array\_count9([1, 2, 9]) → 1 array\_count9([1, 9, 9]) → 2 array\_count9([1, 9, 9, 3, 9]) → 3**

**Program Code:**

class SP\_7

{

public static void main(String[] args)

{

SP\_7 obj=new SP\_7(); int x1[]={1,2,9}; int x2[]={1,9,9}; int x3[]={1,9,9,3,9}; int n1=obj.array\_count(x1); int n2=obj.array\_count(x2); int n3=obj.array\_count(x3); System.out.println(n1);

System.out.println(n2); System.out.println(n3);

}

int array\_count(int[] arr)

{

int cnt=0;

for(int i=0;i<arr.length;i++)

{

if(arr[i]==9)

cnt++;

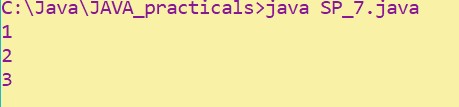
}

return cnt;

}

}

**OUTPUT:**



# PRACTICAL-3

**Given an array of ints, return True if one of the first 4 elements in the array is a 9.**

**The array length may be less than 4.**

**array\_front9([1, 2, 9, 3, 4]) → True array\_front9([1, 2, 3, 4, 9]) → False array\_front9([1, 2, 3, 4, 5]) → False**

**PROGRAM CODE:**

class SP\_8

{

public static void main(String[] args)

{

int[] a1={1,2,9,3,4}; int[] a2={1,2,3,4,9}; int[] a3={1,2,3,4,5}; SP\_8 s1=new SP\_8(); boolean b1=s1.array\_front9(a1); boolean b2=s1.array\_front9(a2); boolean b3=s1.array\_front9(a3);

System.out.println(b1);

System.out.println(b2); System.out.println(b3);

}

boolean array\_front9(int[] arr)

{

int n=arr.length;

if(n>4)

n=4;

for(int i=0;i<n;i++)

{

if(arr[i]==9) return true;

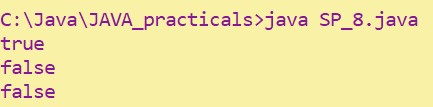
}

return false;

}

}

**OUTPUT:**



# PRACTICAL-4

**Given a string, return a string where for every char in the original, there are two chars.**

**double\_char('The') → 'TThhee' double\_char('AAbb') → 'AAAAbbbb' double\_char('Hi-There') → 'HHii--TThheerree'**

**PROGRAM CODE:**

class SP\_9

{

public static void main(String[] args)

{

SP\_9 s=new SP\_9();

String s1=s.double\_char("ABC");

System.out.println(s1);

}

String double\_char(String str)

{

String str2="";

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

{

str2=str2+str.charAt(i)+str.charAt(i);

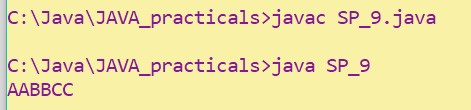
}

return str2;

}

}

**OUTPUT:**



# PRACTICAL-5

**Write a program that will reverse the sequence of letters in each word of your chosen paragraph. For instance, “To be or not to be” would become “oT e bro ton ot eb”.**

**Practical code:**

import java.util.\*; class SP\_10

{

public static void main(String[] args)

{

String tmp;

StringTokenizer s=new StringTokenizer("To be or not to be"); while(s.hasMoreTokens())

{

tmp=s.nextToken();

for(int i=tmp.length();i>0;i--)

{

System.out.print(tmp.charAt(i-1));

}

System.out.print(" ");

}

}

}

**OUTPUT:**

