

# **Java Experiments**

**60004210159**

**Dhruvin Chawda**

**Comps**

**B3**

## Experiment 1 : To implement Java Program Structures & Simple Programs (C01)

Theory: In this program we have used print an instance method of Print class which is used to print strings on the output window or console window. The println function print string in new line. System.out.println() is used to print statements on the next line. System: It is a final class defined in the java.lang package. out: This is an instance of PrintStream type, which is a public and static member field of the System class.

i. WAP to display hello Message on screen.

```
class Hello{
    public static void main(String args []){
        System.out.println("Hello World ");
    }
}
```

Output:

```
PS C:\Coding And Programming\javac\g> & 'C:\Users\Acer\AppData\Local\Programs\Eclipse Adoptium\jdk-17.0.4.101-hotspot\bin\java.exe' '-XX:+ShowCodeDetailsInExceptionMessages' '-cp' 'C:\Users\Acer\AppData\Roaming\Code\User\workspaceStorage\3c4c4f775fb2e598a4fec7679beda268\redhat.java\jdt_ws\javac\g_9864129f\bin' 'Hello'
Hello World
PS C:\Coding And Programming\javac\g>
```

ii) Write a Java program that reads a positive integer from command line and count the number of digits the number (less than ten billion) has.

```
public class commandlineinput
{
    public static void main(String args[])
    {
        int a=Integer.parseInt(args[0]);
        int c=0;
        while(a!=0)
        {
            a=a/10;
            c++;
        }
        System.out.print("Number of digits : "+c);
    }
}
```

## Output:

```
PS C:\Coding And Programming\javac\g> javac .\commandlineinput.java
PS C:\Coding And Programming\javac\g> java commandlineinput 1234
Number of digits : 4
PS C:\Coding And Programming\javac\g> 
```

## Experiment 2 : To implement Java control statements and loops (C01)

**Theory:** In this program I have used scanner function to take input of coefficient of quadratic equation I have calculated determinant and further process is taken with the help of determinant value if positive then formula method and if equal to zero then both roots are equal and if negative then imaginary roots the final roots are shown using normal output method . The if Statement Use the if statement to specify a block of Java code to be executed if a condition is true. If Else Statement Use the else if statement to specify a new condition if the first condition is false. Else Statement Use the else statement to specify a block of code to be executed if the condition is false.

Formula:  $ax^2 + bx + c = 0$

When  $b^2 - 4ac = 0$  there is one real root.

When  $b^2 - 4ac > 0$  there are two real roots.

When  $b^2 - 4ac < 0$  there are two complex roots.

1 . WAP to find roots of a Quadratic equation. Take care of imaginary values

```
import java.util.Scanner;
public class Quad
{
    public static void main(String[] args) {
        Scanner input = new Scanner(System.in);
        System.out.println("Enter a b and c");
        double a = input.nextInt();
        double b = input.nextInt();
        double c = input.nextInt();
        double deter = b * b - 4 * a * c;
        if (deter > 0) {
            double deter_root = Math.sqrt(deter);
            System.out.println("Root 1=" + (-b + deter_root) / (2 * a));
            System.out.println("Root 2=" + (-b - deter_root) / (2 * a));
        } else if (deter == 0) {
            double deter_root = Math.sqrt(deter);
            System.out.println("Root 1=Root 2" + (-b + deter_root) /
(2 * a));
        } else {
            double deter_root = Math.sqrt(-deter);
            double real = -b / (2 * a);
            double img = deter_root / (2 * a);
            System.out.println("Root 1=" + real+"+"+img+"i");
            System.out.println("Root 1=" + real+"-"+img+"i");
        }
        input.close();
    }
}
```

Output :

```
Enter a b and c
2 4 5
Root 1=-1.0+1.224744871391589i
Root 1=-1.0-1.224744871391589i
PS C:\Coding And Programming\javac\lg> █
```

ii. Write a menu driven program using switch case to perform mathematical operations.

**Theory :** In this program I have taken 3 inputs 2 are numbers and 1 is operation that one wants to perform .When one runs the code is provided with 4 option of add, sub, multiply and divide User input the operation he wants to perform with the help of switch case the time efficiency of code decreases and provides fast output.

**SYNTAX**

```
switch(expression) { case x: // code block break; case y: // code block break;
default: // code block }
```

```
import java.util.Scanner;

public class Switch
{
    public static void main(String args [])
    {
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter two number : ");
        int a =sc.nextInt();
        int b =sc.nextInt();
        System.out.print("Menu\n1)ADD\n2)Subtract\n3)Multi\n4)Div\n");
        System.out.print("Enter choice : ");
        int n = 0;
        int c = sc.nextInt();
        switch(c)
        {
            case 1 :
                System.out.print(a+b);
                break;
            case 2:
                System.out.print(a-b);
                break;
            case 3 :
                System.out.print(a*b);
                break;
            case 4 :
                System.out.print((a/b));
                break;
            default:
                break;
        }
    }
}
```

## Output:

```
Enter two number : 90
10
Menu
1)ADD
2)Subtract
3)Multi
4)Div
Enter choice : 4
9
```

### iii. ) WAP to display odd numbers from given range/ prime numbers from given range

**Theory:** In This 2 program are mix together one is finding odd number which is done using for loop and if condition for a given range of number and 2 is finding prime number using nested for loop where one for loop gives the number and second for loop divide that number to verify is it prime last we get output.

#### SYNTAX

for (statement 1; statement 2; statement 3) { // code block to be executed } Statement 1 is executed (one time) before the execution of the code block. Statement 2 defines the condition for executing the code block. Statement 3 is executed (every time) after the code block has been executed.

```
import java.util.Scanner;

public class prime1
{
    public static void main (String[] args)
    {
        int i =0;
        int num =0;
        Scanner sc = new Scanner(System.in);
        System.out.print("enter a number : ");
        int n = sc.nextInt();
        System.out.print("prime numbers :");
        for (i = 1; i <= n; i++)
        {
            int counter=0;
            for(num =i; num>=1; num--)
            {
                if(i%num==0)
                {
                    counter = counter + 1;
                }
            }
            if (counter ==2)
```

```

{
    if(i%2!=0)
        {System.out.print(i+" ");}
}

}

System.out.print("\nOdd numbers :");
for (i = 1; i <= n; i++){
    if(i%2!=0)
        {System.out.print(i+" ");}
}
}
}

```

Output:

```

enter a number : 30
prime numbers :3 5 7 11 13 17 19 23 29
Odd numbers :1 3 5 7 9 11 13 15 17 19 21 23 25 27 29
PS C:\Coding And Programming\javac\lg> 

```

#### iv. WAP to display default value of primitive data types

##### Theory :

In this program we have print default values of all the primitive data types in java the primitive means those aren't considered objects and represent raw values. There are total 8 primitive data types in java int, byte, short, long, float, double, boolean, and char and all default values are printed. The eight primitives defined in Java are int, byte, short, long, float, double, boolean, and char – those aren't considered objects and represent raw values

```

import java.util.*;
class DefaultValue
{
    int i;
    float f;
    double d;
    long l;
    boolean bl;
    short s;
    byte b;
    char ch;
    public static void main(String args[])
    {
        DefaultValue sc = new DefaultValue();
        System.out.println("The Deafult Value of int is: "+sc.i);
        System.out.println("The Deafult Value of float is: "+sc.f);
        System.out.println("The Deafult Value of double is: "+sc.d);
    }
}

```

```

System.out.println("The Deafult Value of long is: "+sc.l);
System.out.println("The Deafult Value of boolean is: "+sc.bl);
System.out.println("The Deafult Value of short is: "+sc.s);
System.out.println("The Deafult Value of byte is: "+sc.b);
System.out.println("The Deafult Value of char is: "+sc.ch);
}
}

```

Output :

```

The Deafult Value of int is: 0
The Deafult Value of float is: 0.0
The Deafult Value of double is: 0.0
The Deafult Value of long is: 0
The Deafult Value of boolean is: false
The Deafult Value of short is: 0
The Deafult Value of byte is: 0
The Deafult Value of char is:
PS C:\Coding And Programming\javac\lg>

```

v. WAP to display the following patterns:

Pattern 1:

```

import java.util.Scanner;
public class pattern0
{
    public static void main(String args[])
    {
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter a number : ");
        int n = sc.nextInt();

        for(int i =1;i<=n;i++)
        {
            if(i%2==0)//even
            {
                for(int j=i;j>=1;j--)
                {
                    System.out.print(j);
                }
            }
            else
            {
                for(int k=1;k<=i;k++)
                {
                    System.out.print(k);
                }
            }
        }
    }
}

```



```

        // int z=s;

        System.out.println("");
    }
}
}

```

Output:

```

Enter a number : 7
1
21
123
4321
12345
654321
1234567
PS C:\Coding And Programming\javac\lg>

```

Pattern 2:

```

import java.util.Scanner;
public class Pattern1
{
    public static void main(String args[])
    {
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter a number : ");
        int n = sc.nextInt();
        int s=0;
        for(int i =1;i<=n;i++)
        {
            s=s+i;
            for(int k=i;k<=n;k++)
            {
                System.out.print(" ");
            }
            int z=s;
            for(int j=1;j<=i;j++)
            {
                System.out.print((char)(z+64));
                z--;
            }
            System.out.println("");
        }
    }
}

```

```
}  
}
```

**Output :**

```
Enter a number : 4  
  A  
  CB  
  FED  
  JIHG  
PS C:\Coding And Programming\javac1g> |
```

## Experiment 3 : 3. To implement Arrays (C03)

i. WAP to find whether the entered 4 digit number is vampire or not. Combination of digits from this number forms 2 digit number. When they are multiplied by each other we get the original number. (1260=21\*60, 1395=15\*93, 1530=30\*51)

**Theory:** In this program we have to find a whether the number is vampire or not and if number is not four digit then the code is not break and not a vampire is printed if number is 4 digit then the code further process . Java array is an object which contains elements of a similar data type. Additionally, The elements of an array are stored in a contiguous memory location. It is a data structure where we store similar elements. We can store only a fixed set of elements in a Java array

```
import java.util.*;
public class Vampire1
{
    public static void main(String[] args)
    {
        System.out.print("Enter the number : ");
        Scanner sc=new Scanner(System.in);
        int size=4;
        int num,nums[],digits[];
        nums=new int[size];
        digits=new int[size];
        num=sc.nextInt();
        for(int i=0;i<4;i++)
        { digits[i]=(num/(int)Math.pow(10,size-i-1))%10;}
        System.out.println();
        nums[0]=digits[0]*10+digits[1];
        nums[1]=digits[1]*10+digits[0];
        nums[2]=digits[2]*10+digits[3];
        nums[3]=digits[3]*10+digits[2];
        if(nums[0]*nums[2]==num || nums[0]*nums[3]==num || nums[1]*nums[2]==num || nu
ms[0
        ]*nums[3]==num)
        { System.out.println(num+" is vampire");}
        else
        { System.out.println(num+" is not vampire");}
    }
}
```

### Output:

```
Enter the number : 1530
```

```
1530 is vampire
```

```
PS C:\Coding And Programming\javac\lg> █
```

## ii. WAP to display the following using irregular arrays

1

2 3

4 5 6

### Theory:

Irregular array is an array of arrays, where the inner arrays can be of different sizes. So the 2-d array can have a variable number of columns in each row. The instantiation of such arrays is done differently than regular arrays: `int arr[][] = new int[2][]; arr[0] = new int[3];` // the zeroth row has 3 columns `arr[1] = new int[2];` // the first row has 2 columns For traversing irregular arrays we have to use the length property of array in the inner loop. There are two types of array. -

Single Dimensional Array - Multidimensional Array

```
import java.util.*;

public class irregular{
    public static void main(String args[])
    {
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter the number of row : ");
        int n = sc.nextInt();
        int [][]a = new int[n][n];
        for(int i=0; i<n;i++){
            for(int j=0; j<i+1;j++){
                System.out.print("Enter the numbers :");
                a[i][j]= sc.nextInt();
            }
        }
        for(int i=0; i<n;i++){
            for(int j=0; j<i+1;j++){
                System.out.print(a[i][j]);

                // a[i][j]= sc.nextInt();
            }System.out.print('\n');
        }
    }
}
```

Output :

```

Enter the number of row : 3
Enter the numbers :1
Enter the numbers :2
Enter the numbers :3
Enter the numbers :4
Enter the numbers :5
Enter the numbers :6
1
23
456
PS C:\Coding And Programming\javac\lg>

```

3)

Write a program that queries a user for the no. of rows and columns representing students and their marks.

Reads data row by row and displays the data in tabular form along with the row totals, column totals and grand total

Hint : For the data 1, 3, 6, 7, 9, 8 the output is

1	3	6		10
7	9	8		24
8	12	14		34

#### Theory:

In this program we are taking user input on number of rows, columns and numbers to be entered in these rows and columns. We are storing these numbers in a 2d array. Two arrays are used for storing the sum of numbers in columns and rows respectively. We first fill the arrays with zero using the fill() method, and then add the subsequent values. Finally we traverse through the 2d array and print the values, in the inner loop we print the row sum. At the end we traverse through the column sum array and print the same. Multidimensional Arrays can be defined in simple words as array of arrays. Data in multidimensional arrays are stored in tabular form (in row major order).

```

import java.util.*;

public class table{
    public static void main(String args[])
    {
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter the number of students : ");
        int n = sc.nextInt();
        System.out.print("Enter the number of subjects : ");
        int m = sc.nextInt();
        int [][]a = new int[(n+1)][(m+1)];
        for(int i=0; i<n; i++){
            for(int j=0; j<m; j++){
                System.out.print("Enter the marks of sub "+(j+1)+":");
                a[i][j]= sc.nextInt();
            }
        }
        //
        int c=0;
        // System.out.print(a.length);
        for(int i=0; i<n; i++){
            for(int j=0; j<m; j++){
                c=c+a[i][j];
            }
        }
    }
}

```

```

    }
    a[i][m]=c;
    c=0;
}
//
for(int j=0; j<m;j++){
    for(int i=0; i<n;i++){
        c=c+a[i][j];
    }
    a[n][j]=c;
    c=0;
}

for(int i=0; i<=n;i++){
    c=c+a[i][m];
}
a[n][m]=c;
c=0;

for(int i=0; i<=n;i++){
    for(int j=0; j<=m;j++){
        System.out.print(a[i][j]);
        System.out.print(" ");
        if(j==m-1){
            System.out.print("|");
        }

        // a[i][j]= sc.nextInt();
    }System.out.print('\n');
    if(i==n-1)
    {System.out.print("-----");
    System.out.print('\n');}
}
}
}

```

## Output :

```
Enter the number of students : 4
Enter the number of subjects : 3
Enter the marks of sub 1:10
Enter the marks of sub 2:5
Enter the marks of sub 3:25
Enter the marks of sub 1:9
Enter the marks of sub 2:4
Enter the marks of sub 3:20
Enter the marks of sub 1:10
Enter the marks of sub 2:5
Enter the marks of sub 3:22
Enter the marks of sub 1:8
Enter the marks of sub 2:3
Enter the marks of sub 3:19
10 5 25 |40
9 4 20 |33
10 5 22 |37
8 3 19 |30
-----
37 17 86 |140
PS C:\Coding And Programming\javac1g> 
```

## Experiment 4 : To implement Vectors (C03)

4. i)WAP that accepts a shopping list of items and performs the following operations: Add an item at a specified location, delete an item in the list, and print the contents of the vector

**Theory:** This is a menu driven program which uses array for maintaining a shopping list. We accept the number of items to be inserted in the shopping list from the user and create a Vector of that size with the increment size 3. That is, whenever the number of elements increase the size, the size is increased by 3. We accept the elements in the list and add in the Vector. Next, we show the menu with possible options. We use the add() method to put a value at specific location given by the user, we use the remove() method to remove the element mentioned by the user. Vector is a class in java in java.util package. It implements a dynamic array which can increase and decrease in size. It can be accessed by index like in arrays. The Iterators returned by the Vector class are fail-fast. In case of concurrent modification, it fails and throws the ConcurrentModificationException. Therefore it is recommended to use the Vector class in the thread-safe implementation only. The Vector class has a number of pre defined methods for working with them. Vector v = new Vector(); // default vector of the initial capacity is 10. Vector v = new Vector(int size); // Initial capacity is specified by size. Vector v = new Vector(int size, int incr); // initial capacity is specified by size and increment is //specified by incr. It specifies the number of elements to allocate each time a vector is resized upward

```
import java.util.*;
public class Vector0

{
    public static void main(String[] args)
    {
        Scanner sc = new Scanner(System.in);
        System.out.print("How many elements are there in List : ");
        int n = sc.nextInt();
        Vector v = new Vector(n, 3);
        for (int i = 0; i < n; i++)
        {
            System.out.printf("Enter the item %d : ", i + 1);
            String str = sc.next();
            v.addElement(str);
        }
        System.out.println("\t\tOperation");
        System.out.println("\t1)add at specific location");
        System.out.println("\t2>Delete item");
        System.out.println("\t3)Display list");
        System.out.println("\t4)Exit");
        int choice = 1;
        while (choice != 4)
        {
            System.out.print("Enter the choice : ");
            choice = sc.nextInt();
            switch (choice)
            {
                case 1:
                    System.out.print("Enter the Location(index) : ");
```



```

        int l = sc.nextInt();
        System.out.print("Enter the item : ");
        String str = sc.next();
        v.add(l, str);
        break;
        case 2:
        System.out.print("Enter the item want to delete : ");
        String item = sc.next();
        v.remove(item);
        break;
        case 3:
        System.out.print("The elements in List are : " + v + "\n");
        break;
        case 4:
        return;
        default:
        System.out.print("wrong choice ");
    }
}
}
}

```

## Output :

```

How many elements are there in List : 4
Enter the item 1 : peru
Enter the item 2 : tomato
Enter the item 3 : potato
Enter the item 4 : mango
        Operation
        1)add at specific location
        2)Delete item
        3)Display list
        4)Exit
Enter the choice : 3
The elements in List are : [peru, tomato, potato, mango]
Enter the choice : 1
Enter the Location(index) : 1
Enter the item : watermelom
Enter the choice : 3
The elements in List are : [peru, watermelom, tomato, potato, mango]
Enter the choice : 2
Enter the item want to delete : peru
Enter the choice : 3
The elements in List are : [watermelom, tomato, potato, mango]
Enter the choice : 4
PS C:\Coding And Programming\javac\lg>

```

i. Write a java programs to find frequency of an element in the given Vector array.

Theory:

In this program we accept the size of vector from the user and the values to be inserted. We also accept the element whose frequency is to be calculated. The indexOf() method returns the index of the element from the mentioned index. We increase the counter and change the start index passed to indexOf() until it returns -1 i.e. there are no more occurrences. We display the counter

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

class Vector1
{
    public static void main(String args[])
    {
        Scanner sc = new Scanner(System.in);
        int c=0;
        Vector<Integer>v = new Vector<Integer>();
        // System.out.println(v);
        while(c<6)
        {
            // System.out.println(v.size());
            System.out.println("choice : ");
            c = sc.nextInt();
            switch(c){
                case 1 : System.out.print("add : ");v.add(sc.nextInt());break;
                case 2 : System.out.print("remove : ");v.remove((sc.nextInt()-
1));break;
                case 3 : System.out.print("displaying : 
");System.out.print(v);break;
                case 4 : System.out.print("get : 
");System.out.println("find"+v.get(sc.nextInt()));break;
                case 5 :
                {
                    System.out.println("frequency Of : ");
                    int k=sc.nextInt(),j=0;
                    for(int i =0;i<v.size();i++)
                    {if(v.get(i)==k)j++;}
                    System.out.println("Freq : "+j);
                }
            }
        }
    }
}
```

## Output:

```
choice :  
1  
add : 1  
choice :  
1  
add : 5  
choice :  
1  
add : 6  
choice :  
1  
add : 5  
choice :  
  
1  
add : 5  
choice :  
1  
add : 6  
choice :  
3  
displaying : [1, 5, 6, 5, 5, 6]choice :  
5  
frequency Of :  
5  
Freq : 3  
choice :  
5  
frequency Of :  
6  
Freq : 2  
choice :  
6  
PS C:\Coding And Programming\javac\lg> █
```

## Experiment 5 : To implement Strings (C03)

i. WAP to check if 2 strings are Meta strings or not. Meta strings are the strings which can be made equal by exactly one swap in any of the strings. Equal string are not considered here as Meta strings. Example: str1 = "geeks", str2 = "keegs" By just swapping 'k' and 'g' in any of string, both will become same. Example: str1 = "Converse", str2 = "Conserve" By just swapping 'v' and 's' in any of string, both will become same.

**Theory:** In this program we take input of two strings. We check if their length is the same, if it is we check the positions where the strings are unequal. If there are more than two positions then the strings are not meta. For two positions we check if swapping them can make the string equal, if it does, then the strings are meta.

```
import java.util.*;
public class Meta1
{
    static boolean isMeta(String word1,String word2)
    {
        int wrong_count=0,wrong_pos[];
        wrong_pos=new int[2];
        if(word1.length()!=word2.length())
            return false;
        for(int i=0;i<word1.length();i++)
        {
            if(word1.charAt(i)!=word2.charAt(i))
            {
                wrong_count++;
                if(wrong_count>2)
                    return false;
                wrong_pos[wrong_count-1]=i;
            }
        }
        if(wrong_count==2)
        {
            if(word1.charAt(wrong_pos[0])==word2.charAt(wrong_pos[1])&&word1.ch
arAt(wrong_pos[1])==word2.charAt(wrong_pos[0]))
                return true;
        }
        return false;
    }
    public static void main(String[] args)
    {
        System.out.println("Enter two strings : ");
        Scanner sc=new Scanner(System.in);
        String word1,word2;
        word1=sc.next();
        word2=sc.next();
        if(isMeta(word1, word2))
```

```

        {System.out.println("Strings are meta");}
        else
        {System.out.println("Strings are not meta");}
    }
}

```

Output:

```

Enter two strings :
geeks
keegs
Strings are meta
PS C:\Coding And Programming\javac\lg> 

```

ii. Write a java program to count number of alphabets, digits, special symbols, blank spaces and words from the given sentence. Also count number of vowels and consonants

**Theory:**

In this program we take the string input from the user. We count the number of vowels and consonants by traversing every character and comparing them with the vowel and consonant sets. We call the count() method, which counts the number of alphabets, digits, special symbols and blank spaces. This is done by methods of the Character class isLetter(), isSpaceChar(), isDigit(). We call the word() method to calculate the number of words. We parse through the string and increase the word counter if there is a space before the current character or if it is the first character of the string.

```

import java.util.*;
import java.io.*;
public class Countsen{
    public static void num(String s)
    {
        int num=0, alhpa=0, space=0, syn=0;
        for(int i =0; i<s.length();i++)
        {
            char c = s.charAt(i);
            if(Character.isDigit(c))
            {num++;}
            else if(Character.isLetter(c))
            {alhpa++;}
            else if(Character.isSpaceChar(c))
            {space++;}
            else{syn++;}
        }
        System.out.println("num : "+ num);
        System.out.println("alpha : "+ alhpa);
        System.out.println("space : "+ space);
        System.out.println("Sysmbol : "+ syn);
    }
    public static void main(String args[])
    {
        Scanner sc = new Scanner(System.in);
        System.out.println("Enter a sentence : ");
    }
}

```

```

String s = sc.nextLine();
int vowel = 0, cons=0;
// String []a = new String[s.length()];
for(int i=0;i<s.length();i++){
    // System.out.print(s.charAt(i)+" ");
    char c = s.charAt(i);
    if(c=='a' || c=='o' || c=='i' || c=='e' || c=='u')
        {vowel++;}
    else if(c>'a' && c<='z')
        {cons++;}

}
System.out.println("vowels : "+ vowel);
System.out.println("cons : "+ cons);
num(s);
}
}

```

## Output:

```

Enter a sentence :
Scarlett Witch 31 !#
vowels : 3
cons : 8
num : 2
alpha : 13
space : 3
Sysmbol : 2
PS C:\Coding And Programming\javac\lg> █

```

## Experiment 6 : To implement Functions, recursive functions and overloading (C01)

### 1) WAP to display area of square and rectangle using the concept of overloaded functions

#### Theory:

This is a Java Program to Find Area of Square And Rectangle using Method Overloading. We declare three methods of same name but with different number of arguments or with different data types. Now when we call these methods using objects, corresponding methods will be called as per the number of arguments or their datatypes. Here is the source code of the Java Program to Find Area of Square, Rectangle using Method Overloading. The Java program is successfully compiled and run on a Windows system. The program output is also shown below. Overloading allows different methods to have the same name, but different signatures where the signature can differ by the number of input parameters or type of input parameters or both. Overloading is related to compile-time (or static) polymorphism.

```
import java.util.*;

class Overload
{
    void area(float s)
    {
        System.out.println("Area of square is : "+s*s);
    }
    void area(float l, float b)
    {
        System.out.println("Area of rec is : "+l*b);
    }
}

public class OL
{
    public static void main(String arg[])
    {
        Scanner sc = new Scanner (System.in);
        Overload o = new Overload();

        int n=0;
        while(n<3){
            System.out.print("Enter the choice \n 1)square 2) Rec \n");
            n = sc.nextInt();
            switch (n) {
                case 1:
                    System.out.print("Enter the sides : ");
                    float s = sc.nextFloat();
                    o.area(s);
                    break;

                case 2:
                    System.out.print("Enter the sides : ");
```

```

        float u = sc.nextFloat();
        float v= sc.nextFloat();
        o.area(u,v);
        break;

    default:
        break;
    }}
}
}

```

## Output:

```

Enter the choice
1)square 2) Rec
1
Enter the sides : 6.2
Area of square is : 38.44
Enter the choice
1)square 2) Rec
2
Enter the sides :
5
6.2
Area of rec is : 31.0
Enter the choice
1)square 2) Rec
3
PS C:\Coding And Programming\javac\lg>

```

## iii. Write menu driven program to implement recursive functions for following tasks

### Theory:

In this program we will be writing a menu driven program implementing the concept of recursive functions. Recursion is the technique of making a function call itself. It is a process in which a method calls itself continuously. A method in java that calls itself is called recursive method. This technique provides a way to break complicated problems down into simple problems which are easier to solve. Here we've created multiple classes which execute a given respective function and all traversed using a menu written using switch case. Recursion in java is a process in which a method calls itself continuously. A method in java that calls itself is called recursive method. It makes the code compact but complex to understand. Syntax: `returntype methodName(){ //code to be executed methodName();//calling same method }`



```
import java.util.Scanner;
public class Rec2
{
    // gcd lcm
    static public int gcd(int a, int b) {
        if (b == 0) {
            return a;
        }
        return gcd(b, a % b);
    }
    static int lcm(int a, int b) {
        return (a / gcd(a, b)) * b;
    }
    static int dig = 0;
    static int s = 0;
    // reverse
    static int solve(int n) {
        if (n != 0) {
            dig = dig * 10 + n % 10;
            solve(n / 10);
        }
        return dig;
    }
    // sum of digits
    static int sum(int n) {
        if (n != 0) {
            s = s + n % 10;
            sum(n / 10);
        }
        return s;
    }
    static int sum = 0;
    static int solveN(int nth) {
        if (nth > 0) {
            sum = sum + nth;
            solveN(nth - 1);
        }
        return sum;
    }
    static int a = 0;
    static int b = 1;
    static int c = 0;
    static int m = 1;
    static void fibo(int n) {
        if (n > 0) {
            c = a + b;
            System.out.print(c + " ");
            a = b;
            b = c;
            fibo(n - 1);
        }
    }
    static int multi(int x, int y) {
        if (y > 0) {
```

```

m = m * x;
multi(x, y - 1);
}
return m;
}
public static void main(String[] args)
{
Scanner input = new Scanner(System.in);
int fn = 0;
while(fn < 7){
System.out.println("Enter the function");
System.out.println("1) To find GCD and LCM");
System.out.println("2) To find X^Y ");
System.out.println("3) To print n Fibonacci numbers");
System.out.println("4) To find reverse of number ");
System.out.println("5) To 1+2+3+4+..+ (n-1)+n");
System.out.println("6) Calculate the sum of digits of a number ");
fn = input.nextInt();
switch (fn)
{
case 1:
int temp;
System.out.println("Enter the two numbers:");
int a = input.nextInt();
int b = input.nextInt();
if (a < b) {
temp = a;
a = b;
b = temp;
}
System.out.print("GCD is :");
System.out.println(gcd(a, b));
System.out.print("LCM is :");
System.out.println(lcm(a, b));
break;
case 2:
System.out.println("Enter the x and y value");
int x = input.nextInt();
int y = input.nextInt();
if (y == 0) {
System.out.println("Answer:" + 1);
} else {
System.out.println("Answer:" + multi(x, y));
}
break;
case 3:
// fibo
a = 0;
b = 1;
System.out.println("Enter the nth value");
int count = input.nextInt();
System.out.print(a + " ");
System.out.print(b + " ");
fibo(count - 2);

```

```
break;
case 4:
System.out.println("Enter the number");
int n = input.nextInt();
System.out.println("reverse is " + solve(n));
break;
case 5:
System.out.println("Enter the nth value");
int nth = input.nextInt();
for (int i = 1; i <= nth; i++) {
if (i == nth) {
System.out.print(i);
} else {
System.out.print(i + "+");
}
}
System.out.print("=" + solveN(nth));
break;
case 6:
System.out.println("Enter the number");
int m = input.nextInt();
System.out.println("Sum is " + sum(m));
break;
default:
System.out.println("Invalid input");
}
}
}}
```

**Output:**

```

Enter the function
1) To find GCD and LCM
2) To find X^Y
3) To print n Fibonacci numbers
4) To find reverse of number
5) To 1+2+3+4+...+ (n-1)+n
6) Calculate the sum of digits of a number
1
Enter the two numbers:
9
6
GCD is :3
LCM is :18
Enter the function
1) To find GCD and LCM
2) To find X^Y
3) To print n Fibonacci numbers
4) To find reverse of number
5) To 1+2+3+4+...+ (n-1)+n
6) Calculate the sum of digits of a number
2
Enter the x and y value
5
3
Answer:125
Enter the function
1) To find GCD and LCM
2) To find X^Y
3) To print n Fibonacci numbers
4) To find reverse of number
5) To 1+2+3+4+...+ (n-1)+n
6) Calculate the sum of digits of a number
6) Calculate the sum of digits of a number
4
Enter the number
1234
reverse is 4321
Enter the function
1) To find GCD and LCM
2) To find X^Y
3) To print n Fibonacci numbers
4) To find reverse of number
5) To 1+2+3+4+...+ (n-1)+n
6) Calculate the sum of digits of a number
5
Enter the nth value
10
1+2+3+4+5+6+7+8+9+10=55Enter the function
1) To find GCD and LCM
2) To find X^Y
3) To print n Fibonacci numbers
4) To find reverse of number
5) To 1+2+3+4+...+ (n-1)+n
6) Calculate the sum of digits of a number
6
Enter the number
549
Sum is 18
Enter the function
1) To find GCD and LCM
2) To find X^Y
3) To print n Fibonacci numbers
4) To find reverse of number
5) To 1+2+3+4+...+ (n-1)+n
6) Calculate the sum of digits of a number
7
Invalid input
PS C:\Coding And Programming\javac\lg>

```

## Experiment 7 : To implement Array of Objects (C02)

i. W00P to arrange the names of students in descending order of their total marks, input data consists of students details such as names, ID.no, marks of maths, physics, chemistry. (Use array of objects)

### Theory:

In this program we implement the use of array of object, The array of Objects the name itself suggests that it stores an array of objects. Unlike the traditional array stores values like String, integer, Boolean, etc an Array of Objects stores objects that mean objects are stored as elements of an array, here we are accepting and storing and displaying multiple student data such as name, rollno and marks into its respective object stored in an array. The objects in the array are traversed with each object accepting and displaying the user input it had stored and also sort the data in descending order of the data.

```
import java.util.Scanner;
class Student
{
int roll,phy,chem,math,total; String name;
void input()
{
Scanner scan=new Scanner(System.in);
System.out.println();
System.out.print("Enter student name:");
name=scan.nextLine();
System.out.print("Enter Roll_no:");
roll=scan.nextInt();
System.out.println("Enter Marks:");
System.out.print("Physics Marks:");
phy=scan.nextInt();
System.out.print("Chemistry Marks:");
chem=scan.nextInt();
System.out.print("Mathematics Marks:");
math=scan.nextInt();
total=phy+chem+math;
System.out.println();
System.out.println("*****Student details registered*****");
}
void output()
{
System.out.println("Student: "+name+" ,roll_no: "+roll+" ,marks: ");
System.out.println("Physics:"+phy); System.out.println("Chemistry:"+chem);
System.out.println("Mathematics:"+math); System.out.println("Total:"+total);
}
}
class Arr0
{
public static void main(String args[])
{
int i,j;
Student s[]=new Student[5]; for(i=0;i<5;i++)
{
s[i]=new Student();
}
}
```

```

System.out.println("Enter Details: "); for(i=0;i<5;i++)
{
s[i].input();
}
for(i=0;i<5;i++)
{
s[i].output();
}
Student temp; for(i=0;i<4;i++)
{
for(j=0;j<4-i;j++)
{
if(s[j].total<s[j+1].total)
{
temp=s[j]; s[j]=s[j+1]; s[j+1]=temp;
}
}
}
}
System.out.println("Student Marks in Descendin Order:"); for(i=0;i<5;i++)
{
System.out.println("Student Name: "+s[i].name+", Student RollNo: "+s[i].roll+",
Total: "+s[i].total);
}
}
}
}

```

## Output :

```

PS C:\Coding And Programming\javac\g> javac Arr0.java
PS C:\Coding And Programming\javac\g> java Arr0
Enter Details:

Enter student name:dhruvin
Enter Roll_no:159
Enter Marks:
Physics Marks:50
Chemistry Marks:50
Mathematics Marks:50

****Student details registered****

Enter student name:dev
Enter Roll_no:130
Enter Marks:
Physics Marks:51
Chemistry Marks:23
Mathematics Marks:29

****Student details registered****

Enter student name:param
Enter Roll_no:161
Enter Marks:
Physics Marks:60
Chemistry Marks:59
Mathematics Marks:39

```

\*\*\*\*\*Student details registered\*\*\*\*\*

Enter student name:devansh  
Enter Roll\_no:100  
Enter Marks:  
Physics Marks:50  
Chemistry Marks:40  
Mathematics Marks:60

\*\*\*\*\*Student details registered\*\*\*\*\*

Enter student name:prerak  
Enter Roll\_no:154  
Enter Marks:  
Physics Marks:50  
Chemistry Marks:60  
Mathematics Marks:60

\*\*\*\*\*Student details registered\*\*\*\*\*

Student: dhruvin ,roll\_no: 159 ,marks:  
Physics:50  
Chemistry:50  
Mathematics:50  
Total:150

Student: dev ,roll\_no: 130 ,marks:  
Physics:51  
Chemistry:23  
Mathematics:29  
Total:103

Student: param ,roll\_no: 161 ,marks:  
Physics:60  
Chemistry:59  
Mathematics:39  
Total:158

Student: devansh ,roll\_no: 100 ,marks:  
Physics:50  
Chemistry:40  
Mathematics:60  
Total:150

Student: prerak ,roll\_no: 154 ,marks:  
Physics:50  
Chemistry:60  
Mathematics:60  
Total:170

Student Marks in Descendin Order:

Student Name: prerak, Student RollNo: 154, Total: 170  
Student Name: param, Student RollNo: 161, Total: 158  
Student Name: dhruvin, Student RollNo: 159, Total: 150  
Student Name: devansh, Student RollNo: 100, Total: 150  
Student Name: dev, Student RollNo: 130, Total: 103

PS C:\Coding And Programming\javac\lg> █

## Experiment 8 : To implement Constructors and overloading (CO2)

### i. WAP find area of square and rectangle using overloaded constructor

#### Theory:

In this program we have implemented the concept of constructor overloading, The constructor overloading can be defined as the concept of having more than one constructor with different parameters so that every constructor can perform a different task. Here we have created a class named Shape and two constructors but having different numbers and types of parameters hence applying the concept of Constructor Overloading. The constructor overloading can be defined as the concept of having more than one constructor with different parameters so that every constructor can perform a different task.

```
import java.util.*;

class Cons
{
    float a,d;
    Cons(float s)
    {
        a=s;
        d=s;
    }
    Cons(float l, float b)
    {
        a=l;
        d=b;
    }
    void area()
    {
        System.out.println("Area : "+a*d);
    }
}

class ConsOL
{
    public static void main(String arg[])
    {
        Scanner sc = new Scanner (System.in);
        System.out.print("Enter the choice \n 1)square 2) Rec \n");
        int n = sc.nextInt();
        switch (n) {
            case 1:
                System.out.print("Enter the sides : ");
                Cons sq = new Cons(sc.nextInt());
                sq.area();
                break;

            case 2:
                System.out.print("Enter the sides : ");
                Cons rec = new Cons(sc.nextInt(),sc.nextInt());
                rec.area();
                break;

            default:
                break;
        }
    }
}
```



```
}  
}
```

## Output:

```
Enter the choice  
1)square 2) Rec  
1  
Enter the sides : 5  
Area : 25.0  
PS C:\Coding And Programming\javac\g> c:: cd 'c:\Coding And Programming\javac\g  
'; & 'C:\Users\Acer\AppData\Local\Programs\Eclipse Adoptium\jdk-17.0.4.101-hotsp  
ot\bin\java.exe' '-XX:+ShowCodeDetailsInExceptionMessages' '-cp' 'C:\Users\Acer\  
AppData\Roaming\Code\User\workspaceStorage\3c4c4f775fb2e598a4fec7679beda268\redh  
at.java\jdt_ws\javac\g_9864129f\bin' 'ConsOL'  
Enter the choice  
1)square 2) Rec  
2  
Enter the sides :  
6  
5  
Area : 30.0  
PS C:\Coding And Programming\javac\g> |
```

ii. Create Rectangle and Cube class that encapsulates the properties of a rectangle and cube i.e. Rectangle has default and parameterized constructor and area() method. Cube has default and parameterized constructor and volume() method. They share no ancestor other than Object. Implement a class Size with size() method. This method accepts a single reference argument z. If z refers to a Rectangle then size(z) returns its area and if z is a reference of Cube, then z returns its volume. If z refers to an object of any other class, then size(z) returns -1. Use main method in Size class to call size(z) method.

### Theory:

The following program implements the use of abstract class where we declare all the functions and define and use it in another class by extending the abstract class. An abstract class is used if you want to provide a common, implemented functionality among all the implementations of the component. Abstract classes will allow you to partially implement your class. Therefore, it is also known as data hiding, and as discussed previously, Constructor overloading is when there are multiple method which has a same name as the class or constructors, act different on having different parameters, later in this code we have also implemented the class Size with the method size(), it is a method is used to get the size of the Set or the number of elements present in the Set. Parameterized Constructor – A constructor is called Parameterized Constructor when it accepts a specific number of parameters. To initialize data members of a class with distinct values. With a parameterized constructor for a class, one must provide initial values as arguments, otherwise, the compiler reports an error.

```
import java.util.*;  
class Rect{  
    private int l,b;  
    Rect(int l,int b)  
    {  
        this.l=l;
```

```

    this.b=b;
}
int area() {
return l*b;
}
}
class Cube{
    private int side;
    Cube(int side)
    {
this.side=side;
}
int volume(){
return side*side*side;
}
}
class Size{
public static int size(Object o){
if(o instanceof Rect){
return ((Rect)o).area();
}
else if(o instanceof Cube){
return ((Cube)o).volume();
}
else {
return -1;
}
}
}
public class Exp8
{
public static void main(String[] args)
{
    Scanner sc = new Scanner(System.in);
    Rect r = new Rect(5,6);
    Cube c = new Cube(4);
    System.out.println("Area of Rectangle : "+Size.size(r));
    System.out.println("Volume of Cube : "+Size.size(c));
    System.out.println("Other objects : "+Size.size(sc));
}
}
}

```

## Output:

```

PS C:\Coding And Programming\javac\g> javac .\Exp8.java
PS C:\Coding And Programming\javac\g> java Exp8
Area of Rectangle : 30
Volume of Cube : 64
Other objects : -1
PS C:\Coding And Programming\javac\g> █

```

## Experiment 9 : To implement Abstract classes (C04)

### i. Write a abstract class program to calculate area of circle, rectangle and triangle

#### Theory:

In this below given program we have implemented concepts like data encapsulation , constructor overloading. Encapsulation in Java is a mechanism of wrapping the data (variables) and code acting on the data (methods) together as a single unit. In encapsulation, the variables of a class will be hidden from other classes, and can be accessed only through the methods of their current class. Abstract class called Shape has three subclasses say Triangle,Rectangle,Circle. Method area() in the abstract class and override this area() in these three subclasses to calculate for specific object i.e.area() of Triangle subclass should calculate area of triangle etc. Same for Rectangle and Circle. An abstract class is like a blueprint/format about the minimum required functions.A method which is declared as abstract and does not have implementation is known as an abstract method

```
abstract class Shape
{
    double s1;
    double s2;
    Shape(double l, double b)
    {s1 = l;
    s2 = b;}
    abstract void area();
}
class Circle extends Shape
{
    Circle(double r, double y)
    {
        super(r,y);
    }
    void area()
    {
        double a = 3.14*s1*s1;
        System.out.println("Area of Circle "+a);
    }
}
class Rec extends Shape
{
    Rec(double l, double b)
    {super(l,b);}
    void area()
    {
        double a1 = s1*s2;
        System.out.println("Area of rectangle "+a1);
    }
}
class Tri extends Shape
{
    Tri(int h, int b)
```

```
{super(h,b);}  
void area()  
{  
    double a2 = 0.5*s1*s2;  
    System.out.println("Area of Triangle "+a2);  
}  
}  
public class AbEx  
{  
public static void main(String args[])  
{  
Circle c = new Circle(10,0);  
Rec r = new Rec(6,7);  
Tri t = new Tri(2,20);  
c.area();  
r.area();  
t.area();  
}  
}
```

## Output :

```
PS C:\Coding And Programming\javac\lg> javac .\AbEx.java  
PS C:\Coding And Programming\javac\lg> java AbEx  
Area of Circle 314.0  
Area of rectangle 42.0  
Area of triangle 20.0  
PS C:\Coding And Programming\javac\lg> █
```

## Experiment 10 : To implement Inheritance, interfaces and method overriding (CO4)

i. WAP to implement three classes namely Student, Test and Result. Student class has member as roll no, Test class has members as sem1\_marks and sem2\_marks and Result class has member as total. Create an interface named sports that has a member score (). Derive Test class from Student and Result class has multiple inheritances from Test and Sports. Total is formula based on sem1\_marks, sem2\_mark and score.

### Theory:

In this program we created an interface named sports which consists of score function and created 3 classes namely student, text by extending the student class and Result by extending student class and implementing the interface sports. Lastly, we created class multiple and executed all the functions. Inheritance in Java is a mechanism in which one object acquires all the properties and behaviors of a parent object. It is an important part of OOPs (Object Oriented programming system). An interface is a reference type, similar to a class, that can contain only constants, method signatures, default methods, static methods, and nested types.

```
interface Sports
{
    int score = 50;
    void m_score();
}

class Student
{
    int roll;
    void input(int n)
    {roll = n;}
    void print()
    {System.out.println("Roll no : "+roll);}
}

class Test extends Student
{
    int s1,s2;
    void input(int n, int m)
    {s1=n;
    s2=m;}
    void print()
    {System.out.println("total mrk : "+(s1+s2));}
}

class Result extends Test implements Sports
{
    //int r;
    public void m_score()
    {System.out.println("Total : "+(s1+s2+score));}
}

class IhIt
{
    public static void main(String args[])
    {
```

```
Result q = new Result();
q.input(100);
q.print();
q.input(23,25);
q.print();
q.m_score();
}
}
```

## Output:

```
PS C:\Coding And Programming\javac1g> javac .\IhIt.java
PS C:\Coding And Programming\javac1g> java IhIt
total mrk : 0
total mrk : 48
Total : 98
PS C:\Coding And Programming\javac1g> █
```

## Experiment 11 : To implement Package (C02)

i. WAP to create a user defined package & import the package in another program.

### Theory:

Package in Java is a mechanism to encapsulate a group of classes, sub packages and interfaces. To create package in Java: - First create a directory within the name of the package. - Create a java file in the newly created directory. - In this java file you must specify the package name with the help of package keyword. - Save this file with same name of public class Note: only one class in a program can declare as public.

Below code is in Javacfg named folder :

```
package Javacfg;  
public class User  
{  
  
    public static void name()  
    {  
        System.out.println("Hello This is package program !!!");  
    }  
}
```

Below program using Javacfg package :

```
import Javacfg.User;  
  
class User2  
{  
    public static void main(String args[])  
    {  
        User.name();  
    }  
}
```

Output :

```
PS C:\Coding And Programming\javacfg> javac User2.java  
PS C:\Coding And Programming\javacfg> java User2  
Hello This is package program !!!  
PS C:\Coding And Programming\javacfg> █
```

## Experiment 12 : To implement exceptions in Java (C05)

i. Write a Java Program to input the data through command Line and Find out total valid and invalid integers. (Hint: use exception handling)

### Theory:

In Java, an exception is an event that disrupts the normal flow of the program. It is an object which is thrown at runtime. The core advantage of exception handling is to maintain the normal flow of the application. An exception normally disrupts the normal flow of the application; that is why we need to handle exceptions. The `java.lang.Throwable` class is the root class of Java Exception hierarchy inherited by two subclasses: `Exception` and `Error`. Java Exception Handling in which we are using a try-catch statement to handle the exception.

```
import java.util.*;

class Exception_hand
{
    public static void main(String args[])
    {
        try
        {
            int n = Integer.parseInt(args[0]);
            System.out.println("User entered : "+n);
        }
        catch(Exception e)
        {
            System.out.println("Exception occurred : "+e);
        }
    }
}
```

### Output:

```
PS C:\Coding And Programming\javac\lg> javac .\Exception_hand.java
PS C:\Coding And Programming\javac\lg> java Exception_hand 123
User entered : 123
PS C:\Coding And Programming\javac\lg> java Exception_hand hello
Exception occurred : java.lang.NumberFormatException: For input string: "hello"
PS C:\Coding And Programming\javac\lg> █
```

ii. Write a Java Program to calculate the Result. Result should consist of name, seatno, date, center number and marks of semester three exam. Create a User Defined Exception class `MarksOutOfBoundsException`, If Entered marks of any subject is greater than 100 or less than 0, and then program should create a user defined Exception of type `MarksOutOfBoundsException` and must have a provision to handle it.



### Theory:

In Java, we can create our own exceptions that are derived classes of the Exception class. Steps to create custom Exception Handling : Create a new class whose name should end with an Exception like MarksOutOfBoundsException .This is a convention to differentiate an exception class from regular ones. Make the class extends one of the exceptions which are subtypes of the java.lang.Exception class. Generally, a custom exception class always extends directly from the Exception class. Create a constructor with a String parameter which is the detailed message of the exception. In this constructor, simply call the super constructor and pass the message

```
import java.util.*;
import java.io.*;
class MarksOutOfBoundException extends Exception
{
MarksOutOfBoundException(String err)
{
System.out.println(err);
}
}
public class Exp12
{
public static void main(String args[])
{
Scanner input = new Scanner(System.in);
int m,m2,m3,seatNo,centerNum,choice=1;
String name,date;
while(choice == 1)
{
try{
System.out.println("Enter the Seat Number : ");
seatNo =input.nextInt();
String str1 = input.nextLine();
System.out.println("Enter Name of Student : ");
name = input.nextLine();
System.out.println("Enter the Center Number : ");
centerNum =input.nextInt();
String str =input.nextLine();
System.out.println("Enter Date : ");
date = input.nextLine();
System.out.println("Enter the Marks in Maths : ");
m = input.nextInt();
System.out.println("Enter the Marks in Chemistry : ");
m2 = input.nextInt();
System.out.println("Enter the Marks in Physics : ");
m3 = input.nextInt();
Marks(seatNo,centerNum,date,name,m,m2,m3);
}
catch(Exception e)
{
System.out.println(e);
}
System.out.println("\nEnter your choice : \n1.Enter more Student data \n2.Exit
");
choice = input.nextInt();
}
}
```

```

}
public static void Marks(int seatNo , int centerNo ,String
date,String name , int marks ,int marks2,int marks3) throws
MarksOutOfBoundsException
{
if(marks >= 100 || marks <= 0)
{
throw new MarksOutOfBoundsException("Input marks of all subjects should be
greater than 0 and less than 100");
}
else if(marks2 >= 100 || marks2 <= 0)
{
throw new MarksOutOfBoundsException("Input marks of all subjects should be
greater than 0 and less than 100");
}
else if(marks3 >= 100 || marks3 <= 0)
{
throw new MarksOutOfBoundsException("Input marks of all subjects should be
greater than 0 and less than 100");
}
else{
System.out.println("\nStudent Details :\nName : " + name + "\nSeat Number: " +
seatNo + "\nCenter Number : " + centerNo + "\nDate : " + date);
System.out.println("Marks inMaths : " + marks + "\nMarks in physics : " +
marks2 +
"\nMarks in chemistry: " + marks3 );
}
}
}
}

```

**Output:**

```
PS C:\Coding And Programming\javac\lg> javac .\Exp12.java
PS C:\Coding And Programming\javac\lg> java Exp12
Enter the Seat Number :
115
Enter Name of Student :
Stud1
Enter the Center Number :
225
Enter Date :
3-3-2022
Enter the Marks in Maths :
50
Enter the Marks in Chemistry :
50
Enter the Marks in Physics :
50

Student Details :
Name : Stud1
Seat Number: 115
Center Number : 225
Date : 3-3-2022
Marks in Maths : 50
Marks in physics : 50
Marks in chemistry: 50

Enter your choice :
1.Enter more Student data
2.Exit
1
Enter the Seat Number :
116
Enter Name of Student :
Stud2
Enter the Center Number :
226
Enter Date :
3-2-2022
Enter the Marks in Maths :
50
Enter the Marks in Chemistry :
50
Enter the Marks in Physics :
100
Input marks of all subjects should be greater than 0 and less than 100
MarksOutOfBoundException

Enter your choice :
1.Enter more Student data
2.Exit
█
```

## Experiment 13 : To implement Multithreading (C05)

i. Write java program to print Table of Five, Seven and Thirteen using Multithreading (Use Thread class for the implementation). Also print the total time taken by each thread for the execution

### Theory:

Multi-threading enables you to write in a way where multiple activities can proceed concurrently in the same program. By definition, multitasking is when multiple processes share common processing resources such as a CPU. Multi-threading extends the idea of multitasking into applications where you can subdivide specific operations within a single application into individual threads. Each of the threads can run in parallel. The OS divides processing time not only among different applications, but also among each thread within an application.

```
class Five extends Thread {
public void run() {
long start = System.currentTimeMillis();
for (int i = 1; i < 10; i++) {
System.out.println("5*" + i + "=" + i * 5);
try {
// milliseconds time
Thread.sleep(2000);
} catch (InterruptedException e) {
}
}
long end = System.currentTimeMillis();
System.out.println("Total time taken by 5 table:" + (end - start));
}
}

class Seven extends Thread {
public void run() {
long start = System.currentTimeMillis();
for (int i = 1; i < 10; i++) {
System.out.println("7*" + i + "=" + i * 7);
try {
// milliseconds time
Thread.sleep(2000);
} catch (InterruptedException e) {
}
}
long end = System.currentTimeMillis();
System.out.println("Total time taken by 7 table:" + (end -
start));
}
}

class Thirteen extends Thread {
public void run() {
long start = System.currentTimeMillis();
for (int i = 1; i < 10; i++) {
System.out.println("13*" + i + "=" + i * 13);
try {
// milliseconds time
```

```

Thread.sleep(2000);
} catch (InterruptedException e) {
}
}
}
long end = System.currentTimeMillis();
System.out.println("Total time taken by 13 table:" + (end -
start));
}
}
public class MultiThread{
public static void main(String[] args) {
Five f = new Five();
Seven s = new Seven();
Thirteen t = new Thirteen();
f.start();
s.start();
t.start();
}
}
}

```

## Output:

```

PS C:\Coding And Programming\javac\lg> & 'C:\Users\Acer\AppData\Local\Programs\Eclipse Adoptium\jdk-17.0.4.101-hotspot\bin\java.exe' '-XX:+ShowCodeDetailsInExceptionMessages' '-cp' 'C:\Users\Acer\AppData\Roaming\Code\User\workspaceStorage\3c4c4f775fb2e598a4fec7679beda268\redhat.java\jdt_ws\javac\lg_9864129f\bin' 'MultiThread'
5*1=5
13*1=13
7*1=7
5*2=10
13*2=26
7*2=14
5*3=15
13*3=39
7*3=21
5*4=20
13*4=52
7*4=28
5*5=25
13*5=65
7*5=35
5*6=30
13*6=78
7*6=42
5*7=35
13*7=91
7*7=49
5*8=40
13*8=104
7*8=56
5*9=45
13*9=117
7*9=63
Total time taken by 5 table:18061
Total time taken by 13 table:18077
Total time taken by 7 table:18093
PS C:\Coding And Programming\javac\lg>

```

## 2) Write java program to implement the concept of Thread Synchronization

### Theory:

When we start two or more threads within a program, there may be a situation when multiple threads try to access the same resource and finally they can produce unforeseen result due to concurrency issues. The function `Thread.sleep()` is used so that it sleeps a thread for the specified amount of time. Till the time another thread is running. The function `isAlive()` is used so that it tests if the thread is alive. (It returns a boolean value).

```
class Movie2 extends Thread
{
    int v=1,r;
    Movie2(int x)
    {r=x;}
    public synchronized void run()
    {
        if(r<=v){System.out.println(Thread.currentThread().getName());
            try{
                Thread.sleep(1000);
            }catch(Exception e){}
            v = v-r;
        }
        else
        {
            System.out.println("Housefull");
        }
    }
}

class Movie1
{
    public static void main(String args[])
    {
        Movie2 m =new Movie2(1);
        Thread t1 = new Thread(m);
        Thread t2 = new Thread(m);
        t1.setName("p1");t2.setName("p2");
        t1.start();t2.start();
    }
}
```

### Output:

```
PS C:\Coding And Programming\javac\lg> javac .\Movie1.java
PS C:\Coding And Programming\javac\lg> java Movie1.java
error: can't find main(String[]) method in class: Movie2
PS C:\Coding And Programming\javac\lg> java Movie1
p1
Housefull
PS C:\Coding And Programming\javac\lg> █
```

## Experiment 14 : Designing Graphical User Interfaces in Java using AWT and Event handling (C06)

i. Write java program to create a registration form using AWT.

### Theory:

Java AWT (Abstract Window Toolkit) is an API to develop Graphical User Interface (GUI) or windows-based applications in Java. Java AWT components are platform-dependent i.e. components are displayed according to the view of operating system. AWT is heavy weight i.e. its components are using the resources of underlying operating system (OS). The java.awt package provides classes for AWT API such as TextField, Label, TextArea, RadioButton, CheckBox, Choice, List etc. The AWT tutorial will help the user to understand Java GUI programming in simple and easy steps

```
import java.awt.*;
import javax.swing.*;
import java.awt.event.*;

class Log2 extends JFrame implements ActionListener
{
    Container c;
    JLabel lblUserName, lblPassword;
    JTextField txtUserName;
    JPasswordField txtPassword;
    JButton btnSubmit, btnClear, btnExit;
    String strUserName, strPassword;
    Log2()
    {
        c= getContentPane();
        c.setLayout(new FlowLayout());
        lblUserName = new JLabel("User Name: ");
        lblPassword = new JLabel("Password: ");
        txtUserName = new JTextField(10);
        txtPassword = new JPasswordField(10);
        txtPassword.setEchoChar('*');
        btnSubmit = new JButton("Submit");
        btnClear = new JButton("Clear");
        btnExit = new JButton("Exit");
        c.add(lblUserName);
        c.add(txtUserName);
        c.add(lblPassword);
        c.add(txtPassword);
        c.add(btnSubmit);
        c.add(btnClear);
    }
}
```

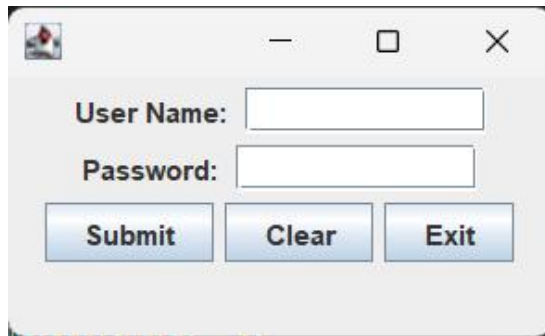
```

        c.add(btnExit);
        btnSubmit.addActionListener(this);
        btnClear.addActionListener(this);
        btnExit.addActionListener(this);
    }
    public void actionPerformed(ActionEvent ae)
    {
        if(ae.getSource() == btnSubmit)
        {
            strUserName = txtUserName.getText();
            strPassword = txtPassword.getText();
            if(strUserName.equals("Carlos31") &&
strPassword.equals("Carlos31!"))
            {
                JOptionPane.showMessageDialog(c,"Successful Login");
                System.exit(0);
            }
            else
            {
                JOptionPane.showMessageDialog(c,"Unsuccessful Login");
                txtUserName.setText("");
                txtPassword.setText("");
                txtUserName.requestFocus();
            }
        }
        else if(ae.getSource() == btnClear)
        {
            txtUserName.setText("");
            txtPassword.setText("");
            txtUserName.requestFocus();
        }
        else
        {
            System.exit(0);
        }
    }
}
public static void main(String z[])
{
    Log2 frm = new Log2();
    frm.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    frm.setBounds(200,200,250,150);
    frm.setVisible(true);
}
}

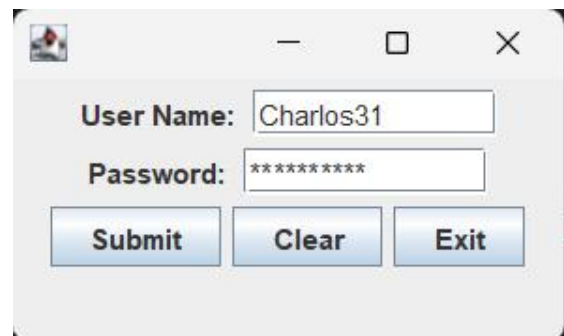
```



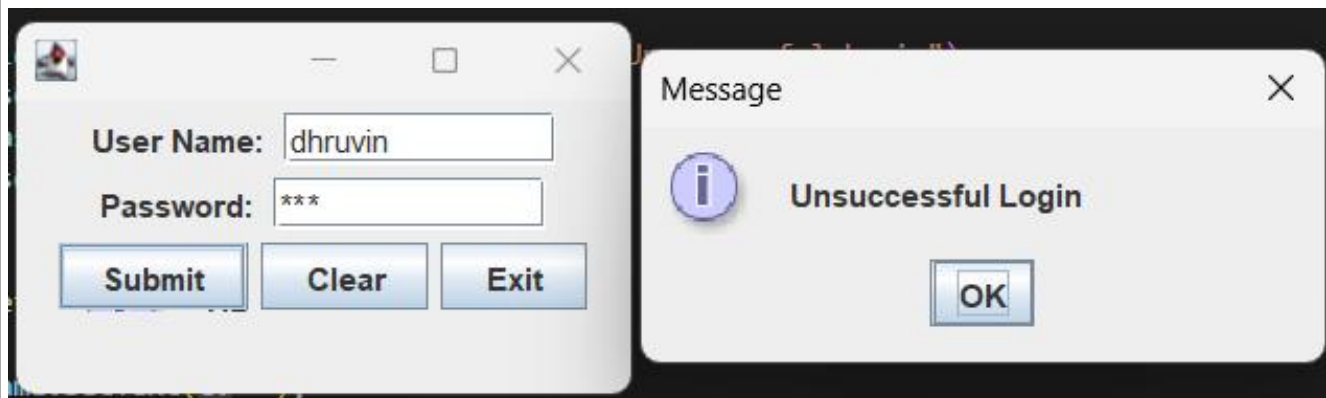
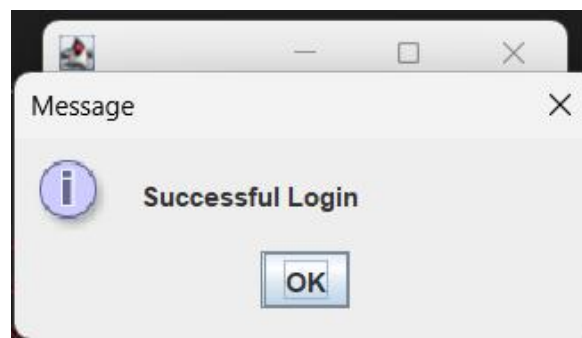
## Output:



A screenshot of a login form window. It has a title bar with a small icon, a minus button, a maximize button, and a close button. The form contains two text input fields: "User Name:" and "Password:". Below the fields are three buttons: "Submit", "Clear", and "Exit".



A screenshot of the same login form window, but with the "User Name:" field containing the text "Charlos31" and the "Password:" field containing a series of asterisks "\*\*\*\*\*". The "Submit", "Clear", and "Exit" buttons are still present.



## Experiment 15 : Develop simple swing applications and complex GUI using Java Swing classes. (C06)

i. Write a program to create a window with four text fields for the name, street, city and pin code with suitable labels. Also windows contains a button MyInfo. When the user types the name, his street, city and pincode and then clicks the button, the types details must appear in Arial Font with Size 32, Italics.

### Theory:

Swing in Java is a lightweight GUI toolkit which has a wide variety of widgets for building optimized window based applications. It is a part of the JFC( Java Foundation Classes). It is build on top of the AWT API and entirely written in java. It is platform independent unlike AWT and has lightweight components. It becomes easier to build applications since we already have GUI components like button, checkbox etc. This is helpful because we do not have to start from the scratch. Different methods of JFrame class are : public void add(Component c) : Inserts a component on this component. public void setSize(int width,int height) : Sets the size (width and height) of the component. public void setLayout(LayoutManager m) : Defines the layout manager for the component. public void setVisible(boolean status) : Changes the visibility of the component, by default false. An object of the java.awt.Font class represents a font in a Java program

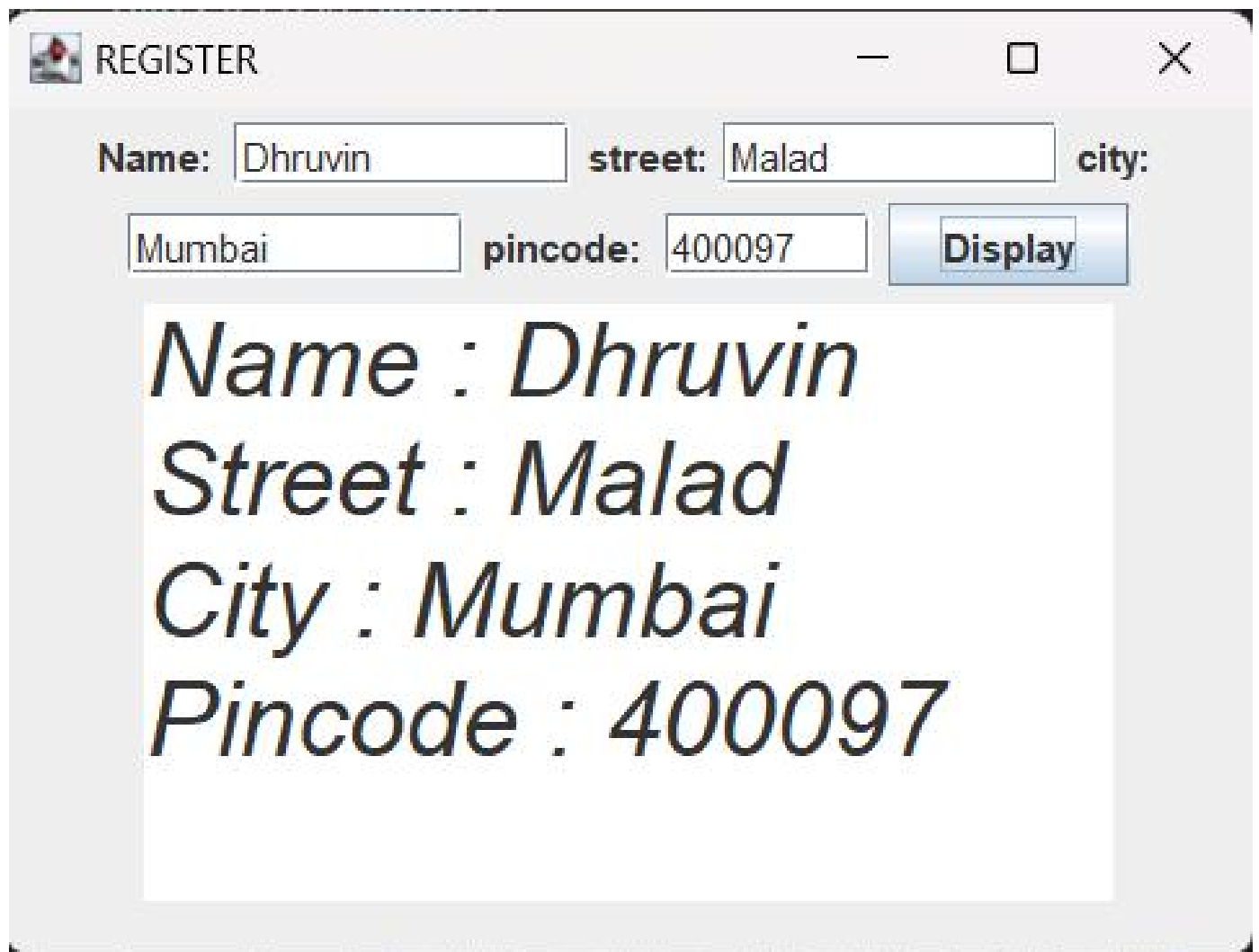
```
import java.awt.*;
import javax.swing.*;
import java.awt.event.*;
class SwingJava extends JFrame implements ActionListener {
    Container c;
    JLabel name, street, city, pincode;
    JTextField tname, tstreet, tcity, tpincode;
    JTextArea tout;
    JButton MyInfo, btnClear, btnExit;
    String sname, sstreet, scity, spincode;
    SwingJava() {
        c = getContentPane();
        c.setLayout(new FlowLayout());
        // label
        name = new JLabel("Name: ");
        street = new JLabel("street:");
        city = new JLabel("city: ");
        pincode = new JLabel("pincode: ");
        // text fields
        tname = new JTextField(10);
        tstreet = new JTextField(10);
        tcity = new JTextField(10);
        tpincode = new JTextField(6);
        // buttons
        MyInfo = new JButton("Display");
        c.add(name);
        c.add(tname);
        c.add(street);
        c.add(tstreet);
        c.add(city);
        c.add(tcity);
```

```

c.add(pincod);
c.add(tpincod);
c.add(MyInfo);
tout = new JTextArea();
tout.setFont(new Font("Arial", Font.ITALIC, 32));
tout.setSize(300, 400);
tout.setLocation(100, 500);
tout.setLineWrap(true);
tout.setEditable(false);
c.add(tout);
MyInfo.addActionListener(this);
}
public void actionPerformed(ActionEvent e) {
if (e.getSource() == MyInfo) {
sname = tname.getText();
scity = tcity.getText();
sstreet = tstreet.getText();
spincod = tpincod.getText();
if (sname.equals("") || sstreet.equals("") ||
scity.equals("") ||
spincod.equals("")) {
JOptionPane.showMessageDialog(c, "Input field is empty !!");
tname.requestFocus();
} else {
String data = "Name : " + tname.getText() + "\n" +
"Street : " + tstreet.getText() + "\n" + "City : "
+ tcity.getText() + "\n" + "Pincod : " +
tpincod.getText() + "\n";
tout.setText(data);
tout.setEditable(false);
}
} else if (e.getSource() == btnClear)
{
tname.setText("");
tpincod.setText("");
tstreet.setText("");
tcity.setText("");
tname.requestFocus();
} else {
System.exit(0);
}
}
public static void main(String[] args)
{
SwingJava frm = new SwingJava();
frm.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
frm.setBounds(200, 200, 400, 300);
frm.setVisible(true);
frm.setTitle("REGISTER");
}
}

```

Output :



REGISTER

Name:  street:  city:

pincode:

*Name : Dhruvin*  
*Street : Malad*  
*City : Mumbai*  
*Pincode : 400097*

ii. WA applet with 4 swing buttons with suitable texts on them. When the user presses a button a message should appear in the label as to which button was pressed by the user

Theory:

Interface ActionListener : The listener interface for receiving action events. The class that is interested in processing an action event implements this interface, and the object created with that class is registered with a component, using the component's addActionListener method. When the action event occurs, that object's actionPerformed method is invoked. The EventObject contains the getSource() method. Suppose you have many buttons in your application. So, you can find which button is clicked by using the getSource() method. The getSource() method returns the source of the event. Java CollationElementIterator setText(String source) Set a new string over which to iterate.

```
import java.awt.*;
import javax.swing.*;
import java.awt.event.*;
public class Buttons extends JFrame implements ActionListener {
    Container c;
    JButton b1, b2, b3, b4;
    JLabel label;
    Buttons() {
        c = getContentPane();
        c.setLayout(null);
        b1 = new JButton("Java");
        b2 = new JButton("NodeJS!!");
        b3 = new JButton("Python");
        b4 = new JButton("Django!!");
        label = new JLabel(" ");
        label.setSize(200, 60);
        b1.setLocation(100, 50);
        b2.setLocation(100, 110);
        b3.setLocation(100, 170);
        b4.setLocation(100, 230);
        b1.setSize(100, 50);
        b3.setSize(100, 50);
        b4.setSize(100, 50);
        b2.setSize(100, 50);
        c.add(b1);
        c.add(b2);
        c.add(b3);
        c.add(b4);

        c.add(label);
        b1.addActionListener(this);
        b3.addActionListener(this);
        b4.addActionListener(this);
        b2.addActionListener(this);
    }
    public static void main(String[] args) {
        Buttons frm = new Buttons();
        frm.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frm.setBounds(200, 200, 400, 500);
        frm.setVisible(true);
    }
}
```

```
frm.setTitle("clickme");
}
public void actionPerformed(ActionEvent e) {
if (e.getSource() == b1) {
label.setText("Java");
label.setLocation(220, 50);
}
if (e.getSource() == b2) {
label.setText("NodeJS!!");
label.setLocation(220, 110);
}
if (e.getSource() == b3) {
label.setText("Python");
label.setLocation(220, 160);
}
if (e.getSource() == b4) {
label.setText("Django!!");
label.setLocation(220, 230);
}
}
}
```

Output :

