Programming lab Experiments

Name: Dishant Zaveri Sap Id: 60004200090

Batch: A/A2 Branch: S.E Computer Engineering (2021-22)

Experiment 1:

1.i) To implement Java Program Structures & Simple Programs
WAP to display hello Message on screen.

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.

CODE:

```
public class exp1_1
   {
      public static void main(String[] args) {
      System.out.println("Hello Sir");
   }
}
```

Output :

Hello Sir

1.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.

Theory :

In this program we have to take input from command line like after we compile program while running the program we have to provide the integer of which digits count we want
To find the digits count will be stored in main class args[] and store that in integer called num. The **Scanner** class is used to get user input, and it is found in the java.util package.
To use the Scanner class, create an object of the class and use any
of the available methods found in the Scanner class
documentation

Code:

```
public class Dishant
{
	public static void main(String args[]){
	int num = Integer.parseInt(args[0]);
	int count=0;
	while(num>0){
	num /=10;
	count++;
}
System.out.println(count);
}
}
```

Output:

```
java Dishant 12456
5
```

EXPERIMENT 2 : To implement Java control statements and loops

2.i) WAP to find roots of a Quadratic equation. Take care of imaginary values.

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:
$ax2 + bx + c = 0$
When b2−4ac=0 there is one real root.
When b2−4ac>0 there are two real roots.
When b2−4ac<0 there are two complex roots.

Code:

```
import java.util.Scanner;
public class Roots {
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();
}
6
}
```

Output:

```
Enter a b and c
10 20 5
Root 1=-0.2928932188134524
Root 2=-1.7071067811865475
```

2.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
}
```

Code:

```
import java.util.Scanner;
public class MathematicalOperation
{
   public static void main(String args[])
   {
      int a,b,ch;
      Scanner sc = new Scanner(System.in);
```

```
        System.out.println("1.Addition\n 2.subtraction\n 3.Multiplication\n 4.Modulus\n
5.Division\n");

        System.out.println("Enter the values of a&b: ");
        a = sc.nextInt();
        b = sc.nextInt();
        System.out.println("Enter Your Choice: ");
        ch=sc.nextInt();
       switch(ch)
       {
        case 1:
            int result=a+b;
            System.out.println("sum of A and B is "+ result);
            break;
        case 2:
            int result1=a-b;
            System.out.println("Subtraction of A and B is" +result1);
            break;
        case 3:
            int result2=a*b;
            System.out.println("Multiplication of A and B is" +result2);
            break;
        case 4:
            int result3=a%b;
            System.out.println("Modulud of A and B is" +result3);
            break;
        case 5:
            int result4=a/b;
            System.out.println("Divisionof A and B is" +result4);
            break;
        default:
            System.out.println("Invalid choice Please try again: ");
                }
   }
}
```

Output:

```
1.Addition
 2.subtraction
 3.Multiplication
 4.Modulus
 5.Division
Enter the values of a & b:
10
5
Enter Your Choice:
3
Multiplication of A and B is 50
```

2.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 divde that number to verify is it prime last we ger 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.

Code:

```
import java.util.*;
public class oddandprime{
public static void main(String args[])
{
        Scanner sc = new Scanner(System.in);
        int x;
        System.out.println("Enter the number");
        x = sc.nextInt();
        System.out.println("The odd number between 1 to "+x +"is");
        for(int i=1;i<=x;i++){
        if(i%2!=0){
        System.out.print(i+"    ");
        }
    }
        System.out.println("The prime number between 1 to "+x+"is");
        int i,j,chk;
        for(i=2; i<=x; i++)
    {
       chk = 0;
       for(j=2; j<i; j++)
       {
          if(i%j==0)
          {
             chk++;
             break;
          }
       }
       if(chk==0)
          System.out.print(i+"   ");
```

```
    }
} }
```

Output:

```
Enter the number
50
The odd number between 1 to 50is
1   3   5   7   9   11   13   15   17   19   21   23   25   27   29   31   33   35
37   39   41   43   45   47   49
The prime number between 1 to 50 is
2   3   5   7   11   13   17   19   23   29   31   37   41   43   47
```

2.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

Code:

```
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:
```

2.v)WAP to display the following patterns:

1
2 1
1 2 3
4 3 2 1
1 2 3 4 5
6 5 4 3 2 1
1 2 3 4 5 6 7

Theory:

In this program we have taken number of rows as input the pattern in this is quite thinkable here the odd number of rows have the number written in straight format and even number of rows have number in reverse format using for loops we can print the pattern .
Nested loop means a loop statement inside another loop statement. That is why nested loops are also called as “loop inside loop“.
Syntax for Nested For loop:

```
for ( initialization; condition; increment ) {
for ( initialization; condition; increment ) {
// statement of inside loop
}
// statement of outer loop}
```

Code:

```
import java.util.*;
public class pattern1{
public static void main(String args[])
{
        Scanner sc = new Scanner(System.in);
        int x;
        System.out.println("Enter the number of rows");
```

```
        x = sc.nextInt();
        for(int i=1;i<=x;i++){
        if(i%2 !=0){
        for(int j=1;j<=i;j++){
        System.out.print(j+" ");
        }
        }
        else{
        for(int j=i;j>=1;j--){
        System.out.print(j+" ");
}
}
        System.out.println("\n");
} } }
```

Output :

```
Enter the number of rows
7

1
2 1
1 2 3
4 3 2 1
1 2 3 4 5
6 5 4 3 2 1
1 2 3 4 5 6 7
```

```
    A
  CB
 FED
JIHG
```

Theory :

In this program we have to print a alphabetical pattern where we have to print alphabet starting from last position with white spaces include in it and the pattern white pattern should go on decreasing row by row and alphabets should go on increasing and should print upto the given number of rows.

Code:

```
import java.util.*;
public class pattern2{
public static void main(String args[])
{
        Scanner sc = new Scanner(System.in);
        int num,temp=65,alpha,temp1=1;
        num = sc.nextInt();
```

```
        int count = num -1;
        for(int i =1;i<=num;i++){
                for(int k =1;k<=count;k++)
{
        System.out.print(" ");
}
        alpha = temp;
        for(int j=1;j<=i;j++){
        System.out.print((char)(alpha));
        alpha--;
}
        temp = temp + (++temp1);
        System.out.println("\n");
        count--;
}
}
}
```

Output:

```
4
     A

    CB

   FED

  JIHG
```

Experiment 3: To implement Arrays

3.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.

Code:

```
import java.util.*;
public class Main {
public static void main(String[] args) {
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||nums[0
]*nums[3]==num)
{
System.out.println(num+" is vampire");
}
else
{
System.out.println(num+" is not vampire");
}
}
}
```

Output:

1200
1200 is not vampire
1260
1260 is vampire

3.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

Code:

```
public class JaggedArray {
public static void main(String[] args) {
int arr[][] = new int[3][];
arr[0] = new int[1];
arr[1] = new int[2];
arr[2] = new int[3];
int count = 1;
for (int i = 0; i < arr.length; i++) {
for (int j = 0; j < arr[i].length; j++) {
arr[i][j] = count++;
}
}
System.out.println("Irregular Array");
for (int i = 0; i < arr.length; i++) {
for (int j = 0; j < arr[i].length; j++) {
System.out.print(arr[i][j] + " ");
}
System.out.println();
}
}
}
```

Output:

Irregular Array
1
2 3
4 5 6

3.iii)

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).

Code:

```
import java.util.*;
public class Main {
public static void main(String[] args) {
Scanner sc = new Scanner(System.in);
System.out.print("Enter rows of matrix : ");
int r = sc.nextInt();
System.out.print("Enter colums of matrix : ");
int c = sc.nextInt();
int matrix[][] = new int[r][c];
for (int i = 0; i < r; i++)
for (int j = 0; j < c; j++)
matrix[i][j] = sc.nextInt();
int sumOfRow[] = new int[r];
Arrays.fill(sumOfRow, 0);
int sumOfCol[] = new int[c];
Arrays.fill(sumOfCol, 0);
```

```
int grandTotal = 0;
for (int i = 0; i < r; i++) {
for (int j = 0; j < c; j++) {
sumOfRow[i] += matrix[i][j];
sumOfCol[j] += matrix[i][j];
}
grandTotal += sumOfRow[i];
}
for (int i = 0; i < r; i++) {
for (int j = 0; j < c; j++) {
System.out.print(matrix[i][j] + "\t");
}
System.out.print("| " + sumOfRow[i] + " \n");
}
for (int j = 0; j < c; j++) {
System.out.print("----\t");
}
System.out.println("----");
for (int j = 0; j < c; j++) {
System.out.print(sumOfCol[j] + "\t");
}
System.out.print("| " + grandTotal + "\t");
}
}
```

Output:

```
Enter rows of matrix : 4
Enter colums of matrix : 4
1
2
3
4
1
2
3
4
1
2
3
4
1
2
3
4
1	2	3	4	| 10
1	2	3	4	| 10
1	2	3	4	| 10
1	2	3	4	| 10
----	----	----	----	----
4	8	12	16	| 40
```

Experiment 4: To implement Vectors

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<E> v = new Vector<E>(); // default vector of the initial capacity is 10.
Vector<E> v = new Vector<E>(int size); // Initial capacity is specified by size.
Vector<E> v = new Vector<E>(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

Code:

```
import java.util.*;
public class Main {
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 : a
Enter the item 2 : b
Enter the item 3 : c
Enter the item 4 : 4
            Operation
        1)add at specific location
        2)Delete item
        3)Display list
        4)Exit
Enter the choice : 1
Enter the Location(index) : 2
Enter the item : 3
Enter the choice : 2
Enter the item want to delete : c
Enter the choice : 3
The elements in List are : [a, b, 3, 4]
Enter the choice :
```

4.ii) 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

Code:

```
import java.util.*;
class Main {
public static void main(String args[]) {
int i, n, x, count = 0;
Scanner scan = new Scanner(System.in);
System.out.print("Enter the size of the vector array:");
n = scan.nextInt();
System.out.println("Enter " + n + " elements:");
Vector v = new Vector();
for (i = 0; i < n; i++) {
    System.out.print("Enter the element for position " + (i + 1) + ":");
v.addElement(scan.nextInt());
}
System.out.println("Vector array:" + v);
System.out.print("Enter the element whose frequency to be known:");
x = scan.nextInt();
int startpos=0,newpos;
while(startpos<v.size())
{
newpos=v.indexOf(x,startpos);
if(newpos==-1)
{
break;
}
count++;
startpos=newpos+1;
}
System.out.print("Frequency:" + count);
}
}
```

Output:

```
Enter the size of the vector array:6
Enter 6 elements:
Enter the element for position 1:1
Enter the element for position 2:4
Enter the element for position 3:3
```

```
Enter the element for position 4:5
Enter the element for position 5:3
Enter the element for position 6:4
Vector array:[1, 4, 3, 5, 3, 4]
Enter the element whose frequency to be known:4
Frequency:2
```

Experiment 5:To implement Strings

5.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.
Algorithm (if reqd): 1. Check if both strings are of equal length or not, if not return false. 2. Otherwise, start comparing both strings and count number of unmatched characters and also store the index of unmatched characters. 3. If unmatched characters are more than 2 then return false. 4. Otherwise check if on swapping any of these two characters in any string would make the string equal or not. 5. If yes then return true. Otherwise return false.

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.

Code:

```
import java.util.*;
public class Main {
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.charAt(wrong
_pos[1])==word2.charAt(wrong_pos[0]))
```

```
return true;
}
return false;
}
public static void main(String[] args) {
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:

```
dishant
sidhant
Strings are meta
anand
naadn
Strings are not meta
```

5.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 currect character or if it is the first character of the string.

Code:

```
import java.util.*;
class Main {
```

```
public static void main(String args[]) {
int v = 0;
int c = 0;
String str;
Scanner scan = new Scanner(System.in);
System.out.println("Enter a sentence:");
str = scan.nextLine();
count(str);
word(str);
str = str.toLowerCase();
for (int i = 0; i < str.length(); i++) {
char ch = str.charAt(i);
if (ch == 'a' || ch == 'e' || ch == 'i' || ch == 'o' || ch == 'u') {
v++;
} else if ((ch >= 'a' && ch <= 'z')) {
c++;
}
}
System.out.println("No.of vowels:" + v);
System.out.println("No.of consonants:" + c);
}
public static void count(String x) {
char[] ch = x.toCharArray();
int letter = 0;
int space = 0;
int num = 0;
int symbol = 0;
int i;
for (i = 0; i < x.length(); i++) {
if (Character.isLetter(ch[i])) {
letter++;
} else if (Character.isDigit(ch[i])) {
num++;
} else if (Character.isSpaceChar(ch[i])) {
space++;
} else {
symbol++;
}
}
System.out.println("No. of alphabets:" + letter);
System.out.println("No. of digits:" + num);
System.out.println("No. of special symbols:" + symbol);
System.out.println("No. of blank spaces:" + space);
}
public static void word(String str) {
int word = 0;
char ch[] = new char[str.length()];
for (int i = 0; i < str.length(); i++) {
ch[i] = str.charAt(i);
if (((i > 0) && (ch[i] != ' ') && (ch[i - 1] == ' ')) || ((ch[0] != ' ') && (i == 0))) {
```

```
word++;
}
}
System.out.println("No.of words:" + word);
}
}
```

Output:

```
Enter a sentence:
Hi , my name isDish@123
No. of alphabets:14
No. of digits:3
No. of special symbols:2
No. of blank spaces:4
No.of words:5
No.of vowels:5
No.of consonants:9
```

Experiment 6:To implement Functions, recursive functions, and Overloading

6.i) 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.

Code:

```
class Overload
{
   void area(float x)
   {
      System.out.println("the area of the square is "+Math.pow(x, 2)+" sq units");
   }
   void area(float x, float y)
   {
      System.out.println("the area of the rectangle is "+x*y+" sq units");
   }
}
class Main
{
    public static void main(String args[])
        {
          Overload ob = new Overload();
          ob.area(5);
          ob.area(11,12);
      }
}
```

Output:
the area of the square is 25.0 sq units
the area of the rectangle is 132.0 sq units

6.ii) Write menu driven program to implement recursive functions for following tasks.
To find GCD and LCM
To find X^Y
To print n Fibonacci numbers
To find reverse of number
To 1+2+3+4+…….+ (n-1)+n
Calculate sum of digits of a number

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
}
```

Code:

```
import java.util.Scanner;
public class Rec {
// 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);
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 ");
int 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:

```
1) To find GCD and LCM
2) To find X^Y
3) To print n Fibonacci numbers
43
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:
6 9
GCD is :3
LCM is : 18

2
Enter the x and y value
9 2
Answer:81

3
Enter the nth value
5
0 1 1 2 3

4
Enter the number
235
reverse is 532

5
Enter the nth value
10
1+2+3+4+5+6+7+8+9+10=55

6
```

Enter the number
2345
Sum is 14

Experiment 7: To implement Array of Objects

7.i) WOOP 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.

Code:

```
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 Main {
```

```
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:

```
Enter Details:

Enter student name:dishant
Enter Roll_no:12
Enter Marks:
Physics Marks:87
Chemistry Marks:98
Mathematics Marks:97

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

Enter student name:prem
Enter Roll_no:21
Enter Marks:
Physics Marks:87
```

Chemistry Marks:9
Mathematics Marks:85

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

Enter student name:smit
Enter Roll_no:32
Enter Marks:
Physics Marks:67
Chemistry Marks:98
Mathematics Marks:76

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

Enter student name:sahil
Enter Roll_no:11
Enter Marks:
Physics Marks:76
Chemistry Marks:89
Mathematics Marks:76

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

Enter student name:vivek
Enter Roll_no:15
Enter Marks:
Physics Marks:76
Chemistry Marks:89
Mathematics Marks:77

*****Student details registered*****
Student: dishant ,roll_no: 12 ,marks:
Physics:87
Chemistry:98
Mathematics:97
Total:282
Student: prem ,roll_no: 21 ,marks:
Physics:87
Chemistry:9
Mathematics:85
Total:181
Student: smit ,roll_no: 32 ,marks:
Physics:67
Chemistry:98
Mathematics:76
Total:241
Student: sahil ,roll_no: 11 ,marks:
Physics:76
Chemistry:89
Mathematics:76

Total:241
Student: vivek ,roll_no: 15 ,marks:
Physics:76
Chemistry:89
Mathematics:77
Total:242
Student Marks in Descendin Order:
Student Name: dishant, Student RollNo: 12, Total: 282
Student Name: vivek, Student RollNo: 15, Total: 242
Student Name: smit, Student RollNo: 32, Total: 241
Student Name: sahil, Student RollNo: 11, Total: 241
Student Name: prem, Student RollNo: 21, Total: 181

Experiment 8 : To implement Constructors and overloading

8.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.

Code:

```
import java.util.*;
class Shape{
int s1,s2;
    Shape(int s){
        s1=s;
        s2=s;
    }
     Shape(int l ,int b){
        s1=l;
        s2=b;
    }
    float area(){
        return s1*s2;
    }
}
public class Main {
    public static void main(String args[]){
        Shape square=new Shape(6);
        Shape rect=new Shape(4,5);
        System.out.println("Area of Square of side 6 is " +square.area());
        System.out.println("Area of Rectangle of length 4 and breadth 5 is "
+rect.area());
    }
}
```

Output:

```
Area of Square of side 6 is 36.0
Area of Rectangle of length 4 and breadth 5 is 20.0
```

8.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 voliume. 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.

Code:

```
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 Main{
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:

```
Area of Rectangle : 30
Volume of Cube : 64
Other objects : -1
```

Experiment 9 :To implement Abstract classes

9.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.

Code:

```
import java.lang.Math;

abstract class Shape
{
 abstract void area();
 double area;
}

class Triangle extends Shape
{
 double b=50,h=15;
 void area()
 {
  area = (b*h)/2;
  System.out.println("area of Triangle -->"+area);
 }
}

class Rectangle extends Shape
{
 double w=70,h=20;
 void area()
 {
  area = w*h;
  System.out.println("area of Rectangle -->"+area);
 }
}
```

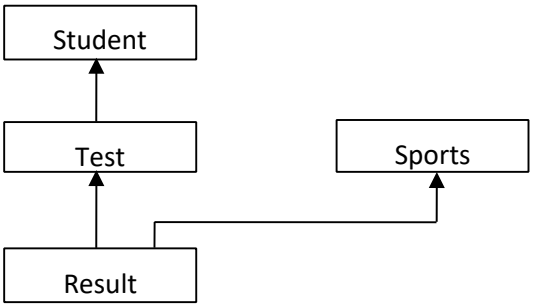
```
class Circle extends Shape
{
 double r=5;
 void area()
 {
  area = Math.PI * r * r;
  System.out.println("area of Circle -->"+area);
 }
}

class Main
{
 public static void main(String [] args)
 {
  Triangle t= new Triangle();
  Rectangle r =new Rectangle();
  Circle c =new Circle();

  t.area();
  r.area();
  c.area();
 }
}
```

Output:

```
area of Triangle -->375.0
area of Rectangle -->1400.0
area of Circle -->78.53981633974483
```

Experiment 10 :To implement Inheritance, interfaces and method Overriding

10.i) WAP to implement three classes namely Student, Test and Result. Student class has member as rollno, 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 and 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.

Code:

```
import java.util.Scanner;

interface Sports{
    int score=100;
    void member_score();
}

class Student{
    int roll_no;
    void read(int n){
        roll_no = n;
    }
    void display(){
        System.out.println(roll_no);
```

```
    }
}
class Test extends Student{
    int sem1_marks,sem2_marks;
    void read1(int n){
        sem1_marks = n;
    }
    void read2(int n){
        sem2_marks = n;
    }
    void display(){
        System.out.println(sem1_marks+sem1_marks);
    }
}

class Result extends Test implements Sports{
    public void member_score(){
        int total;
        total = sem1_marks+sem2_marks+score;
        System.out.println("The total score is "+total);
    }
}

public class Main {
    public static void main(String args[]){
        Scanner s = new Scanner(System.in);
        Result r = new Result();
        System.out.println("Enter roll no.");
        int roll = s.nextInt();
        System.out.println("Enter sem1.");
        int sem1 = s.nextInt();
        System.out.println("Enter sem2.");
        int sem2 = s.nextInt();
        r.read(roll);
        r.read1(sem1);
        r.read2(sem2);
        r.member_score();
    }
}
```

Output:

```
Enter roll no.
45
Enter sem1.
444
Enter sem2.
324
The total score is 868
```

Experiment 11 : To implement Package

11.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.

Code:

```
package Dishant;
public class User {
public String name;
public void msg() {
System.out.println("Hello " + name);
}
}
import Dishant.User;
class Test {
public static void main(String args[]) {
User obj = new User();
obj.name="Dishant";
obj.msg();
}
}
```

Output:

```
Hello Dishant
```

Experiment 12 : To implement exceptions in Java

12.i) Write a Java Program to input the data through command Line and Find out total valid and in-valid 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.

Code:

```
import java.util.Scanner;
public class Main {
public static void main(String[] args) {
Scanner input = new Scanner(System.in);
System.out.println("Enter the number");
try {
int number = input.nextInt();
System.out.println("Number is valid " + number);
} catch (Exception e) {
System.out.println("EXCEPTION :" +e);
}
System.out.println("Program hasn't crashed");
input.close();
}
}
```

Output:

```
Enter the number
56
Number is valid 56
Program hasn't crashed
Enter the number
Abcdefg
EXCEPTION :java.util.InputMismatchException
Program hasn't crashed
```

12.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.

Code:

```
import java.util.*;
import java.io.*;
class MarksOutOfBoundException extends Exception
{
MarksOutOfBoundException(String err)
{
System.out.println(err);
}
}
public class Main{
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 main(int seatNo , int centerNo ,String
date,String name , int marks ,int marks2,int marks3) throws
MarksOutOfBoundException
{
if(marks >= 100 || marks <= 0)
{
throw new MarksOutOfBoundException("Input marks of all subjects should be
greater than 0 and less than 100");
}
else if(marks2 >= 100 || marks2 <= 0)
{
throw new MarksOutOfBoundException("Input marks of all subjects should be
greater than 0 and less than 100");
}
else if(marks3 >= 100 || marks3 <= 0)

{
throw new MarksOutOfBoundException("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:
Enter the Seat Number :

65
Enter Name of Student :
Dishant
Enter the Center Number :
16
Enter Date :
2-01-22
Enter the Marks in Maths :
86
Enter the Marks in Chemistry :
77
Enter the Marks in Physics :
72
96
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

13.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.

Code:

```
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 {

// miliseconds 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 {
// miliseconds 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 {
// miliseconds time
Thread.sleep(2000);
} catch (InterruptedException e) {
}
}
long end = System.currentTimeMillis();
System.out.println("Total time taken by 13 table:" + (end -
start));
}
}
public class Main {
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:

```
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 7 table:18116
Total time taken by 5 table:18109
Total time taken by 13 table:18107
```

13.ii) 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).

Code:

```
class Movie extends Thread {
int vacant = 1, required;
Movie(int x) {
required = x;
}
public synchronized void run() {
if (required <= vacant) {
System.out.println(required + " for  " +
Thread.currentThread().getName());
try {
Thread.sleep(100);
} catch (Exception e) {
}
vacant = vacant - required;
} else {
System.out.println("none for " +
```

```
Thread.currentThread().getName());
}
}
}
class Main {
public static void main(String z[]) {
Movie m = new Movie(1);
Thread t1 = new Thread(m);
Thread t2 = new Thread(m);
t1.setName("Dishant");
t2.setName("Smit");
t1.start();
t2.start();
}
}
```

Output:
1 for Dishant
none for Smit

Experiment 14 : To implement Applets

14.i) Write java program to draw the house on an applet.

Theory:

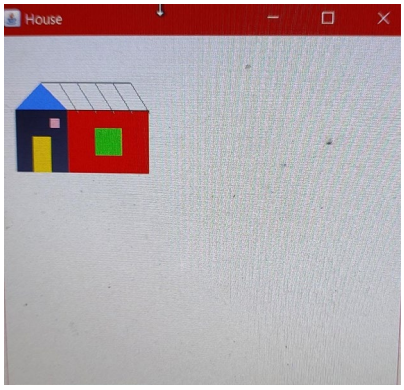
The javax.swing.JFrame class is a type of container which inherits the java.awt.Frame class. JFrame works like the main window where components like labels, buttons, textfields are added to create a GUI.
Unlike Frame, JFrame has the option to hide or close the window with the help of setDefaultCloseOperation(int) method.
Here, I have created a house using VSCode IDE .

Code:

```
import java.awt.Color;
import java.awt.Graphics;
import javax.swing.JFrame;
import javax.swing.JPanel;
public class House {
public static void main(String[] args) {
JFrame jf = new JFrame();
jf.setVisible(true);
jf.setTitle("House");
jf.setSize(300, 300);
jf.setLocation(300, 100);
jf.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
JPanel p = new JPanel() {
@Override
public void paint(Graphics g) {
g.setColor(Color.DARK_GRAY);
g.fillRect(20, 80, 60, 70);
g.setColor(Color.RED);
g.fillRect(80, 80, 90, 70);
g.setColor(Color.PINK);
g.fillRect(60, 90, 10, 10);
g.setColor(Color.yellow);
g.fillRect(40, 110, 20, 40);
g.setColor(Color.BLACK);
g.drawLine(50, 50, 150, 50);
g.setColor(Color.GRAY);
g.drawLine(150, 50, 171, 83);
g.drawLine(110, 50, 131, 83);
g.drawLine(130, 50, 151, 83);
g.drawLine(90, 50, 111, 83);
g.drawLine(70, 50, 91, 83);
g.setColor(Color.BLACK);
g.drawLine(80, 80, 170, 80);
```

```
g.setColor(Color.GREEN);
g.fillRect(110, 100, 30, 30);
g.setColor(Color.BLUE);
g.fillPolygon(new int[] { 50, 20, 80 }, new int[] { 50, 80,
80 }, 3);
}
};
jf.add(p);
}
}
```

Output:



14.ii) Write java program to create an advertisement banner on an applet using multithreading

Theory:

Applets are embeddable Java applications that are expected to start and stop themselves on command, possibly many times in their lifetime. A Java-enabled web browser normally starts an applet when the applet is displayed and stops it when the user moves to another page or (in theory) when the user scrolls the applet out of view. To conform to this API, we would like an applet to cease its nonessential activity when it is stopped and resume it when started again.
An important compromise was made early in the design of Swing relating to speed, GUI consistency, and thread safety. To provide maximum performance and simplicity in the common case, Swing does not explicitly synchronize access to most Swing component methods. This means that most Swing components are, technically, not threadsafe for multithreaded applications.

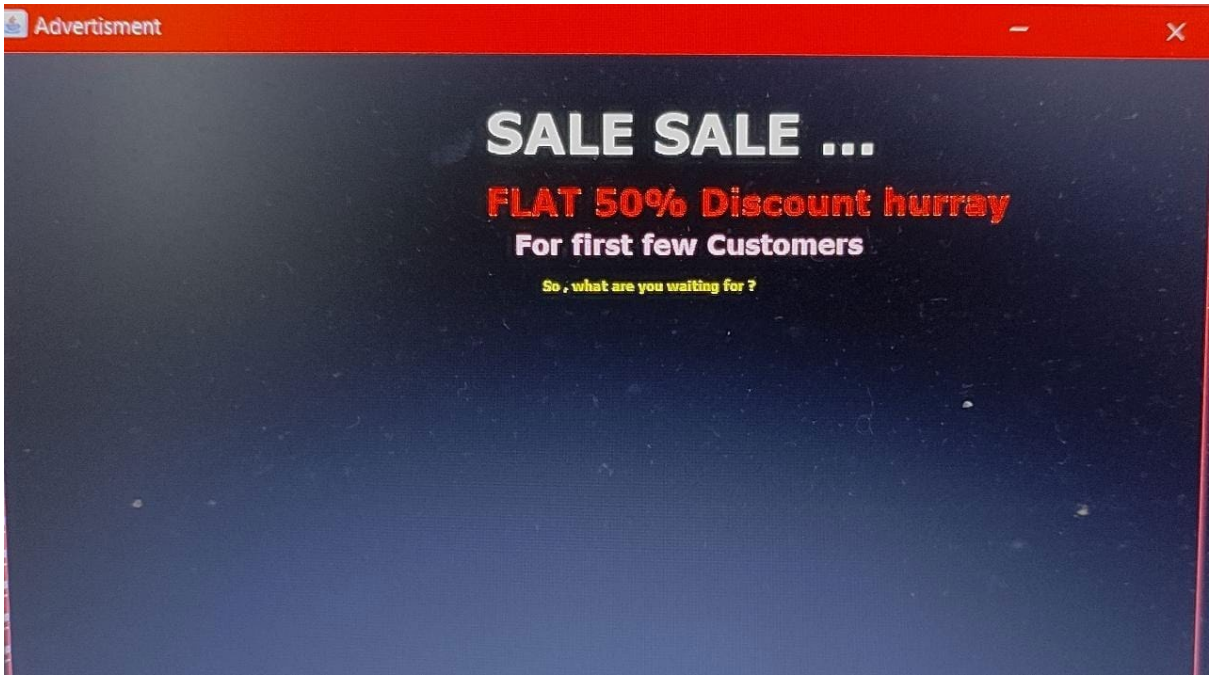
Code:

```
import javax.swing.*;
import java.awt.*;
class MyFrame extends JFrame implements Runnable {
Container c;
```

```
JLabel title, ad,ad1;
public MyFrame() {
// add(new Board());
setTitle("Advertisment");
setBounds(300, 90, 900, 600);
setDefaultCloseOperation(EXIT_ON_CLOSE);
setResizable(false);
setVisible(true);
c = getContentPane();
c.setBackground(Color.black);
c.setLayout(null);
title = new JLabel("SALE SALE SALE !!!");
title.setSize(300, 50);
title.setLocation(350, 30);
title.setForeground(Color.white);
title.setFont(new Font("Verdana", Font.BOLD, 38));
c.add(title);
ad = new JLabel("FLAT 50% Discount hurray");
ad.setSize(400, 30);
ad.setForeground(Color.red);
ad.setFont(new Font("Verdana", Font.BOLD, 25));
ad.setLocation(350, 90);
c.add(ad);
ad1 = new JLabel("For first few Customers");
ad1.setSize(400, 30);
ad1.setForeground(Color.pink);
ad1.setFont(new Font("Verdana", Font.BOLD, 19));
ad1.setLocation(370, 120);
c.add(ad1);
ad1 = new JLabel("So , what are you waiting for ?");
ad1.setSize(400, 30);
ad1.setForeground(Color.yellow);
ad1.setFont(new Font("Verdana", Font.BOLD, 9));
ad1.setLocation(390, 150);
c.add(ad1);
new Thread(this).start();
}
public void run() {
try {
while (true) {
if (title.getText() == null) {
title.setText("SALE SALE SALE !!!");
Thread.sleep(500);
} else {
title.setText(null);
Thread.sleep(500);
}
}
} catch (InterruptedException ex) {
}
```

```
}
}
public class lab {
public static void main(String[] args) {
new MyFrame();
}
}
```

Output:



Experiment 15 :Designing Graphical User Interfaces in Java using AWT and Event handling

15.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

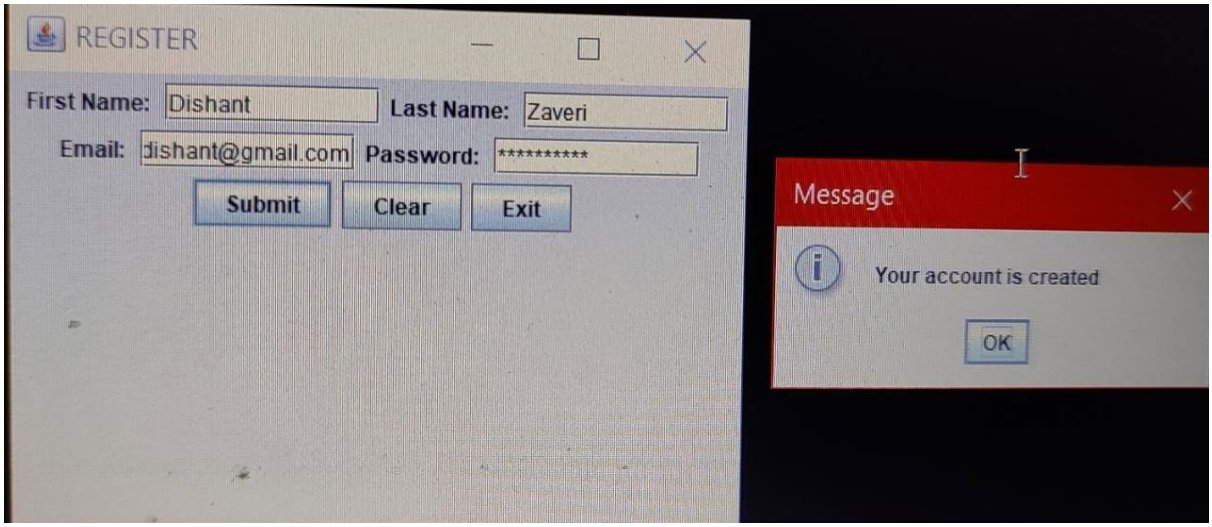
Code:

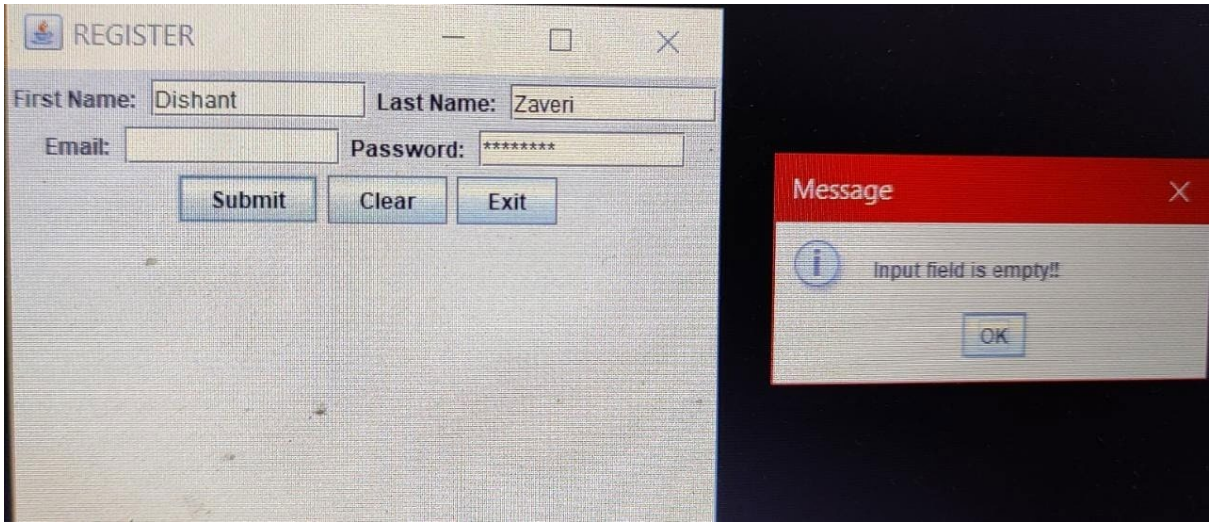
```
import java.awt.*;
import javax.swing.*;
import java.awt.event.*;
class Register extends JFrame implements ActionListener {
Container c;
JLabel lFName, lLName, lPassword, lEmail;
JTextField txtFName, txtLName;
JTextField txtEmail;
JPasswordField txtPassword;
JButton btnSubmit, btnClear, btnExit;
String strFName, strLName, strPassword, strEmail;
Register() {
c = getContentPane();
c.setLayout(new FlowLayout());
// labels
lFName = new JLabel("First Name: ");
lLName = new JLabel("Last Name: ");
lPassword = new JLabel("Password: ");
lEmail = new JLabel("Email: ");
// text fields
txtFName = new JTextField(10);
txtLName = new JTextField(10);
txtEmail = new JTextField(10);
txtPassword = new JPasswordField(10);
txtPassword.setEchoChar('*');
// buttons
btnSubmit = new JButton("Submit");
```

```
btnClear = new JButton("Clear");
btnExit = new JButton("Exit");
c.add(lFName);
c.add(txtFName);
c.add(lLName);
c.add(txtLName);
c.add(lEmail);
c.add(txtEmail);
c.add(lPassword);
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) {
strLName = txtLName.getText();
strFName = txtFName.getText();
strPassword = txtPassword.getText();
strEmail = txtEmail.getText();
if (strFName.equals("") || strLName.equals("") ||
strPassword.equals("") ||
strEmail.equals("")) {
JOptionPane.showMessageDialog(c, "Input field is empty!!");
txtFName.requestFocus();
} else {
JOptionPane.showMessageDialog(c, "Your account is created ");
txtFName.setText("");
txtPassword.setText("");
System.exit(0);
}
} else if (ae.getSource() == btnClear) {
txtFName.setText("");
txtPassword.setText("");
txtLName.setText("");
txtEmail.setText("");
txtFName.requestFocus();
} else {
System.exit(0);
}
}
public static void main(String z[]) {
Register frm = new Register();
frm.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
frm.setBounds(200, 200, 400, 300);
frm.setVisible(true);
frm.setTitle("REGISTER");
```

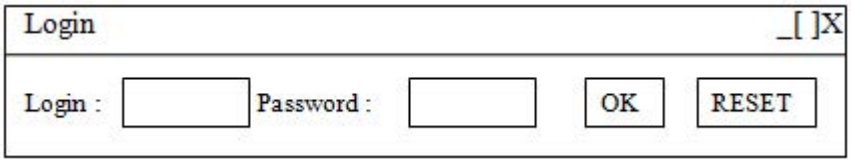
```
}
}
```

Output:





15.ii) On Applet: Take a Login and Password from the user and display it on the third Text Field which appears only on clicking OK button and clear both the Text Fields on clicking RESET button.



Theory:

In this program we have created a login form where used gets logged in if he enters the correct username and password. After OK button is pressed the username and

password is checked, if it matches the actual credentials then used successfully logs in or else it’s an unsuccessful login.
Components Used:

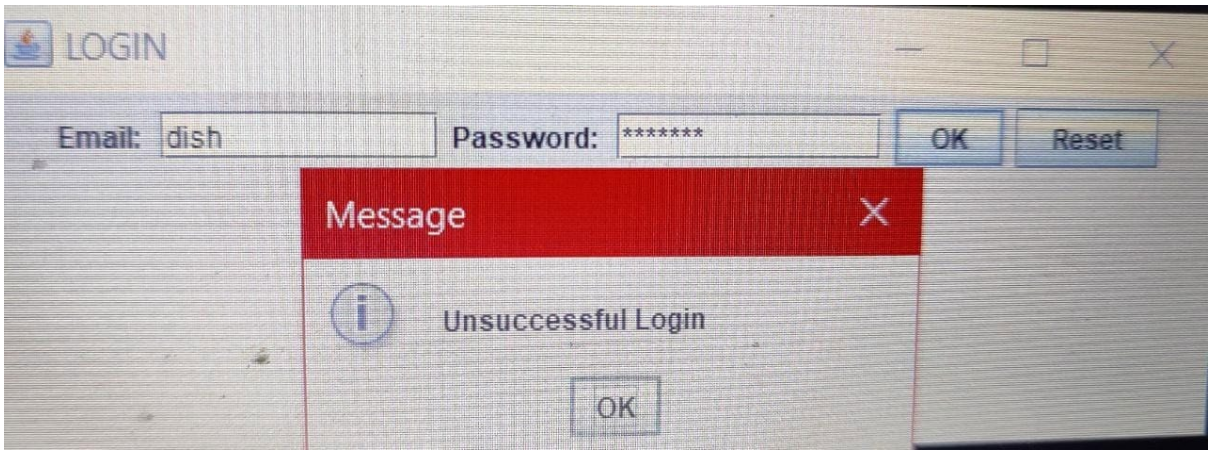
- JFrame
- JTextField
- JPasswordField
- JButton

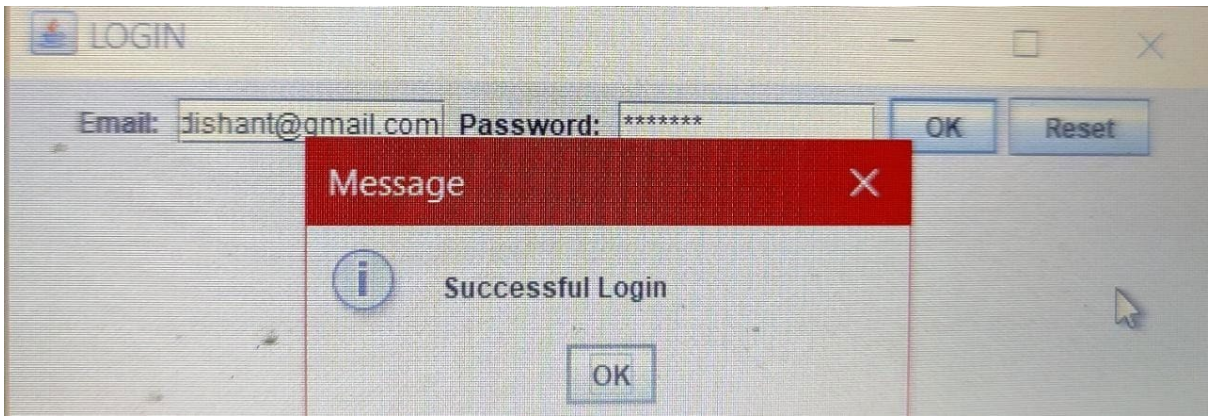
Code:

```
import java.awt.*;
import javax.swing.*;
import java.awt.event.*;
class LoginForm extends JFrame implements ActionListener {
Container c;
JLabel lPassword, lEmail;
JTextField txtEmail;
JPasswordField txtPassword;
JButton btnSubmit, btnClear;
String strPassword, strEmail;
LoginForm() {
c = getContentPane();
c.setLayout(new FlowLayout());
// labels
lPassword = new JLabel("Password: ");
lEmail = new JLabel("Email: ");
// text fields
txtEmail = new JTextField(10);
txtPassword = new JPasswordField(10);
txtPassword.setEchoChar('*');
// buttons
btnSubmit = new JButton("OK");
btnClear = new JButton("Reset");
c.add(lEmail);
c.add(txtEmail);
c.add(lPassword);
c.add(txtPassword);
c.add(btnSubmit);
c.add(btnClear);
btnSubmit.addActionListener(this);
btnClear.addActionListener(this);
}
public void actionPerformed(ActionEvent ae) {
if (ae.getSource() == btnSubmit) {
strPassword = txtPassword.getText();
strEmail = txtEmail.getText();
if (strPassword.equals("Dishant") &&
strEmail.equals("dishant@gmail.com")) {
JOptionPane.showMessageDialog(c, "Successful Login");
System.exit(0);
```

```
} else {
JOptionPane.showMessageDialog(c, "Unsuccessful Login");
txtEmail.setText("");
txtPassword.setText("");
}
} else if (ae.getSource() == btnClear) {
txtPassword.setText("");
txtEmail.setText("");
}
}
public static void main(String z[]) {
LoginForm frm = new LoginForm();
frm.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
frm.setBounds(200, 200, 550, 200);
frm.setVisible(true);
frm.setTitle("LOGIN");
}}
```

Output:





Experiment 16 :Develop simple swing applications and complex GUI using Java Swing classes.

16.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.

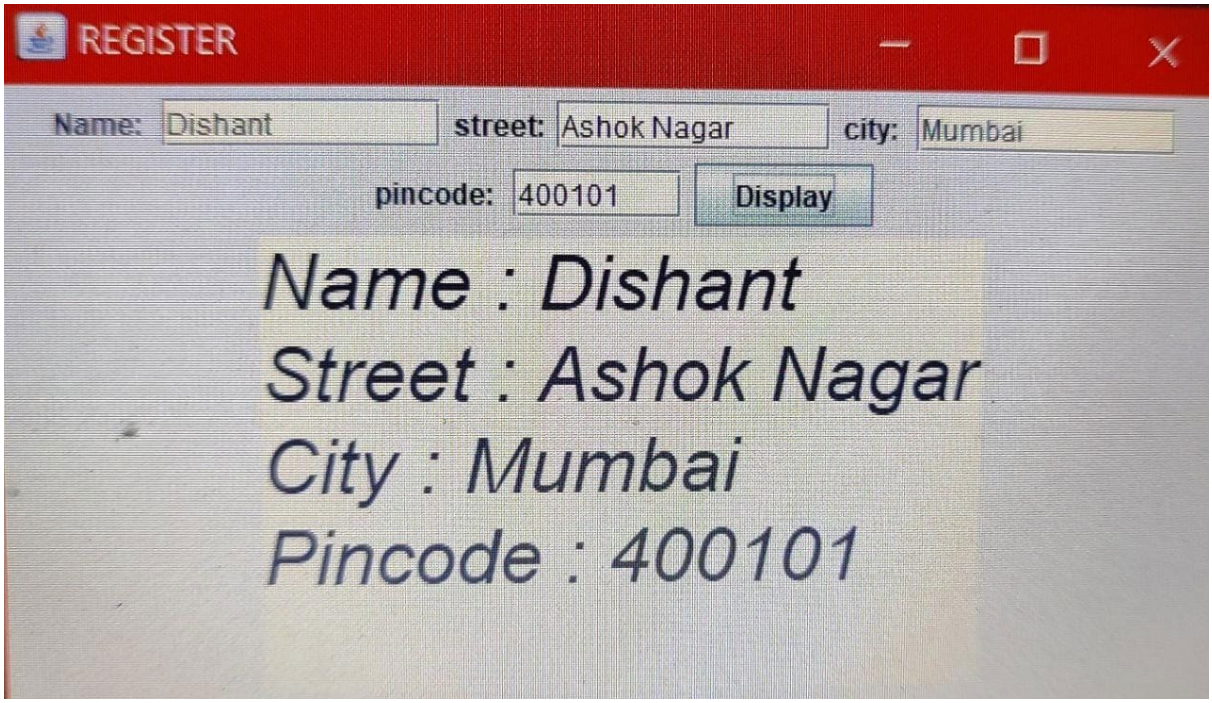
Code:

```
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(pincode);
c.add(tpincode);
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();
spincode = tpincode.getText();
if (sname.equals("") || sstreet.equals("") ||
scity.equals("") ||
spincode.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" + "Pincode : " +
tpincode.getText() + "\n";
tout.setText(data);
tout.setEditable(false);
}
} else if (e.getSource() == btnClear)
{
tname.setText("");
tpincode.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:



16.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.

Code:

```
import java.awt.*;
import javax.swing.*;
import java.awt.event.*;
public class Buttons extends JFrame implements ActionListener {
Container c;
JButton btnHi, btnHowareyou, btnDishant, btnGladtomeet;
JLabel label;
Buttons() {
c = getContentPane();
c.setLayout(null);
btnHi = new JButton("Hi");
btnDishant = new JButton("Dishant");
btnGladtomeet = new JButton("Gladtomeet");
btnHowareyou = new JButton("Howareyou?");
label = new JLabel(" ");
label.setSize(200, 60);
btnHi.setLocation(100, 50);
btnDishant.setLocation(100, 110);
btnGladtomeet.setLocation(100, 170);
btnHowareyou.setLocation(100, 230);
btnHi.setSize(100, 50);
btnDishant.setSize(100, 50);
btnGladtomeet.setSize(100, 50);
btnHowareyou.setSize(100, 50);
c.add(btnHi);
c.add(btnDishant);
c.add(btnGladtomeet);
c.add(btnHowareyou);
c.add(label);
btnHi.addActionListener(this);
btnDishant.addActionListener(this);
btnGladtomeet.addActionListener(this);
btnHowareyou.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() == btnHi) {
label.setText("Hi");
label.setLocation(220, 50);
}
if (e.getSource() == btnDishant) {
label.setText("Dishant?");
label.setLocation(220, 110);
}
if (e.getSource() == btnGladtomeet) {
label.setText("Gladtomeet");
label.setLocation(220, 160);
}
if (e.getSource() == btnHowareyou) {
label.setText("Howareyou");
label.setLocation(220, 230);
}
}
}
```

Output: