

PRATHAM BHOIR

60004200082

Div- A/A2

COMPUTER ENGINEERING

JAVA EXPERIMENTS

AIM 1: To implement Java Program Structures and Simple Programs

1.1) WAP to display hello message on screen.

Theory: In Java, System.out.println() is used to print a statement which has been passed in its argument. There are 2 printing statements in Java, the first being System.out.print() which prints the argument passed through it on the same line and the second being System.out.println() which is similar to System.out.print() method except that it moves the cursor to the next line after printing the result.

Code:

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

Output:

Hello there

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

Theory: The while loop in Java is a control flow statement that allows code to be executed repeatedly based on a given boolean condition. The loop goes on until the boolean condition turns false. When the number of iterations is not known to the user, they can use the while loop

Code:

```
import java.util.Scanner;

public class Main {

    public static void main(String z[]) {
        Scanner s = new Scanner(System.in);
        int count = 0;
        System.out.println("Enter a number");
        long num = s.nextLong();
        while(num != 0)
        {
            num = num/10;
            ++count;
        }
        System.out.println("Number of digits in the given integer is: " +
        count);
    }
}
```

Output:

Enter a number

6143

Number of digits in the given integer is: 4

AIM 2: To implement Java control statements and loops

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

Theory:

Java has the following conditional statements:

- **if** to specify a block of code to be executed, if a specified condition is true
- **else** to specify a block of code to be executed, if the same condition is false
- **else if** to specify a new condition to test, if the first condition is false

Code:

```
import java.util.*;

public class Main {
    public static void main(String z[])
    {
        Scanner s = new Scanner(System.in);
        double a ,b , c ;
        double root1, root2;
        System.out.println("Enter the values of coefficients a , b and c : " );
        a = s.nextDouble();
        b= s.nextDouble();
        c= s.nextDouble();
```

```

double delta = b * b - 4 * a * c;

if (delta > 0) {
    root1 = (-b + Math.sqrt(delta)) / (2 * a);
    root2 = (-b - Math.sqrt(delta)) / (2 * a);
    System.out.println("root1 = " + root1);
    System.out.println("root2 = " + root2);
}
else if (delta == 0)
{
    root1 = root2 = -b / (2 * a);
    System.out.println("root1 = root2 = " + root1);
}
else {
    double real = -b / (2 * a);
    double imaginary = Math.sqrt(-delta) / (2 * a);
    System.out.println("root1 = " + real + " + i" + imaginary);
    System.out.println("root2 = " + real + " - i" + imaginary);
}
}
}

```

Output:

Enter the values of coefficients a , b and c :

2 5 6

root1 = -1.25 + i1.1989578808281798

root2 = -1.25 - i1.1989578808281798

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

Theory: Java has another conditional statement called **switch**. We use switch to specify many alternative blocks of code to be executed. The value for a case must be a constant or a literal. Variables are not allowed. The **break** statement is used inside the switch to terminate a statement sequence.

Code:

```
import java.util.Scanner;

public class Main {

    public static void main(String z[])
    {
        char choice;

        int x, y;

        Scanner s = new Scanner(System.in);

        System.out.println("Enter any two numbers : ");

        x = s.nextInt();

        y = s.nextInt();

        System.out.println("Enter a choice : ");

        System.out.println("+ for Addition");

        System.out.println("- for Subtraction");

        System.out.println("* for Multiplication");

        System.out.println("/ for Division");

        choice = s.next().charAt(0);

        switch(choice)
        {
```

```
case '+':
System.out.println("The sum of " + x + " and " + y + " is " + (x+y));
break;
case '-':
System.out.println("The difference of " + x + " and " + y + " is " + (x-y));
break;
case '*':
System.out.println("The product of " + x + " and " + y + " is " + (x*y));
break;
case '/':
System.out.println("The quotient of " + x + " and " + y + " is " +
(double)((double)x/(double)y));
break;
}
}
}
```

Output:

Enter any two numbers:

5

4

Enter a choice :

+ for Addition

- for Subtraction

* for Multiplication

/ for Division

+

The sum of 5 and 4 is 9

2.3) WAP to display odd numbers from given range/ prime numbers from given range

Theory: In Java, for loop is used to iterate a part of the program several times. If a user knows the number of iterations, then it is recommended to use a for loop. The for statement consumes the initialization, condition and increment/decrement in one line thereby providing a shorter, easy to debug structure of looping.

Code:

```
import java.util.Scanner;

public class Main {

    public static void main(String z[])
    {
        Scanner s = new Scanner(System.in);

        int start,end,i , count , n;

        System.out.println("Enter the start of range : ");
        start = s.nextInt();

        System.out.println("Enter the end of range : ");
        end = s.nextInt();

        System.out.println("The odd numbers in the range is :");
        for(i=start ; i<=end ; i++)
        {
            if(i%2 != 0){
                System.out.println(i);
            }
        }

        System.out.println("The prime numbers in the range is :");
```

```
for(i=start ; i<=end ;i++)
{
count=0;
for(n = i; n>=1 ; n--)
{
    if (i%n == 0)
    {
        count = count +1;
    }
}
if(count == 2)
{
System.out.println(i);
}
}
}
```

Output:

Enter the start of range :

3

Enter the end of range :

20

The odd numbers in the range is :

3

5

7
9
11
13
15
17
19

The prime numbers in the range is :

3
5
7
11
13
17
19

2.4) WAP to display default value of primitive data types

Theory:

Primitive data types are the building blocks of data manipulation. These are the most basic data types available in Java language. 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.*;

public class Main {

    static int a ;
    static double b ;
    static float c;
    static byte d ;
    static short e ;
    static long f;
    static boolean g;
    static char h;

    public static void main(String z[]) {
        Scanner scan = new Scanner(System.in);
        System.out.println("Default value of Integer = " + a);
        System.out.println("Default value of Double = " + b);
        System.out.println("Default value of Float = " + c);
        System.out.println("Default value of Byte = " + d);
        System.out.println("Default value of Short = " + e);
        System.out.println("Default value of Long = " + f);
        System.out.println("Default value of Boolean = " + g);
```

```
System.out.println("Default value of Char = " + h);  
scan.close();  
}  
}
```

Output:

Default value of Integer = 0

Default value of Double = 0.0

Default value of Float = 0.0

Default value of Byte = 0

Default value of Short = 0

Default value of Long = 0

Default value of Boolean = false

Default value of Char =

2.5) 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

A

C B

F E D

J I H G

Theory:

If a loop exists inside the body of another loop, it's called a nested loop. That is why nested loops are also called as “loop inside loop”.

Code:

```
public class Main {  
    public static void main(String args[])  
    {  
        int i,j ,ch='A';  
        for(i=1;i<=7;i++)
```

```

{
    if(i%2!=0)
    {
        for(j=1;j<=i;j++)
        {
            System.out.print(j);
        }
    }
    else
    {
        for(j=i;j>=1;j--)
        {
            System.out.print(j);
        }
    }
    System.out.println();
}

for(i=1;i<=4;i++)
{
    for(j=4-i;j>=1;j--)
    {
        System.out.print(" ");
    }
    for(j=i;j>0;j--)
    {
        System.out.print((char)(ch-1+j));
    }
}

```

```
        ch+=i; System.out.println();  
    }  
}  
}
```

Output:

```
1  
21  
123  
4321  
12345  
654321  
1234567
```

```
    A  
  C B  
F E D  
J I H G
```


AIM 3: To implement Arrays

3.1) 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:

An array is a container object that holds a fixed number of values of a single type. The length of an array is established during the creation of the array. It is used to store multiple values in a single variable instead of declaring multiple variables for each value.

Code:

```
import java.util.*;

public class Main
{
    public static void main(String[] args)
    {
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter a four digit number: ");
        int i,j,k,l,temp,num;
        num = sc.nextInt(); temp=num;
        int [] arr = new int[4];
        for(i=0;i<4;i++)
        {
```

```

arr[i] = temp%10; temp/=10;
}
for(i=0;i<4;i++)
{
for(j=0;j<4;j++)
{
if(i!=j)
{
for(k=0;k<4;k++)
{
if(k!=i && k!=j)
{
for(l=0;l<4;l++)
{
if(l!=i && l!=j && l!=k)
{
if((10*arr[i]+arr[j])*(10*arr[k]+arr[l])==num)
{
System.out.print(num+" is a Vampire Number");
System.exit(0);
}
}
}
}
}
}
}
}
}
}
}
}
}
}

```

```
}  
    System.out.print(num + " is not a Vampire Number");  
}  
}
```

Output:

Enter a four digit number: 6880

6880 is a Vampire Number

3.2) WAP to display the following using irregular arrays

1

2 3

4 5 6

Theory:

Irregular arrays or jagged arrays is a group of arrays where each array can be of different sizes. We can create a 2D array with variable number of columns in each row.

Code:

```
public class Main{  
    public static void main(String[] args)  
    {  
        int r = 3;  
        int arr[][] = new int[r][];  
        for (int i = 0; i < arr.length; i++)  
            arr[i] = new int[i + 1];  
  
        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("The given pattern is displayed below ");
```

```
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:

The given pattern is displayed below

1

2 3

4 5 6

3.3) 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 , columns total and grand total

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

1	3	6		10
7	9	8		24
<hr/>				
8	12	14		34

Theory: Multidimensional arrays is an array of arrays. Each element of a multidimensional array is an array itself. The total number of elements that can be stored in a multidimensional array can be calculated by multiplying the size of all the dimensions.

Code:

```
import java.util.*;

public class Main{

public static void main(String args[])

{

Scanner in = new Scanner(System.in);

int row , col , i , j ,sum=0;

System.out.println("Enter the number of rows : ");

row = in.nextInt();

System.out.println("Enter the number of columns : ");
```

```

col = in.nextInt();
int[][] arr = new int[(row+1)][(col+1)]; for(i=0;i<row+1;i++)
{
    for(j=0;j<col+1;j++)
    {
        if(j != col && i != row )
        {
            System.out.println("Enter element value : ");
            arr[i][j] = sc.nextInt();
            sum = sum+ arr[i][j];
        }
        else
        {
            arr[i][j] = 0;
        }
    }
}
for(i=0;i<row+1;i++)
{
    for(j=0;j<col+1;j++)
    {
        if(j != col && i != row )
        {
            arr[i][col] = arr[i][col] + arr[i][j];
            arr[row][j] = arr[row][j] + arr[i][j];
        }
        else

```

```
{
    arr[row][col] = sum;
}
}
}
for(i=0;i<row+1;i++)
{
    for(j=0;j<col+1;j++)
    {
        System.out.print(arr[i][j] + "\t");
    }
    System.out.println(" ");
}
}
}
```

Output:

Enter the number of rows :

3

Enter the number of columns :

2

Enter element value :

7

Enter element value :

12

Enter element value :

45

Enter element value :

90

Enter element value :

70

Enter element value :

23

Enter element value :

67

7 12 19

45 90 135

70 23 93

122 125 247

AIM 4: To implement Vectors

4.1) 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:

The Vector class implements a growable array of objects i.e. it can grow and shrink as per the requirements of the user. Unlike array, we can store n-number of elements in it as there is no size limit.

Code:

```
import java.util.*;

public class Main {
    public static void main(String z[])
    {
        Scanner in= new Scanner(System.in);
        System.out.print("How many items are there in List : ");
        int n =in.nextInt();
        Vector v = new Vector(n,3);
        for(int i=0;i<n;i++)
        {
            System.out.printf("Enter item %d : ",(i+1));

            String str = in.next();
            v.addElement(str);
        }
    }
}
```

```
System.out.println("Choose an action to perform");
System.out.println("1. Add an item ");
System.out.println("2. Delete an item ");
System.out.println("3. Display all items ");
System.out.println("4. Exit");
int choice = 1;
while(choice != 3)
{
    System.out.println("Enter your choice : ");
    choice = in.nextInt();
    if(choice == 1)
    {
        System.out.println("Enter the location of the item to be added : ");
        int pos = in.nextInt();
        System.out.println("Enter the item : ");
        String item = in.next();
        v.add((pos-1),item);
        System.out.println("Elements in the list are " + v );
    }
    else if(choice == 2 )
    {
        System.out.println("Enter the item to be deleted : ");
        String items = in.next();
        v.remove(items);
    }
    else if(choice == 3)
    {
```

```
System.out.println("Items in the list are " + v );  
}  
else  
{  
System.out.println("Wrong choice");  
}  
}  
}  
}
```

Output:

How many items are there in List :

4

Enter item 1: wheat

Enter item 2: tea

Enter item 3: ketchup

Enter item 4: salt

Enter your choice :

3

Items in the list are [wheat,tea,ketchup,salt]

4.2) Write a java program to find the frequency of an element in the given Vector array.

Theory:

Vectors in java supports constructors and methods.

Some of the constructors are

- `vector()`-It constructs an empty vector with the default size as 10.
- `vector(int initialCapacity)`- It constructs an empty vector with the specified initial capacity and with its capacity increment equal to zero.
- `vector(int initialCapacity, int capacityIncrement)`- It constructs an empty vector with the specified initial capacity and capacity increment.
- `Vector(Collection<? extends E> c)`- It constructs a vector that contains the elements of a collection c.

Some of the methods of vector class are:

- `add()`-It is used to append the specified element in the given vector.
- `size()`-It is used to get the number of components in the given vector.
- `remove()`-It is used to remove the specified element from the vector. If the vector does not contain the element, it is unchanged.
- `get()`-It is used to get an element at the specified position in the vector.

Code:

```
import java.util.*;

public class Main {

    public static void main(String[] args) {

        int no, i;

        Scanner sc = new Scanner(System.in);

        System.out.print("No of Elements in Vector : ");

        int n = sc.nextInt();

        Vector v = new Vector(n);
```

```

    for (i = 0; i < n; i++) {
        no = i + 1;
        System.out.print("element " + no + " : ");
        String element = sc.next();
        v.addElement(element);
    }
    System.out.println("Vector : " + v);
    System.out.print("Find frequency of element : ");
    String z = sc.next();
    int count = Collections.frequency(v, z);
    System.out.println("Find frequency of element " + z + " is " + count);
}
}

```

Output:

No of Elements in Vector :

9

element 1 :

18

element 2 :

17

element 3 :

69

element 4 :

81

element 5 :

12

element 6 :

90

element 7 :

85

element 8 :

22

element 9 :

17

Vector : [18, 17, 69, 81, 12, 90, 85, 22, 17]

Find frequency of element : 17

Find frequency of element 17 is 2

AIM 5: To implement Strings

5.1) 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:

string is basically an object that represents sequence of char values. An array of characters works same as Java string.

Example:

```
char[] ch={'H','a','r','s','h'};
```

```
String s=new String(ch);
```

Is the same as

```
String s="Harsh";
```

Code:

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

```
public class Main {
```

```
    static String swap(String str, int i, int j) {
```



```

        char arr[] = str.toCharArray();
        char temp = arr[i];
        arr[i] = arr[j];
        arr[j] = temp;
        return String.valueOf(arr);
    }

    public static void main(String[] args) {
        Scanner in = new Scanner(System.in);
        String str1, str2, temp;
        int i, j, n = 0;
        System.out.print("Enter first word: ");
        str1 = in.nextLine();
        System.out.print("Enter second word: ");
        str2 = in.nextLine();
        if (str1.length() == str2.length()) {
            for (i = 0; i < str1.length(); i++) {
                for (j = 0; j < str1.length(); j++) {
                    temp = swap(str1, i, j);
                    if (temp.equals(str2)) {
                        System.out.println("They are a pair of Meta strings.");
                        n = 1;
                        break;
                    }
                }
            }
            if (n == 1) {
                break;
            }
        }
    }

```

```
        }  
    }  
    if (n == 0) {  
        System.out.println("They are not a pair of Meta strings.");  
    }  
}  
}
```

Output:

Enter first word: geeks

Enter second word : keegs

They are a pair of Meta strings

5.2) 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:

Java string class has many useful methods to perform operations on sequence of char values. Some of the are listed below:

- `int length()`-It returns string length
- `boolean contains(CharSequence s)`- It returns true or false after matching the sequence of char value.
- `String toUpperCase()`-Converts all of the characters in this String to upper case using the rules of the default locale.
- `String toLowerCase()`-Converts all of the characters in this String to lowercase using the rules of the default locale.

Code:

```
import java.util.*;

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

        String line;
        Scanner in = new Scanner(System.in);
        System.out.print("\nEnter the string : ");
        line = in.nextLine();
        int vowels = 0, consonants = 0, digits = 0, spaces = 0, symb = 0;
        line = line.toLowerCase();
        for (int i = 0; i < line.length(); ++i) {
```

```
char ch = line.charAt(i);
if (ch == 'a' || ch == 'e' || ch == 'i' || ch == 'o' || ch == 'u') {
    ++vowels;
} else if ((ch >= 'a' && ch <= 'z')) {
    ++consonants;
} else if (ch >= '0' && ch <= '9') {
    ++digits;
} else if (ch == ' ') {
    ++spaces;
} else
    ++symb;
}
```

```
System.out.println("Vowels: " + vowels);
System.out.println("Consonants: " + consonants);
System.out.println("Digits: " + digits);
System.out.println("Blank spaces: " + spaces);
System.out.println("Special Symbols : " + symb);
}
}
```

Output:

Enter the string : Hello world.This is a string

Vowels: 7

Consonants: 16

Digits: 0

Blank spaces: 4

Special Symbols : 1

AIM 6: To implement Functions, recursive functions and overloading

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

Theory:

If a class has multiple methods having same name but different in parameters, it is known as function overloading. Suppose you have to perform addition of the given numbers but there can be any number of arguments, if you write the method such as a(int,int) for two parameters, and b(int,int,int) for three parameters then it may be difficult for you as well as other programmers to understand the behavior of the method because its name differs. So, we perform method overloading to figure out the program quickly.

Code:

```
import java.util.*;

class Main {
    float sq;
    float rect;

    void area(float x) {
        sq = x * x;
        System.out.println("Area of Square is " + sq + " sq units ");
    }

    void area(float l, float b) {
        rect = b * l;
    }
}
```

```

        System.out.println("Area of Rectangle is " + rect + " sq units");
    }

    public static void main(String args[]) {
        Main obj = new Main();
        System.out.println("Enter length of square : ");
        Scanner in = new Scanner(System.in);
        float s = in.nextFloat();
        System.out.println("Enter the length and breadth of Rectangle : ");
        float len = in.nextFloat();

        float br = in.nextFloat();
        obj.area(s);
        obj.area(len, br);
    }
}

```

Output:

Enter length of square :

10

Enter the length and breadth of Rectangle :

13

28

Area of Square is 100.0 sq units

Area of Rectangle is 364.0 sq units

6.2) Write a menu driven program to implement recursive functions for following tasks.

- i. To find GCD and LCM**
- ii. To find X^Y**
- iii. To print n Fibonacci numbers**
- iv. To find reverse of number**
- v. To $1+2+3+4+\dots+(n-1)+n$**
- vi. Calculate sum of digits of a number**

Theory:

Recursion is the technique of making a function call itself. This technique provides a way to break complicated problems down into simple problems which are easier to solve. Adding two numbers together is easy to do, but adding a range of numbers is more complicated. Here recursion comes into the picture. It makes the overall program compact.

Code:

```
import java.util.*;

public class App {

    public static int gcd(int n1, int n2) {
        if (n2 != 0)
            return gcd(n2, n1 % n2);

        else
            return n1;
    }

    public static int power(int x, int y) {
```

```
    if (y != 0)
        return (x * power(x, y - 1));
    else
        return 1;
}
```

```
public static int fibonacci(int n) {
    if (n == 0)
        return 0;
    if (n == 1 || n == 2)
        return 1;
    return fibonacci(n - 2) + fibonacci(n - 1);
}
```

```
public static void ReverseNum(int num) {
    if (num < 10)
        System.out.println(num);
    else {
        System.out.print(num % 10);
        ReverseNum(num / 10);
    }
}
```

```
public static int Add(int n) {
    if (n != 0)
        return n + Add(n - 1);
    else
```



```
        return n;
    }
```

```
public static int sum_of_digit(int n) {
    if (n == 0)
        return 0;
    return (n % 10 + sum_of_digit(n / 10));
}
```

```
public static void main(String args[]) throws java.io.IOException {
    Scanner in = new Scanner(System.in);
    char choice;
    System.out.println(
        "Menu Program :\n 1.To find GCD and LCM \n 2.To find X^y\n 3.To print n Fibonacci numbers \n 4.To find reverse of number \n 5.To find 1+2+3+4...+(n-1)+n \n 64.Calculate sum of digits of a number \n");
    System.out.println("Enter your choice : ");
    choice = (char) System.in.read();
    switch (choice) {
        case '1':

            System.out.println("Finding GCD and LCM of 2 numbers : ");
            int n1, n2;
            System.out.println("Enter 2 numbers : ");
            n1 = in.nextInt();
            n2 = in.nextInt();
            int gcd = gcd(n1, n2);
            int lcm = (n1 * n2) / gcd;
```

```

        System.out.printf("G.C.D of %d and %d is %d.\n", n1, n2, gcd);
        System.out.printf("L.C.M of %d and %d is %d.", n1, n2, lcm);
        break;
case '2':
    System.out.println("Value of X^Y is : ");
    int x, y;
    System.out.print("Enter the value of X : ");
    x = in.nextInt();
    System.out.print("Enter the value of Y : ");
    y = in.nextInt();
    int result = power(x, y);
    System.out.println("The value of " + x + "^" + y + "=" + result);
    break;
case '3':
    System.out.println("Fibonnaci numbers are : ");
    System.out.print("Enter the number of terms : ");
    int n;
    n = in.nextInt();
    System.out.println("The fibbonaci series upto " + n + " numbers is
: ");
    for (int i = 0; i < n; i++) {
        System.out.print(fibonacci(i) + " ");
    }
    break;
case '4':
    System.out.println("Reversing a number : ");
    int num;
    System.out.print("Enter a number to be reversed : ");

```

```

        num = in.nextInt();
        System.out.print("The reverse of " + num + " is ");
        ReverseNum(num);
        break;
    case '5':
        System.out.println("Sum upto N numbers : ");
        int numb;
        System.out.print("Enter n numbers upto which sum is to be found
:");
        numb = in.nextInt();
        int sum = Add(numb);
        System.out.println("The sum upto " + numb + " numbers is :" +
sum);
        break;
    case '6':
        System.out.println("Sum of digits of a Number : ");
        int number;
        System.out.print("Enter a number : ");

        number = in.nextInt();
        int sumdig = sum_of_digit(number);
        System.out.println("The sum of digits of " + number + " is " +
sumdig);
        break;
    default:
        System.out.println("Incorrect Input");
        break;
}

```

```
}  
}
```

Output:

Menu Program :

- 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 find $1+2+3+4\ldots+(n-1)+n$
- 6.Calculate sum of digits of a number

Enter your choice :

4

Reversing a number :

Enter a number to be reversed : 123

The reverse of 123 is 321

AIM 7: To implement Array of Objects

7.1) 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 math, physics, chemistry. (Use array of objects)

Theory:

When we require a single object to store in our program we do it with a variable of type Object. But when we deal with numerous objects, then it is preferred to use an array of objects. 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. Note that when we say array of objects it is not the object itself that is stored in the array but the reference of the object.

Code:

```
import java.util.Scanner;

class Students {
    int roll, phy, chem, math, total;
    String name;

    void input() {
        Scanner in = new Scanner(System.in);
        System.out.println();
        System.out.print("Enter student name:");
```

```

    name = in.nextLine();
    System.out.print("Enter Roll_no:");
    roll = in.nextInt();
    System.out.println("Enter Marks:");
    System.out.print("Physics Marks:");
    phy = in.nextInt();
    System.out.print("Chemistry Marks:");
    chem = in.nextInt();
    System.out.print("Mathematics Marks:");
    math = in.nextInt();
    total = phy + chem + math;

}

void output() {
    System.out.println("Student: " + name + " ,Student ID: " + roll + "
,marks: ");
    System.out.println("Physics:" + phy);
    System.out.println("Chemistry:" + chem);
    System.out.println("Mathematics:" + math);
    System.out.println("Total:" + total);
}
}

class App {
    public static void main(String args[]) {
        int i, j;
        Scanner in = new Scanner(System.in);

```

```

System.out.println("Enter the number of students");
int num = in.nextInt();
Students s[] = new Students[num];
for (i = 0; i < num; i++) {
    s[i] = new Students();
}
System.out.println("Enter Details: ");
for (i = 0; i < num; i++) {
    s[i].input();
}
for (i = 0; i < num; i++) {
    s[i].output();
}
Students temp;
for (i = 0; i < num-1; i++) {
    for (j = 0; j < (num-1) - 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 Descending Order:");
for (i = 0; i < num; i++) {
    System.out.println(
        "Student Name: " + s[i].name + ", Student Id: " + s[i].roll + ",
Total: " + s[i].total);
}

```

```
    }  
  }  
}
```

Output:

Enter the number of students

3

Enter Details:

Enter student name:harsh

Enter Roll_no:7

Enter Marks:

Physics Marks:99

Chemistry Marks:78

Mathematics Marks:89

Enter student name:Prateek

Enter Roll_no:8

Enter Marks:

Physics Marks:78

Chemistry Marks:87

Mathematics Marks:90

Enter student name:sarthak

Enter Roll_no:4

Enter Marks:

Physics Marks:78

Chemistry Marks:99

Mathematics Marks:87

Student: harsh ,Student ID: 7 ,marks:

Physics:99

Chemistry:78

Mathematics:89

Total:266

Student: Prateek ,Student ID: 8 ,marks:

Physics:78

Chemistry:87

Mathematics:90

Total:255

Student: sarthak ,Student ID: 4 ,marks:

Physics:78

Chemistry:99

Mathematics:87

Student Marks in Descending Order:

Student Name: harsh, Student Id: 7, Total: 266

Student Name: sarthak, Student Id: 4, Total: 264

Student Name: Prateek, Student Id: 8, Total: 255

AIM 8: To implement Constructors and overloading

8.1) WAP find area of square and rectangle using overloaded constructor

Theory:

A constructor in Java is a special method that is used to initialize objects. The constructor is called when an object of a class is created. It can be used to set initial values for object attributes. Constructors can be of 2 types:

- Default
- Parameterized

Constructor overloading in Java is a technique of having more than one constructor with different parameter lists. They are arranged in a way that each constructor performs a different task. They are differentiated by the compiler by the number of parameters in the list and their types.

Code:

```
import java.util.*;

public class Area {

    double sq, rect;

    Area(double side) {
        sq = side * side;
        System.out.println("Area of Square is " + sq + " sq units ");
    }

    Area(double len, double width) {
        rect = len * width;
```

```
        System.out.println("Area of Rectangle is " + rect + " sq units ");
    }

    public static void main(String args[]) {
        System.out.println("Enter length of square : ");
        Scanner in = new Scanner(System.in);
        double s = in.nextDouble();
        System.out.println("Enter the length and breadth of Rectangle : ");
        double len = in.nextDouble();
        double br = in.nextDouble();
        Area obj = new Area(s);
        Area obj1 = new Area(len, br);
    }
}
```

Output:

Enter length of square :

6

Enter the length and breadth of Rectangle :

2.2 6.7

Area of Square is 36.0 sq units

Area of Rectangle is 14.74 sq units

8.2) 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:

Encapsulation is one of the key features of object-oriented programming. Encapsulation refers to the bundling of fields and methods inside a single class. It prevents outer classes from accessing and changing fields and methods of a class. This also helps to achieve data hiding. We can create a fully encapsulated class in Java by making all the data members of the class private. Now we can use setter and getter methods to set and get the data in it.

Code:

```
import java.util.*;

class Rectangle {
    int length;
    int breadth;

    Rectangle() {
        length = 2;
        breadth = 2;
    }
}
```

```

    }
    Rectangle(int length1, int breadth1) {
        length = length1;
        breadth = breadth1;
    }
    int area() {
        return length * breadth;
    }
}

class Cube {
    int side;

    Cube() {
        side = 2;
    }
    Cube(int side1) {
        side = side1;
    }
    int volume() {
        return side * side * side;
    }
}

class Size {
    int calcSize(Object obj) {
        if (obj instanceof Cube) {
            return ((Cube) obj).volume();
        }
    }
}

```

```

    if (obj instanceof Rectangle) {

        return ((Rectangle) obj).area();
    } else {
        return -1;
    }
}

}

public class Main {
    public static void main(String[] args) {
        System.out.println("Enter length of cube : ");
        Scanner in = new Scanner(System.in);
        int s = in.nextInt();
        System.out.println("Enter the length and breadth of Rectangle : ");
        int l = in.nextInt();
        int b = in.nextInt();
        Cube cu = new Cube(s);
        Rectangle re = new Rectangle(l, b);
        Size size = new Size();
        int isCube = size.calcSize(cu);
        System.out.println("Volume of cube is : " + isCube);
        int isRect = size.calcSize(re);
        System.out.println("Area of Rectangle is : " + isRect);
    }
}

```

Output:

Enter length of cube :

73

Enter the length and breadth of Rectangle :

13 69

Volume of cube is : 389017

Area of Rectangle is : 897

AIM 9: To implement Abstract classes

9.1) Write a abstract class program to calculate area of circle, rectangle and triangle

Theory:

In Java, Data Abstraction is defined as the process of reducing the object to its essence so that only the necessary characteristics are exposed to the users. Abstraction defines an object in terms of its properties (attributes), behavior (methods), and interfaces (means of communicating with other objects). Consider a real-life example of a man driving a car. The man only knows that pressing the accelerators will increase the speed of a car or applying brakes will stop the car, but he does not know about how on pressing the accelerator the speed is actually increasing, he does not know about the inner mechanism of the car or the implementation of the accelerator, brakes, etc in the car.

Code:

```
import java.util.*;

abstract class AreaOfShape {
    double len1;
    double len2;

    AreaOfShape(double l1, double l2) {
        len1 = l1;
        len2 = l2;
    }

    abstract double area();
}
```



```
}
```

```
class Circle extends AreaOfShape {
```

```
    Circle(double r) {
```

```
        super(r, 0);
```

```
    }
```

```
    double area() {
```

```
        return (3.14 * len1 * len1);
```

```
    }
```

```
}
```

```
class Rectangle extends AreaOfShape {
```

```
    Rectangle(double l1, double l2) {
```

```
        super(l1, l2);
```

```
    }
```

```
    double area() {
```

```
        return (len1 * len2);
```

```
    }
```

```
}
```

```
class Triangle extends AreaOfShape {
```

```
    Triangle(double l1, double l2) {
```

```
        super(l1, l2);
```

```
    }
```

```
double area() {  
    return (0.5 * len1 * len2);  
}  
}
```

```
public class Main {  
    public static void main(String args[]) {  
        Scanner in = new Scanner(System.in);  
        System.out.println("Enter radius of Circle : ");  
        double r = in.nextDouble();  
        Circle c = new Circle(r);  
        System.out.printf("Area of Circle is : %.4f sq.units", c.area());  
        System.out.println("\nEnter length of Rectangle : ");  
        double l = in.nextDouble();  
        System.out.println("Enter breadth of Rectangle : ");  
        double b = in.nextDouble();  
        Rectangle rectarea = new Rectangle(l, b);  
        System.out.printf("Area of Rectangle is : %.4f sq.units",  
rectarea.area());  
        System.out.println("\nEnter height of Triangle: ");  
        double h = in.nextDouble();  
  
        System.out.println("Enter base length of Triangle : ");  
        double base = in.nextDouble();  
        Triangle triarea = new Triangle(h, base);  
        System.out.printf("Area of Triangle is : %.4f sq.units", triarea.area());  
    }  
}
```

}

Output:

Enter radius of Circle :

17

Area of Circle is : 907.4600 sq.units

Enter length of Rectangle :

69

Enter breadth of Rectangle :

42

Area of Rectangle is : 2898.0000 sq.units

Enter height of Triangle:

18

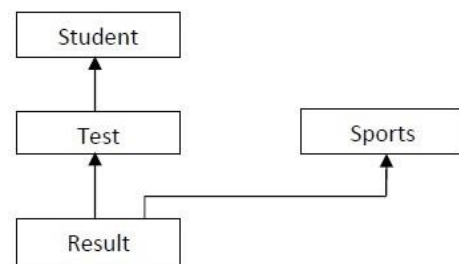
Enter base length of Triangle :

90

Area of Triangle is : 810.0000 sq.units

AIM 10: To implement Inheritance, interfaces and method overriding

10.1) 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:

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).

The idea behind inheritance in Java is that you can create new classes that are built upon existing classes. When you inherit from an existing class, you can reuse methods and fields of the parent class. Moreover, you can add new methods and fields in your current class also.

Inheritance represents the IS-A relationship which is also known as a parent-child relationship.

Code:

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

```
class Student {
```

```
    int rollno;
```

```
    int display() {
```

```
        Scanner in = new Scanner(System.in);
```

```
        System.out.print("Enter Rollno : ");
```

```
        rollno = in.nextInt();
```

```
        return rollno;
```

```
    }
```

```
}
```

```
class Test extends Student {
```

```
    int sem1_marks;
```

```
    int sem2_marks;
```

```
    int displaySem1() {
```

```
        Scanner in = new Scanner(System.in);
```

```
        System.out.print("Enter Sem1 Marks : ");
```

```
        int sem1_marks = in.nextInt();
```

```
        return sem1_marks;
```

```
    }
```

```
    int displaySem2() {
```

```
        Scanner in = new Scanner(System.in);
        System.out.print("Enter Sem2 Marks : ");
        int sem2_marks = in.nextInt();
        return sem2_marks;
    }
}
```

```
class Result extends Test implements Sports {
    public int rollNo() {
        return (display());
    }

    public int total() {
        return (displaySem1() + displaySem2() + score());
    }

    public int score() {
        Scanner in = new Scanner(System.in);
        System.out.print("Enter score : ");
        int score = in.nextInt();
        return score;
    }
}

interface Sports {
    int score();
}
```

```
public class Main {  
    public static void main(String[] args) {  
        Result res = new Result();  
        int roll = res.rollNo();  
        int totalScore = res.total();  
        System.out.println("Roll Number : " + roll);  
        System.out.println("Total score : " + totalScore);  
    }  
}
```

Output:

Enter Rollno : 17

Enter Sem1 Marks : 88

Enter Sem2 Marks : 79

Enter score : 92

Roll Number : 17

Total score : 259

AIM 11: To implement Package

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

Theory:

Packages in Java is a mechanism to encapsulate a group of classes, sub packages and interfaces. Packages are used for:

- Preventing naming conflicts. For example there can be two classes with name Employee in two packages, college.staff.cse.Employee and college.staff.ee.Employee
- Making searching/locating and usage of classes, interfaces, enumerations and annotations easier
- Providing controlled access: protected and default have package level access control. A protected member is accessible by classes in the same package and its subclasses. A default member (without any access specifier) is accessible by classes in the same package only.
- Packages can be considered as data encapsulation (or data-hiding).

Code:

```
// Created custom package calculate with function fibonacci series:
```

```
package calculate;  
  
public class Fibonacci {  
    public void sum(int n)  
    {  
        int first=0,next=1;
```



```

System.out.print(first + "\t" + next);
for(int i=1;i<=n-2;i++)
{
int sum = first + next;
first = next;
next = sum;
System.out.print("\t" + sum);
}
}
}
// Function to import Fibonacci series:
import java.util.*;
import calculate.*;
public class Mypackage {
public static void main(String args[])
{
Scanner in = new Scanner(System.in);
System.out.print("Enter the number of terms of Fibonnaci series :");
}
int num = in.nextInt();
Fibonacci fb = new Fibonacci();
fb.sum(num);
}

```

Output:

Enter the number of terms of Fibonnaci series : 6

0 1 1 2 3 5

AIM 12: To implement exceptions in Java

12.1) 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.

Code:

```
public class Main {  
    public static void main(String args[]) {  
        int valid, invalid, i, num;  
        valid = invalid = i = 0;  
        for (i = 0; i < args.length; i++) {  
            try {  
                num = Integer.parseInt(args[i]);  
            }  
            catch (NumberFormatException e) {  
                invalid++;  
                System.out.println("Invalid Argument : " + args[i]);  
                continue;  
            }  
        }  
    }  
}
```

```
        valid++;  
    }  
    System.out.println("Total valid Arguments : " + valid);  
    System.out.println("Total invalid Arguments : " + invalid);  
}  
}
```

Output:

C:\JavaExpts> java Hello 12 45 world

Invalid Argument : world

Total valid Arguments : 2 Total invalid Arguments : 1

12.2) 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 in = 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 = in.nextInt();  
                String str1 = in.nextLine();  
                System.out.println("Enter Name of Student : ");  
                name = in.nextLine();  
                System.out.println("Enter the Center Number : ");  
                centerNum = in.nextInt();  
                String str = in.nextLine();  
                System.out.println("Enter Date : ");  
                date = in.nextLine();  
                System.out.println("Enter the Marks in Java : ");  
                m = in.nextInt();  
                System.out.println("Enter the Marks in DS : ");  
                m2 = in.nextInt();  
                System.out.println("Enter the Marks in DIS : ");  
                m3 = in.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 = in.nextInt();

}

}

public static void Marks(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 in Java : " + marks + "\nMarks in DS : "

```

```
        + marks2 + "\nMarks in DIS : " + marks3);  
    }  
}  
}
```

Output:

Enter the Seat Number :

12345

Enter Name of Student :

Harsh

Enter the Center Number :

8902

Enter Date :

13/01/2022

Enter the Marks in Java :

170

Enter the Marks in DS :

93

Enter the Marks in DIS :

91

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

2

AIM 13: To implement Multithreading

13.1) Write a 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:

Multithreading is a Java feature that allows concurrent execution of two or more parts of a program for maximum utilization of CPU. Each part of such program is called a thread. So, threads are light-weight processes within a process. However, we use multithreading than multiprocessing because threads use a shared memory area. They don't allocate separate memory area so saves memory, and context-switching between the threads takes less time than process.

Java Multithreading is mostly used in games, animation, etc.

Code:

```
import java.util.*;

class Five implements Runnable {
    public void run() {
        long start = System.currentTimeMillis();
        for (int i = 1; i <= 10; i++)

        {
            System.out.println("5 * " + i + " = " + (5 * i));
        }
    }
}
```



```
        long end = System.currentTimeMillis();
        System.out.println("Time taken by Five table " + (end - start));
    }
}
```

```
class Seven implements Runnable {
    public void run() {
        long start = System.currentTimeMillis();
        for (int i = 1; i <= 10; i++) {
            System.out.println("7 * " + i + " = " + (7 * i));
        }
        long end = System.currentTimeMillis();
        System.out.println("Time taken by Seven table " + (end - start));
    }
}
```

```
class Thirteen implements Runnable {
    public void run() {
        long start = System.currentTimeMillis();
        for (int i = 1; i <= 10; i++) {

            System.out.println("13 * " + i + " = " + (13 * i));
        }
        long end = System.currentTimeMillis();
        System.out.println("Time taken by Thirteen table " + (end - start));
    }
}
```

```
public class Main {  
    public static void main(String args[]) {  
        Five five = new Five();  
        Seven seven = new Seven();  
        Thirteen thirteen = new Thirteen();  
        Thread tA = new Thread(five);  
        Thread tB = new Thread(seven);  
        Thread tC = new Thread(thirteen);  
        tA.start();  
        tB.start();  
        tC.start();  
    }  
}
```

Output:

5 * 1 = 5

13 * 1 = 13

13 * 2 = 26

13 * 3 = 39

13 * 4 = 52

13 * 5 = 65

13 * 6 = 78

13 * 7 = 91

13 * 8 = 104

13 * 9 = 117

$$13 * 10 = 130$$

Time taken by Thirteen table 1

$$7 * 1 = 7$$

$$5 * 2 = 10$$

$$7 * 2 = 14$$

$$5 * 3 = 15$$

$$7 * 3 = 21$$

$$5 * 4 = 20$$

$$7 * 4 = 28$$

$$5 * 5 = 25$$

$$7 * 5 = 35$$

$$5 * 6 = 30$$

$$7 * 6 = 42$$

$$5 * 7 = 35$$

$$7 * 7 = 49$$

$$5 * 8 = 40$$

$$7 * 8 = 56$$

$$5 * 9 = 45$$

$$7 * 9 = 63$$

$$5 * 10 = 50$$

$$7 * 10 = 70$$

Time taken by Five table 3

Time taken by Seven table 3

13.1) Write java program to implement the concept of Thread Synchronization

Theory:

Multi-threaded programs may often come to a situation where multiple threads try to access the same resources and finally produce erroneous and unforeseen results. So it needs to be made sure by some synchronization method that only one thread can access the resource at a given point in time. Java provides a way of creating threads and synchronizing their tasks using synchronized blocks. Synchronized blocks in Java are marked with the synchronized keyword. A synchronized block in Java is synchronized on some object. All synchronized blocks synchronize on the same object can only have one thread executing inside them at a time.

Code:

```
import java.util.*;

class Five implements Runnable {
    public void run() {
        long start = System.currentTimeMillis();
        for (int i = 1; i <= 10; i++) {
            System.out.println("5 * " + i + " = " + (5 * i));
            try {
                Thread.sleep(1000);
            } catch (Exception e) {
            }
        }
        long end = System.currentTimeMillis();
        System.out.println("Time taken by Five table " + (end - start));
    }
}
```

```
}  
}
```

```
class Seven implements Runnable {  
    public void run() {  
        long start = System.currentTimeMillis();  
        for (int i = 1; i <= 10; i++) {  
            System.out.println("7 * " + i + " = " + (7 * i));  
            try {  
                Thread.sleep(1000);  
            } catch (Exception e) {  
            }  
        }  
        long end = System.currentTimeMillis();  
        System.out.println("Time taken by Seven table " + (end - start));  
    }  
}
```

```
class Thirteen implements Runnable {  
    public void run() {  
        long start = System.currentTimeMillis();  
        for (int i = 1; i <= 10; i++) {  
  
            System.out.println("13 * " + i + " = " + (13 * i));  
            try {  
                Thread.sleep(1000);  

```

```

        } catch (Exception e) {
        }
    }
    long end = System.currentTimeMillis();
    System.out.println("Time taken by Thirteen table " + (end - start));
}
}

```

```

public class Main {
    public static void main(String args[]) {
        Five five = new Five();
        Seven seven = new Seven();
        Thirteen thirteen = new Thirteen();
        Thread tA = new Thread(five);
        Thread tB = new Thread(seven);
        Thread tC = new Thread(thirteen);
        tA.start();
        tB.start();
        tC.start();
    }
}

```

Output:

7 * 1 = 7

13 * 1 = 13

5 * 1 = 5

$$7 * 2 = 14$$

$$13 * 2 = 26$$

$$5 * 2 = 10$$

$$7 * 3 = 21$$

$$13 * 3 = 39$$

$$5 * 3 = 15$$

$$7 * 4 = 28$$

$$13 * 4 = 52$$

$$5 * 4 = 20$$

$$7 * 5 = 35$$

$$13 * 5 = 65$$

$$5 * 5 = 25$$

$$7 * 6 = 42$$

$$13 * 6 = 78$$

$$5 * 6 = 30$$

$$7 * 7 = 49$$

$$13 * 7 = 91$$

$$5 * 7 = 35$$

$$7 * 8 = 56$$

$$13 * 8 = 104$$

$$5 * 8 = 40$$

$$7 * 9 = 63$$

$$13 * 9 = 117$$

$$5 * 9 = 45$$

$$7 * 10 = 70$$

$$13 * 10 = 130$$

$$5 * 10 = 50$$

Time taken by Seven table 10071

Time taken by Thirteen table 10086

Time taken by Five table 10117

AIM 14: To implement Swings

14.1) Write java program to draw the house using swings

Theory:

java.awt.Graphics class provides many methods for graphics programming.

Commonly used methods of Graphics class:

- **public void drawRect(int x, int y, int width, int height):** draws a rectangle with the specified width and height.
- **public abstract void fillRect(int x, int y, int width, int height):** is used to fill rectangle with the default color and specified width and height.
- **public abstract void drawOval(int x, int y, int width, int height):** is used to draw oval with the specified width and height.
- **public abstract void fillOval(int x, int y, int width, int height):** is used to fill oval with the default color and specified width and height.

Code:

```
import javax.swing.JFrame;
```

```
import javax.swing.JPanel;
```

```
import java.awt.Graphics;
```

```
import java.awt.Color;
```

```
public class Main {
```

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

```
        JFrame frame = new JFrame();
```

```
        frame.setTitle("House");
```

```
frame.setBounds(10, 10, 700, 500);
frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
JPanel pn = new JPanel() {
    public void paint(Graphics g) {
        int a[] = { 150, 300, 225 };
        int b[] = { 150, 150, 25 };
        g.setColor(Color.red);
        g.fillRect(150, 150, 180, 220);
        g.setColor(Color.green);
        g.fillRect(185, 220, 80, 150);
        g.setColor(Color.orange);
        g.fillPolygon(a, b, 3);
        g.setColor(Color.red);
        g.fillOval(200, 75, 50, 50);
        g.setColor(Color.yellow);
        g.fillOval(240, 280, 15, 15);
        g.setColor(Color.yellow);
        g.fillRect(300, 150, 270, 220);
        g.setColor(Color.blue);

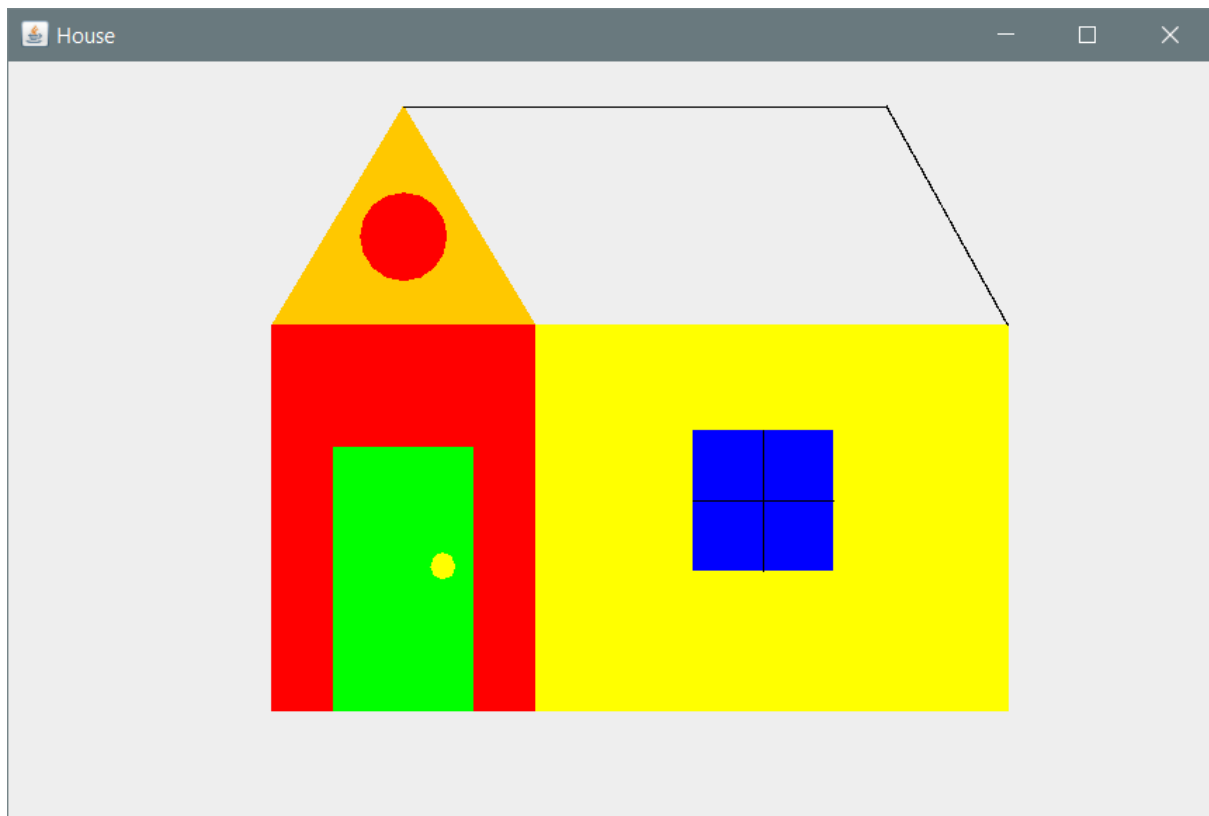
        g.fillRect(390, 210, 80, 80);
        g.setColor(Color.black);
        g.drawLine(430, 210, 430, 290);
        g.setColor(Color.black);
        g.drawLine(470, 250, 390, 250);
        g.setColor(Color.black);
        g.drawLine(500, 26, 225, 26);
```

```
        g.setColor(Color.black);
        g.drawLine(500, 26, 569, 150);

    }
};

frame.add(pn);
frame.setVisible(true);
}
}
```

Output:



14.2) Write java program to create an advertisement banner on an applet using multithreading

Theory:

Java BufferedImage class is a subclass of Image class. It is used to handle and manipulate the image data. A BufferedImage is made of ColorModel of image data. All BufferedImage objects have an upper left corner coordinate of (0, 0).

Code:

```
import javax.swing.*;

import java.awt.*;
import java.io.FileInputStream;
import java.io.IOException;
import java.awt.image.BufferedImage;
import javax.imageio.ImageIO;

class MyFrame extends JFrame implements Runnable {
    Container c;
    JLabel title, ad, label, image, label1;

    public MyFrame() throws IOException {
        setTitle("Advertisement");
        setBounds(300, 90, 900, 600);
        setDefaultCloseOperation(EXIT_ON_CLOSE);
        setVisible(true);
        c = getContentPane();
```

```
c.setBackground(Color.decode("#f1959b"));
c.setLayout(null);
BufferedImage myPicture;
myPicture = ImageIO.read(new FileInputStream("download.jpeg"));
image = new JLabel(new ImageIcon(myPicture));
image.setSize(300, 300);
image.setLocation(280, 60);
c.add(image);
title = new JLabel("Buy a brand new laptop starting @49999INR");
title.setForeground(Color.white);
title.setFont(new Font("Osaka", Font.BOLD, 20));
title.setSize(450, 30);
title.setLocation(250, 30);
c.add(title);

ad = new JLabel("");
ad.setForeground(Color.white);
ad.setFont(new Font("Osaka", Font.BOLD, 25));
ad.setSize(300, 30);
ad.setLocation(350, 360);
c.add(ad);
label = new JLabel("i3, i5, i7 core available");
label.setForeground(Color.white);
label.setFont(new Font("Osaka", Font.BOLD, 20));
label.setSize(1000, 30);
label.setLocation(290, 410);
c.add(label);
```

```

        label1 = new JLabel("Get amazing offers upto 40% discount");
        label1.setForeground(Color.white);
        label1.setFont(new Font("Osaka", Font.BOLD, 20));
        label1.setSize(1000, 30);
        label1.setLocation(270, 460);
        c.add(label1);
        new Thread(this).start();
    }

    public void run() {
        try {
            while (true) {
                if (title.getText() == "Feel the Bass") {
                    title.setText("Feel the Rhythm");
                    Thread.sleep(1000);
                }

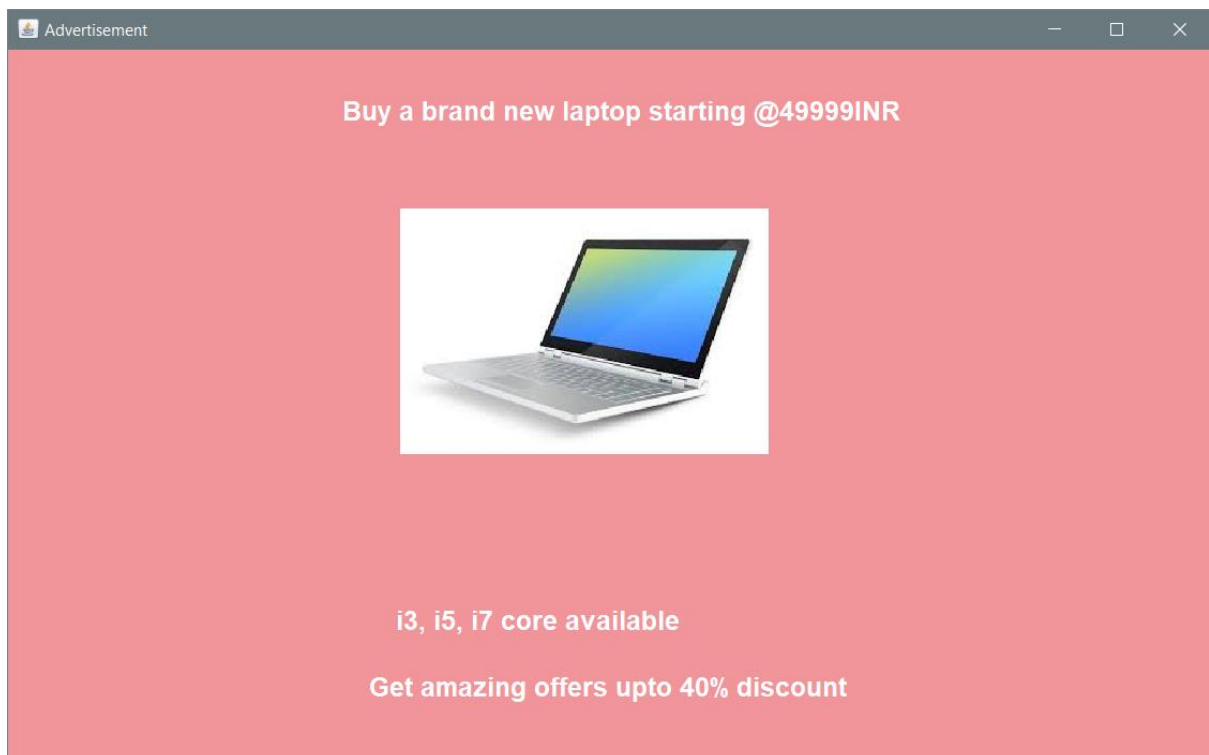
                else if (ad.getText() == "Play your mood") {
                    ad.setText("Live your mood");
                    Thread.sleep(1000);
                }
            }

        } catch (InterruptedException e) {
        }
    }
}

```

```
class Main {  
  
    public static void main(String[] args) throws IOException {  
        MyFrame frame = new MyFrame();  
    }  
}
```

Output:



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

15.1) Write a java program to create a registration form using AWT.

Theory:

AWT stands for Abstract window toolkit is an Application programming interface (API) for creating Graphical User Interface (GUI) in Java. It allows Java programmers to develop window-based applications.

AWT provides various components like button, label, checkbox, etc. used as objects inside a Java Program. AWT components use the resources of the operating system, i.e., they are platform-dependent, which means, component's view can be changed according to the view of the operating system. The classes for AWT are provided by the Java.awt package for various AWT components.

Code:

```
import javax.swing.*;
import javax.swing.JFrame;
import javax.swing.border.EmptyBorder;
import java.awt.*;
import java.awt.event.*;

class MyFrame extends JFrame implements ActionListener {
    JLabel label1, label2, label3, label4, label5;
    JTextField t1, t2;
    JRadioButton male, female;
    JComboBox depts;
```



```
JTextArea adr, screen;
```

```
JCheckBox terms;
```

```
JButton submit;
```

```
JLabel msg;
```

```
MyFrame() {  
    setTitle("Registration Form");  
    setSize(700, 500);  
    setDefaultCloseOperation(EXIT_ON_CLOSE);  
    Container c = getContentPane();  
    c.setLayout(null);  
    // name label  
    label1 = new JLabel("Name");  
    label1.setBounds(40, 30, 100, 20);  
    c.add(label1);  
    // name inputbox  
    t1 = new JTextField();  
    t1.setBounds(120, 30, 150, 20);  
    c.add(t1);  
    // sapid label  
    label2 = new JLabel("SAP Id");  
    label2.setBounds(40, 80, 100, 20);  
    c.add(label2);  
    // sapid inputbox  
    t2 = new JTextField();  
    t2.setBounds(120, 80, 150, 20);
```

```

c.add(t2);

// gender label
label3 = new JLabel("Gender");
label3.setBounds(40, 130, 100, 20);
c.add(label3);

// radiobutton
male = new JRadioButton("Male");
male.setBounds(120, 130, 90, 20);
female = new JRadioButton("Female");
female.setBounds(210, 130, 100, 20);
c.add(male);
c.add(female);

// to select only one radio button
ButtonGroup btn = new ButtonGroup();
btn.add(male);
btn.add(female);

// to select department
label4 = new JLabel("Department");
label4.setBounds(40, 180, 100, 20);
c.add(label4);

// list of departments
String dept[] = { "Computer", "I.T", "EXTC", "ELEX", "Mechanical",
"Chemical", "A I-ML", "DS",
    "IOT & Cyber Security" };
depts = new JComboBox(dept);
depts.setBounds(120, 180, 150, 20);
c.add(depts);

```

```
// to input address
```

```
label5 = new JLabel("Address");
```

```
label5.setBounds(40, 230, 100, 20);
```

```
c.add(label5);
```

```
// text area
```

```
adr = new JTextArea();
```

```
adr.setBounds(120, 230, 150, 70);
```

```
c.add(adr);
```

```
// checkbox
```

```
terms = new JCheckBox("Save and Proceed");
```

```
terms.setBounds(40, 330, 170, 25);
```

```
c.add(terms);
```

```
// submit button
```

```
submit = new JButton("Submit");
```

```
submit.setBounds(40, 380, 170, 25);
```

```
c.add(submit);
```

```
submit.addActionListener(this);
```

```
// display of form details
```

```
screen = new JTextArea();
```

```
screen.setBounds(350, 30, 300, 375);
```

```
screen.setBorder(new EmptyBorder(100, 60, 10, 10));
```

```
c.add(screen);
```

```
// message
```

```
msg = new JLabel("");
```

```
msg.setBounds(40, 430, 170, 25);
```

```
c.add(msg);
```

```

        setVisible(true);
    }

    public void actionPerformed(ActionEvent e) {
        if (terms.isSelected()) {
            msg.setText("Registartion Successful !!");
            String name = t1.getText();
            String sapid = t2.getText();
            String gender = "Male";
            if (female.isSelected()) {
                gender = "Female";
            }
            String dept = (String) depts.getSelectedItem();
            String address = adr.getText();
            screen.setText("Name - " + name + "\n" + "SAP Id -" + sapid +
                "\n" + "Gender - " + gender + "\n"
                + "Department - " + dept + "\n" + "Address - " + address +
                "\n");
        } else {
            msg.setText("Kindly Save and Procced !!");
            screen.setText(" ");
        }
    }
}

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

```

```
MyFrame frame = new MyFrame();  
}  
}
```

Output:

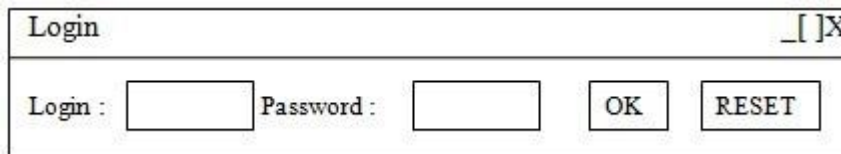
The screenshot shows a Java Swing window titled "Registration Form". The window contains a registration form with the following fields and controls:

- Name:** A text field containing "Pratham".
- SAP Id:** A text field containing "60004200082".
- Gender:** Two radio buttons, "Male" (selected) and "Female".
- Department:** A dropdown menu with "Computer" selected.
- Address:** A text area containing "Sherbet Towers, MD Road, Thane".
- Save and Proceed:** A checkbox that is checked.
- Submit:** A button labeled "Submit".

On the right side of the window, there is a preview of the entered data:

Name - Pratham
SAP Id -60004200082
Gender - Male
Department - Computer
Address - Sherbet Towers,
MD Road,
Thane

15.2) 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:

Java swing provides use with a few components which can be modified as per the users requirements. Some of the components are:

- A JButton is also known as a push button. The user presses or clicks a JButton to perform an action. JButton can display text and an icon.
- A JCheckBox has two states: selected and unselected. A group of JCheckBoxes is used when we need the user to make multiple choices. We can use a combination of an Action object, a string label, an icon, and a boolean flag to indicate if it is selected by default to create JCheckbox
- A JPasswordField is a JTextField and it hides the characters typed in. We can set our own echo character by using its setEchoChar(char newEchoChar) method. The JPasswordField class has the same set of constructors as the JTextField class.
- A JTextField can handle one line of plain text. Its constructors accept a combination of the following three values.
 - A string - specifies the initial text. Default to null.
 - The number of columns - specifies the width. Default to 0.
 - A Document object - specifies the model.

Code:

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

class MyFrame extends JFrame implements ActionListener {
    JLabel label1, label2;
    JTextField username;
    JPasswordField password;
    JButton submit, reset, exit;
    String strusername, strpassword;
    Container c = getContentPane();
    MyFrame() {
        setTitle("Login");
        setSize(700, 500);
        setDefaultCloseOperation(EXIT_ON_CLOSE);
        c = getContentPane();
        c.setLayout(null);
        label1 = new JLabel("User Name: ");
        label1.setBounds(40, 30, 250, 25);
        label1.setFont(new Font("Osaka", Font.BOLD, 15));
        c.add(label1);
        username = new JTextField();
        username.setBounds(150, 30, 150, 25);
        c.add(username);
        label2 = new JLabel("Password: ");
        label2.setBounds(40, 80, 250, 25);
```

```

label2.setFont(new Font("Osaka", Font.BOLD, 15));
c.add(label2);
password = new JPasswordField(10);
password.setBounds(150, 80, 150, 25);
c.add(password);
password.setEchoChar('*');
submit = new JButton("Submit");
submit.setBounds(40, 150, 100, 25);
submit.setFont(new Font("Osaka", Font.BOLD, 15));
reset = new JButton("Reset");
reset.setBounds(190, 150, 100, 25);
reset.setFont(new Font("Osaka", Font.BOLD, 15));
exit = new JButton("Exit");
c.add(submit);
c.add(reset);
c.add(exit);
submit.addActionListener(this);
reset.addActionListener(this);
exit.addActionListener(this);
setVisible(true);
}

public void actionPerformed(ActionEvent e) {
    if (e.getSource() == submit) {
        strusername = username.getText();
        strpassword = password.getText();
        if (strusername.equals("Harsh123") &&
            strpassword.equals("Harsh@0989")) {
            JOptionPane.showMessageDialog(c, "Successful Login");

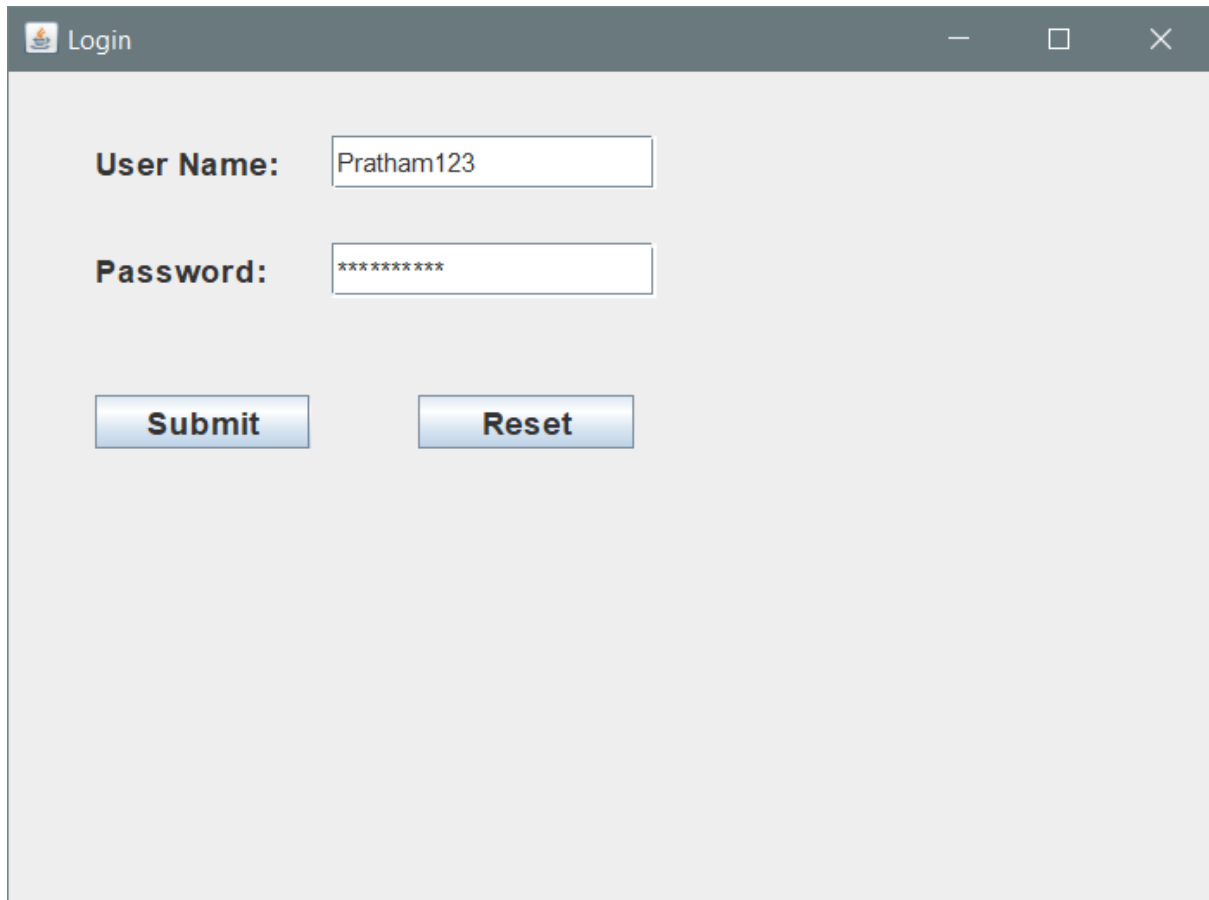
```



```
        System.exit(0);
    } else {
        JOptionPane.showMessageDialog(c, "Unsuccessful Login");
        username.setText("");
        password.setText("");
        username.requestFocus();
    }
} else if (e.getSource() == reset) {
    username.setText("");
    password.setText("");
    username.requestFocus();
}
else {
    System.exit(0);
}
}

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

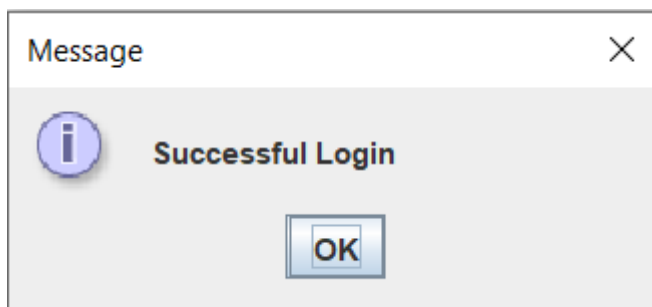
Output:



A screenshot of a Java Swing window titled "Login". The window has a dark gray title bar with standard window controls (minimize, maximize, close). The main content area is light gray and contains two text input fields. The first field is labeled "User Name:" and contains the text "Pratham123". The second field is labeled "Password:" and contains eight asterisks "*****". Below the password field are two buttons: "Submit" and "Reset", both with a blue gradient and a slight shadow.

User Name:

Password:



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

16.1) 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:

JFrame provides the user with different methods that can help the user tackle problems they are facing. Some of the methods are stated below

- `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.

Code:

```
import javax.swing.*;
import javax.swing.JFrame;
import javax.swing.border.EmptyBorder;
import java.awt.*;
import java.awt.event.*;
```

```
class MyFrame extends JFrame implements ActionListener {
    JLabel label1, label2, label3, label4;

    JTextField t1, t2, t3, t4;
    JTextArea screen;
    JCheckBox terms;
    JButton submit;
    JLabel msg;

    MyFrame()
    {
        setTitle("User Information");
        setSize(700, 500);
        setDefaultCloseOperation(EXIT_ON_CLOSE);
        Container c = getContentPane();
        c.setLayout(null);
        // name label
        label1 = new JLabel("Name");
        label1.setBounds(40, 30, 100, 25);
        c.add(label1);
        // name inputbox
        t1 = new JTextField();
        t1.setBounds(120, 30, 150, 25);
        c.add(t1);
        // Street label
        label2 = new JLabel("Street");
        label2.setBounds(40, 80, 100, 25);
```

```
c.add(label2);  
// street inputbox  
t2 = new JTextField();  
  
t2.setBounds(120, 80, 150, 25);  
c.add(t2);  
// City label  
label3 = new JLabel("City");  
label3.setBounds(40, 130, 100, 25);  
c.add(label3);  
// city inputbox  
t3 = new JTextField();  
t3.setBounds(120, 130, 150, 25);  
c.add(t3);  
// pincode label  
label4 = new JLabel("Pincode");  
label4.setBounds(40, 180, 100, 25);  
c.add(label4);  
// pincode textfield  
t4 = new JTextField();  
t4.setBounds(120, 180, 150, 25);  
c.add(t4);  
// checkbox  
terms = new JCheckBox("Save and Proceed");  
terms.setBounds(40, 250, 170, 25);  
  
c.add(terms);
```

```

// submit button
submit = new JButton("Submit");
submit.setBounds(40, 300, 170, 25);

c.add(submit);
submit.addActionListener(this);
// display of form details
screen = new JTextArea();
screen.setBounds(350, 30, 300, 375);
screen.setBorder(new EmptyBorder(100, 30, 10, 10));
c.add(screen);
// message
msg = new JLabel("");
msg.setBounds(40, 350, 200, 25);
c.add(msg);
setVisible(true);
}

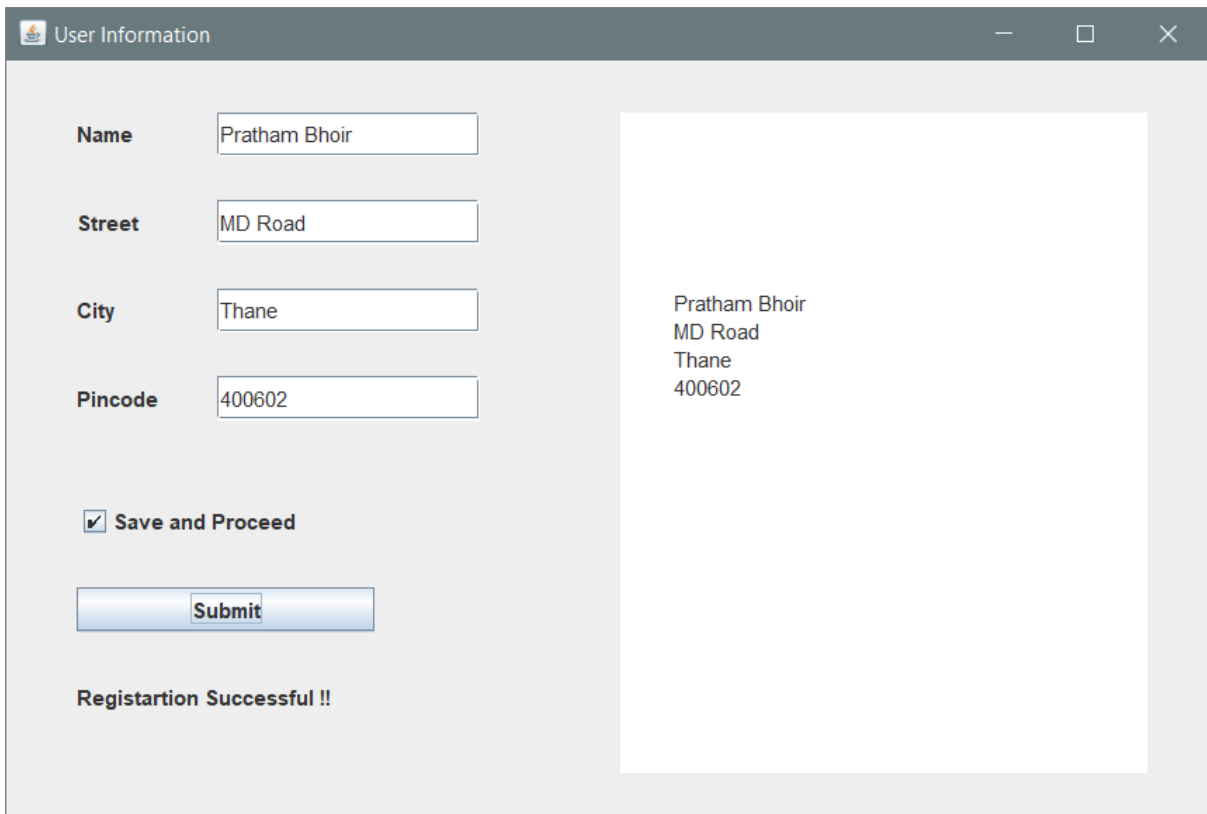
public void actionPerformed(ActionEvent e) {
    if (terms.isSelected()) {
        msg.setText("Registration Successful !!");
        String name = t1.getText();
        String street = t2.getText();
        String city = t3.getText();
        String pincode = t4.getText();
        screen.setText(name + "\n" + street + "\n" + city + "\n" + pincode
+ "\n");
    } else {

```

```
        msg.setText("Kindly Save and Procced !!");  
        screen.setText(" ");  
    }  
}  
}
```

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

Output:



The screenshot shows a Java Swing window titled "User Information". The window contains a registration form with the following elements:

- Name:** Text field containing "Pratham Bhoir".
- Street:** Text field containing "MD Road".
- City:** Text field containing "Thane".
- Pincode:** Text field containing "400602".
- Save and Proceed:** A checkbox that is checked.
- Submit:** A button labeled "Submit".
- Registartion Successful !!:** A message displayed below the form.
- Text Area:** A large text area on the right side of the window displaying the entered information:
Pratham Bhoir
MD Road
Thane
400602

16.2) 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:

The Java ActionListener is notified whenever you click on the button or menu item. It is notified against ActionEvent. The ActionListener interface is found in java.awt.event package. It has only one method: actionPerformed(). The actionPerformed() method is invoked automatically whenever you click on the registered component.

Code:

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

class MyFrame extends JFrame implements ActionListener {
    JCheckBox terms;
    JButton btn1, btn2, btn3, btn4;
    JLabel msg;

    MyFrame() {
        setTitle("Buttons");
        setSize(700, 500);
        setDefaultCloseOperation(EXIT_ON_CLOSE);
        Container c = getContentPane();
```



```
c.setSize(300, 25);
c.setLayout(new GridLayout(2, 2, 20, 20));
c.setLayout(new FlowLayout());
// button 1
btn1 = new JButton("Button 1 ");
btn1.setBounds(40, 30, 250, 25);
btn1.addActionListener(this);
c.add(btn1);
// button 2

btn2 = new JButton("Button 2 ");
btn2.setBounds(40, 80, 250, 25);
btn2.addActionListener(this);
c.add(btn2);
// button 3
btn3 = new JButton("Button 3 ");
btn3.setBounds(40, 130, 250, 25);
btn3.addActionListener(this);
c.add(btn3);
// button 4
btn4 = new JButton("Button 4 ");
btn4.setBounds(40, 180, 250, 25);
btn4.addActionListener(this);
c.add(btn4);
// message
msg = new JLabel("");
msg.setBounds(40, 350, 200, 25);
```

```

        c.add(msg);
        setVisible(true);
    }

    public void actionPerformed(ActionEvent e) {

        if (e.getSource() == btn1) {
            msg.setText("Button 1 clicked");
        } else if (e.getSource() == btn2) {
            msg.setText("Button 2 clicked");
        } else if (e.getSource() == btn3) {
            msg.setText("Button 3 clicked");
        } else if (e.getSource() == btn4) {
            msg.setText("Button 4 clicked");
        } else {
            msg.setText("No button clicked");
        }
    }
}

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

        MyFrame frame = new MyFrame();
    }
}

```

Output:

