Java Programming Lab Experiments

Name: Ayush Jain

SAP ID: 60004200132

Div.: B1

Branch: Computer Engineering

Experiment 1

Aim: To implement Java Program Structures & Simple Programs

1.1) WAP to display hello Message on screen.

Theory:

- In the Java programming language, every application must contain a main method whose signature is:

  public static void main (String [] args)

- `System.out.println("Hello, World");`

This line outputs the string “Hello, World” followed by a new line on the screen.

Code:

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

Output:

```
Hello
```

1.2) 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:

- Command line arguments is a methodology which user will give inputs through the console using commands. When a user enters the inputs on the console using commands, we sending the input as an argument to the main method in java that’s why in public static void main() we creating a string array to store values which work at executing time.

Code:

```
import java.util.Scanner;
public class Main {
 public static void main(String[] args) {
 int count = 0;
 Scanner input = new Scanner(System.in);
 long num = input.nextLong();
 while(num != 0)
 {
 num = num/10;
 ++count;
 }
 System.out.println("Number of digits: " + count);
 }
}
```

Output:

```
123456
Number of digits: 6
```

Experiment 2

Aim: To implement Java control statements and loops

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

Theory:

- The Math.pow() is used to calculate a number raise to the power of some other number. This function accepts two parameters and returns the value of first parameter raised to the second parameter.

Code:

```
import java.util.Scanner;
import java.lang.Math;
public class Main{
public static void main(String args[])
{
 double a,b,c,x1,x2,det,sqrt;
 Scanner scan=new Scanner(System.in);
 System.out.print("Enter the coefficients of the quadratic equation:");
 a=scan.nextDouble();
 b=scan.nextDouble();
 c=scan.nextDouble();
 det=(b*b)-(4*a*c);
 sqrt=Math.pow(det,0.5);
 if(det>0)
 {
 x1=(-b+sqrt)/(2*a);
 x2=(-b-sqrt)/(2*a);
 System.out.print("The roots of the entered quadratic equation are:"+x1+","+x2+" respectively");
 }
 else if(det==0)
 {
 x1=x2=(-b+sqrt)/(2*a);
 System.out.print("The roots of the entered quadratic equation are equal and that is:"+x1);


 }


 else
 {
 System.out.print("The roots of the entered quadratic equation are imaginary.");
 }
}
}
```

Output:

```
Enter the coefficients of the quadratic equation:1
 2
 1
 The roots of the entered quadratic equation are equal and that is:1.0
 Enter the coefficients of the quadratic equation:1
 1
 1
 The roots of the entered quadratic equation are imaginary.
```

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

Theory:

The Java switch statement executes one statement from multiple conditions. It is like if-else-if ladder statement. The switch statement works with byte, short, int, long, string and some wrapper types like Byte, Short, Int and Long. In other words, the switch statement tests the equality of a variable against multiple values.

Code:

```
import java.util.Scanner;
class Main {
public static void main(String args[])
{
 int n1,n2,result;
 Scanner scan=new Scanner(System.in);
 System.out.print("Enter any 2 nos.:");
 n1=scan.nextInt();
 n2=scan.nextInt();
 System.out.println("The following operations can be performed on the entered nos.");
 System.out.println("+ for Addition");
 System.out.println("- for Subtraction");
 System.out.println("* for Multiplication");
 System.out.println("/ for Division");
 System.out.println("% for Modular Division");
 System.out.print("Enter your choice:");
 char a=scan.next().charAt(0);
 switch(a)
 {
 case '+':
 result=n1+n2;
 System.out.print("Addition is "+result);
 break;
 case '-':
 result=n1-n2;
 System.out.print("Subtraction is "+result);
 break;
 case '*':
 result=n1*n2;
 System.out.print("Multiplication is "+result);
 case '/':
 result=n1/n2;
 System.out.print("Division is "+result);
 break;
 case '%':
 result=n1%n2;
 System.out.print("Modular Division is "+result);
 break;
 default:
 System.out.print("Invalid choice.");
 break;
 }
}
}
```

Output:

```
Enter any 2 nos.:2
 3
 The following operations can be performed on the entered nos.
 + for Addition
- for Subtraction
* for Multiplication
/ for Division
% for Modular Division
Enter your choice:+
Addition is 5
```

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

Theory:

In the below program, the range of numbers is taken as input and stored in the variables ‘l’ and ‘u’. Then using for-loop, the numbers between the interval of a and b are traversed. For each number in the for loop, it is checked if this number is prime or not. If found prime, print the number. Then the next number in the loop is checked, till all numbers are checked.

Code:

```
import java.util.Scanner;
 class Main {
 public static void main(String args[])
 {
 int u,l,i,j,flag;
 Scanner scan=new Scanner(System.in);
 System.out.print("Enter the lower limit:");
 l=scan.nextInt();
 System.out.print("Enter the upper limit:");
 u=scan.nextInt();
 System.out.println("The odd nos. between the range "+l+" and "+u+" are:");
 if(l%2==0)
 {
 l++;
 }
 for(i=l;i<=u;i++)
 {
 if(i%2!=0)
 {
 System.out.print(i+"\t");
 }
 }
 System.out.println();
 System.out.println("The prime nos. between the range "+l+" and "+u+" are:");
 for(i=l;i<=u;i++)
 {
 if(i==0||i==1)
 {
 continue;
 }
 flag=1;
 for(j=2;j<=i/2;++j)
 {
 if(i%j==0)
 {
 flag=0;
 break;
 }
 }
 if(flag==1)
 {
 System.out.print(i+"\t");
 }
 }
 }
 }
```

Output:

```
Enter the lower limit:1
Enter the upper limit:9
The odd nos. between the range 1 and 9 are:
 1  3  5  7  9
The prime nos. between the range 1 and 9 are:
 2  3  5  7
```

2.4) WAP to display default value of primitive data types

Theory:

Given below is a *java program* that displays default values of different primitive data types like int, float, double, boolean, String. Default value *means the* value assigned automatically by Java compiler to the unassigned variables in java.

Code:

```
public class Main {
 static boolean val1;
 static double val2;
 static float val3;
 static int val4;
 static long val5;
 static String val6;
 public static void main(String[] args) {
 System.out.println("Default values");
 System.out.println("Val1 = " + val1);
 System.out.println("Val2 = " + val2);
 System.out.println("Val3 = " + val3);
 System.out.println("Val4 = " + val4);
 System.out.println("Val5 = " + val5);
 System.out.println("Val6 = " + val6);
 }
}
```

Output:

```
Default values

Val1 = false

Val2 = 0.0

Val3 = 0.0

Val4 = 0

Val5 = 0

Val6 = null
```

2.5.1) 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:

Each pattern program has two or more than two loops. The number of the loop depends on the complexity of pattern or logic. The first for loop works for the row and the second loop works for the column.

Code:

```
class Main {
public static void main(String args[])
{
 int i,j;
 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();
 }
}
}
```

Output:

```
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
```

2.5.2) WAP to display the following patterns:

```
   A
  CB
 FED
JIHG
```

Theory:

Each pattern program has two or more than two loops. The number of the loop depends on the complexity of pattern or logic. The first for loop works for the row and the second loop works for the column.

Code:

```
class Main {
public static void main(String args[]){
 int i,j,z;
 z=0;
 for(i=1;i<=4;i++)
 {
 z=z+i;
 for(j=1;j<=4-i;j++)
 {
 System.out.print(" ");
 }
 for(j=1;j<=i;j++)
 {
 z=z-1;
 System.out.print((char)(z+65));
 }
 z=z+i;
 System.out.println();
 }
}
}
```

Output:

```
   A
  CB
 FED
JIHG
```

Experiment 3

Aim: 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 otherwe get the original number. (1260=21*60, 1395=15*93, 1530=30*51)

Theory:

A vampire number has an even number of digits and is formed by multiplying a pair of numbers containing half the number of digits of the result. The digits are taken from the original number in any order. Pairs of trailing zeroes are not allowed.

Code:

```
import java.util.Scanner;
import java.lang.Math;
class Main
{
 public static void main(String args[])
 {
 int i,j,n,x,c1,c2,k,p,c=0,f=0;
 Scanner scan=new Scanner(System.in);
 System.out.print("Enter a number:");
 n=scan.nextInt();
 for(i=n;i>0;i=i/10)
 {
 c++;
 }
 if(c%2!=0)
 {
 f=1;
 System.out.print("Not a Vampire Number.");
 }
 else
 {
 for(i=(int)Math.pow(10,c/2-1);i<(int)Math.pow(10,c/2);i++)
 {
 if(n%i==0)
 {
 j=n/i;
 if(!(j>=Math.pow(10,c/2-1)&&j<Math.pow(10,c/2)))
 {
 f=1;
 }
 x=i*(int)Math.pow(10,c/2)+j;
 if(i%10==0 && j%10==0)
 {
 f=1;
 break;
 }
 else
 {
 f=0;
 for(k=n;k>0;k=k/10)
 {
 c1=0;
 c2=0;
 for(p=n;p>0;p=p/10)
 {
 if(k%10==p%10)
 {
 c1++;
 }
 }
 for(p=x;p>0;p=p/10)
 {
 if(k%10==p%10)
 {
 c2++;
 }
 }
 if(c1!=c2)
 {
 f=1;
 break;
 }
 }
 if(f==0)
 {
 break;
 }
 }
 }
 }
 if(f==1)
 {
 System.out.println("Not a vampire number");
 }
 else
 {
 System.out.println("A vampire number");
 }
 }
 }
}
```

Output:

```
Enter a number:1260

A vampire number
```

3.2)WAP to display the following using irregular arrays

```
1

2  3

4   5   6
```

Theory:

A jagged array is an array of arrays such that member arrays can be of different sizes, i.e., we can create a 2-D array but with a variable number of columns in each row. These types of arrays are also known as Jagged arrays.

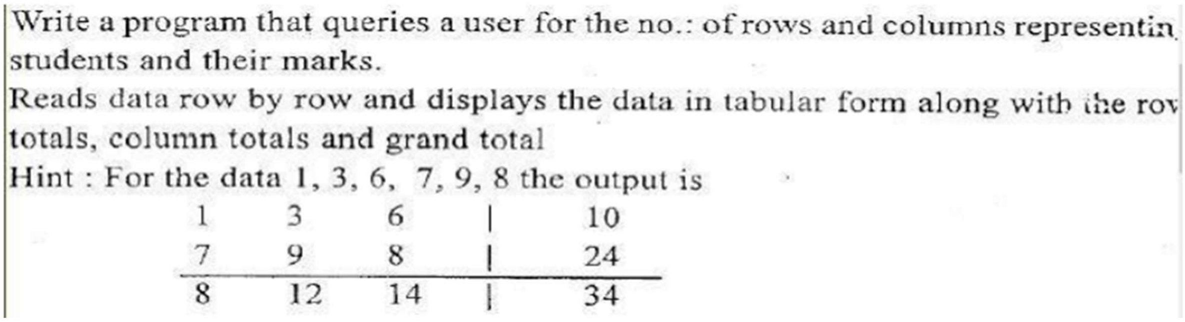
Code:

```
public class Main {
 public static void main(String[] args) {
 int arr[][]=new int[][] {{1},{2,3},{4,5,6}};
 for(int i=0;i<3;i++){
 for(int j=0;j<arr[i].length;j++){
 System.out.print(arr[i][j]+" ");
 }
 System.out.println();
 }
 }
}
```

Output:

```
1

2  3

4   5   6
```

3.3)



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

Reads data row by row and displays the data in tabular form along with the rov 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
```

Code:

```
import java.util.*;
public class Main {
 public static void main(String[] args) {
 @SuppressWarnings("resource")
 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 : 3
Enter colums of matrix : 4
1
2
3
4
5
6
7
8
9
10
11
12
1       2       3       4       | 10
5       6       7       8       | 26
9       10      11      12      | 42
----    ----    ----    ----     ----
15      18      21      24       | 78
```

Experiment 4

Aim: To implement Arrays

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. Vectors fall in legacy classes, but now it is fully compatible with collections. It is found in java.util.package and implement the list interface, so we can use all the methods of List interface

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.print("The elements in List are : "+v);
 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 : Medicines
Enter the item 2 : Stationary
Enter the item 3 : Fruits
Enter the item 4 : Clothes
            Operation
            1)add at specific location
            2)Delete item
            3)Display list
            4)Exit
Enter the choice : 1
Enter the Location(index) : 3
Enter the item : Guitar
Enter the choice : 3
The elements in List are : [Medicines, Stationary, Fruits, Guitar, Clothes]
Enter the choice : 2
Enter the item want to delete : Guitar
Enter the choice : 3
The elements in List are : [Medicines, Stationary, Fruits, Clothes]
Enter the choice : 4
```

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

Theory:

In this program, we have an array of elements to count the occurrence of its each element. One of the approaches to resolve this problem is to maintain one array to store the counts of each element of the array. Loop through the array and count the occurrence of each element as frequency and store it in another array

Code:

```
import java.util.*;
import java.util.Vector;
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();
 Object a[]=v.toArray();
 for(i=0;i<n;i++)
 {
 if((int)a[i]==x)
 {
 count++;
 }
 }
 System.out.print("Frequency:"+count);
 }
}
```

Output:

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

Experiment 5

Aim: 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.

Theory:

Given two strings, the task is to check whether these 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.

Code:

```
import java.util.Scanner;
public class Main {
 static boolean checkMetaString(String str1,String str2) {
 if(str1.equals(str2) || (str1.length()!=str2.length())) {
 return false;
 }
 else {
 int count=0;
 char check[]=new char[2];
 for(int i=0;i<str1.length();i++) {
 if(str1.charAt(i)!=str2.charAt(i)) {
 if(count>=2)
 return false;
 if(count==0) {
 count++;
 check[0]=str1.charAt(i);
 check[1]=str2.charAt(i);
 }
 else if(count==1)
 {
 if(check[0]!=str2.charAt(i) || check[1]!=str1.charAt(i)) {
 return false;
 }
 }
 }
 }
 return true;
 }
 }
 public static void main(String[] args) {
 Scanner sc = new Scanner(System.in);
 System.out.print("Enter the 1st String : ");
 String str1 = sc.next().trim().toLowerCase();
 System.out.print("Enter the 2nd String : ");
 String str2 = sc.next().trim().toLowerCase();
 boolean check=checkMetaString(str1,str2);
 if(check) {
 System.out.print("The Given Two strings are meta strings");
 }
 else {
 System.out.print("The Given Two strings are not meta strings");
 }
 }
}
```

Output:

```
Enter the 1st String : geeks

Enter the 2nd String : keegs

The Given Two strings are 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:

In this program, we've 4 conditions for each of the checks.

- The first `if` condition is to check whether the character is a **vowel** or not.
- The `else if` condition following `if` is to check whether the character is a **consonant** or not. The order should be the same otherwise, all vowels are treated as consonants as well.
- The second `else if` is to check whether the character is between **0 to 9** or not.
- Finally, the last condition is to check whether the character is a **space** character or not.

Code:

```
import java.util.Scanner;
import java.lang.*;
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:
Life is beautiful if we see it that way
No. of alphabets:31
No. of digits:0
No. of special symbols:0
No. of blank spaces:8
No.of words:9
No.of vowels:15
No.of consonants:16
```

Experiment 6

Aim: To implement Functions, recursive functions and overloading

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

Theory:

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.

Code:

```
class Main{
 void area(int s){
 System.out.println("Sqaure Area is "+(s*s));
 }
 void area(float s){
 System.out.println("Float() Sqaure Area is "+(s*s));
 }
 void area(int l,int b){
 System.out.println("Reactangle Area is "+(l*b));
 }
}
public class OverloadingArea {
 public static void main(String[] args) {
 Area a=new Area();
 a.area(8);
 a.area(4,2);
 }
}
```

Output:

```
Sqaure Area is 64
Reactangle Area is 8
```

____

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

a) To find GCD and LCM

b) To find XY

c) To print n Fibonacci numbers

d) To find reverse of number

e) To 1+2+3+4+…….+ (n-1)+n

f) Calculate sum of digits of a number

Theory:

A recursive function is a function that calls itself during its execution.

The process may repeat several times during its execution .The process may repeat several times, outputting the result and the end of each iteration.

Code:

```
import java.util.Scanner;
import java.lang.Math;
class Main
{
 public static void main(String args[])
 {
 int choice;
 System.out.println("*****MAIN MENU*****");
 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 sum of digits of a number");
 System.out.println("7.EXIT");
 Scanner scan=new Scanner(System.in);
 System.out.print("Enter your choice:");
 choice=scan.nextInt();
 Gclc gc=new Gclc();
 Po pw=new Po();
 Fib fi=new Fib();
 Rev rv=new Rev();
 S sm=new S();
 D dg=new D();
 switch(choice)
 {
 case 1:
 gc.gcd();
 break;
 case 2:
 pw.pr();
 break;
 case 3:
 fi.fib();
 break;
 case 4:
 rv.re();
 break;
 case 5:
 sm.su();
 break;
 case 6:
 dg.di();
 break;
 case 7:
 System.exit(0);
 break;
 }
 }
}
class Gclc
{
 public static void gcd()
 {
 int c,d,hcf,lcm;
 Scanner scan=new Scanner(System.in);
 System.out.print("Enter any 2 nos. whose gcd and lcm you want:");
 GL g=new GL();
 c=scan.nextInt();
 d=scan.nextInt();
 hcf=g.GCD(c,d);
 System.out.println("The GCD Of entered nos. is "+hcf);
 System.out.println("The LCM of entered nos. is "+(c*d)/hcf);
 }
}
class GL
{
 static int GCD(int x,int y)
 {
 if(y==0)
 {
 return x;
 }
 else
 {
 return GCD(y,x%y);
 }
 }
}
class Po
{
 public static void pr()
 {
 double base,p,result;
 Scanner scan=new Scanner(System.in);
 Power pr=new Power();
 System.out.print("Enter base part of the number:");
 base=scan.nextDouble();
 System.out.print("Enter the power part of the number:");
 p=scan.nextDouble();
 result=pr.power(base,p);
 System.out.print("The result is:"+base+"^"+p+"="+result);
 }
}
class Power
{
 static double power(double base,double p)
 {
 if(p!=0)
 {
 return(base*Math.pow(base,p-1));
 }
 else
 {
 return 1;
 }
 }
}
class Fib
{
 public static void fib()
 {
 int a,n,i=0;
 Scanner scan=new Scanner(System.in);
 Fibonacci f=new Fibonacci();
 System.out.print("Enter the no.of terms of Fibonacci series to show:");
 n=scan.nextInt();
 System.out.println("Fibonacci series of "+n+" terms:");
 for(a=0;a<n;a++)
 {
 System.out.print(f.fibonacci(i)+"\t");
 i++;
 }
 }
}
class Fibonacci
{
 static int fibonacci(int b)
 {
 if(b==0)
 {
 return 0;
 }
 else if(b==1)
 {
 return 1;
 }
 else
 {
 return(fibonacci(b-1)+fibonacci(b-2));
 }
 }
}
class Rev
{
 public static void re()
 {
 int n,m,reverse,count=0;
 Scanner scan=new Scanner(System.in);
 Reverse r=new Reverse();
 System.out.print("Enter a number:");
 n=scan.nextInt();
 m=n;
 while(m>0)
 {
 count++;
 m=m/10;
 }
 reverse=r.reverse(n,count);
 System.out.print("The reversed no. is:"+reverse);
 }
}
class Reverse
{
 static int reverse(int x,int length)
 {
 if(length==1)
 {
 return x;
 }
 else
 {
 int b=x%10;
 x=x/10;
 return(int)((b*Math.pow(10,length-1))+reverse(x,--length));
 }
 }
}
class S
{
 public static void su()
 {
 int n,sum;
 Scanner scan=new Scanner(System.in);
 Sum s=new Sum();
 System.out.print("Enter the value of n:");
 n=scan.nextInt();
 sum=s.sum(n);
 System.out.print("The sum of "+n+" natural numbers is:"+sum);
 }
}
class Sum
{
 static int sum(int n)
 {
 if(n==0)
 {
 return 0;
 }
 else
 {
 return(n+sum(n-1));
 }
 }
}
class D
{
 public static void di()
 {
 int n,sum;
 Scanner scan=new Scanner(System.in);
 Digit d=new Digit();
 System.out.print("Enter a number:");
 n=scan.nextInt();
 sum=d.digit(n);
 System.out.print("The sum of digits of entered no. is:"+sum);
 }
}
class Digit
{
 static int digit(int n)
 {
 if(n!=0)
 {
```

```
return(n%10 + digit(n/10));
 }
 else
 {
 return 0;
 }
 }
}
```

Output:

```
*****MAIN MENU*****
1.To find GCD and LCM
2.To find X^Y
3.To print n Fibonacci numbers
4.To find reverse of number
5.To 1+2+3+4+......+(n-1)+n
6.Calculate sum of digits of a number
7.EXIT
Enter your choice:1
Enter any 2 nos. whose gcd and lcm you want:5
15
The GCD Of entered nos. is 5
The LCM of entered nos. is 15

*****MAIN MENU*****
1.To find GCD and LCM
2.To find X^Y
3.To print n Fibonacci numbers
4.To find reverse of number
5.To 1+2+3+4+......+(n-1)+n
6.Calculate sum of digits of a number
7.EXIT
Enter your choice:2
Enter base part of the number:5
Enter the power part of the number:2
The result is:5.0^2.0=25.0

*****MAIN MENU*****
1.To find GCD and LCM
2.To find X^Y
3.To print n Fibonacci numbers
4.To find reverse of number
5.To 1+2+3+4+......+(n-1)+n
6.Calculate sum of digits of a number
7.EXIT
Enter your choice:3
Enter the no.of terms of Fibonacci series to show:5
Fibonacci series of 5 terms: 0   1       1       2       3
```

```
*****MAIN MENU*****
1.To find GCD and LCM
2.To find X^Y
3.To print n Fibonacci numbers
4.To find reverse of number
5.To 1+2+3+4+......+(n-1)+n
6.Calculate sum of digits of a number
7.EXIT Enter your choice:4
Enter a number:1234
The reversed no. is:4321


*****MAIN MENU*****
1.To find GCD and LCM
2.To find X^Y
3.To print n Fibonacci numbers
4.To find reverse of number
5.To 1+2+3+4+......+(n-1)+n
6.Calculate sum of digits of a number
7.EXIT Enter your choice:5
Enter the value of n:5
The sum of 5 natural numbers is:15
```

____

Experiment 7

Aim: To implement Array of Objects

7) 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:

An array of objects, all of whose elements are of the same class, can be declared just as an array of any built-in type. Each element of the array is an object of that class. Being able to declare arrays of objects in this way underscores the fact that a class is similar to a type.

Code:

```
import java.util.Scanner;
class Main
{
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:Adnan Ahmed
Enter Roll_no:1
Enter Marks:
Physics Marks:96
Chemistry Marks:90
Mathematics Marks:98

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


Enter student name:Laxman Shrivastav
Enter Roll_no:2
 Enter Marks:
Physics Marks:85
Chemistry Marks:96
Mathematics Marks:94

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


Enter student name:Sayli Maheshwari
Enter Roll_no:3
Enter Marks:
Physics Marks:90
Chemistry Marks:88
Mathematics Marks:95
```

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


Enter student name:John Fernandez
Enter Roll_no:4
Enter Marks:
Physics Marks:65
Chemistry Marks:77
Mathematics Marks:68

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


Enter student name:Rameshwari Iyer
Enter Roll_no:5
Enter Marks:
Physics Marks:44
Chemistry Marks:50
Mathematics Marks:38

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

Student: Adnan Ahmed ,roll_no: 1 ,marks:
Physics:96
Chemistry:90
Mathematics:98
Total:284

Student: Laxman Shrivastav ,roll_no: 2 ,marks:
Physics:85
Chemistry:96
Mathematics:94
Total:275

Student: Sayli Maheshwari ,roll_no: 3 ,marks:
Physics:90
Chemistry:88
Mathematics:95
Total:273

Student: John Fernandez ,roll_no: 4 ,marks:
Physics:65
Chemistry:77
Mathematics:68
Total:210
```

```
Student: Rameshwari Iyer ,roll_no: 5 ,marks:
Physics:44
Chemistry:50
Mathematics:38
Total:132

Student Marks in Descendin Order:
Student Name: Adnan Ahmed, Student RollNo: 1, Total: 284
Student Name: Laxman Shrivastav, Student RollNo: 2, Total: 275
Student Name: Sayli Maheshwari, Student RollNo: 3, Total: 273
Student Name: John Fernandez, Student RollNo: 4, Total: 210
Student Name: Rameshwari Iyer, Student RollNo: 5, Total: 132
```

Experiment 8

Aim: To implement Constructors and overloading

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

Theory:

Method 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.

Code:

```
import java.util.*;
class Main{
 int s1,s2;
 Shape(int s){
 s1=s;
 s2=s;
 }
 Shape(int l,int b){
 s1=l;
 s2=b;
 }
 int area(){
 return s1*s2;
 }
}
public class Expirment8_1 {
 public static void main(String args[]){
 Shape square=new Shape(4);
 Shape rect = new Shape(2,4);
 System.out.println("Area of Square of side 4 : "+square.area());
 System.out.println("Area of rectangle of side 2,4 : "+rect.area());
 }
}
```

Output:

```
Area of Square of side 4 : 16

Area of rectangle of side 2,4 : 8
```

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 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:

Constructor Overloading means a class having multiple constructors with the same name, but with different number of arguments or different type of arguments. The compiler differentiates the constructors based on the number of parameters in the list and their types. That means, on the basis of the number and type of the arguments that we pass into the constructor, the compiler determines which constructor to call.

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 Expir8_2 {
 public static void main(String[] args) {
 Scanner sc = new Scanner(System.in);
 Rect r = new Rect(2,4);
 Cube c = new Cube(2);
 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: 8
Volume of Cube: 8
Other objects: -1
```

Experiment 9

Aim: To implement Abstarct classes

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

Theory:

A class which is declared as abstract is known as an abstract class. It can have abstract and non-abstract methods. It needs to be extended and its method implemented. It cannot be instantiated.

Code:

```
abstract class CalArea{
 abstract double rectArea(double l,double b);
 abstract double circleArea(double r);
 abstract double triangleArea(double height,double base);
}
class FindArea extends CalArea{
 double rectArea(double l,double b){
 return l*b;
 }
 double circleArea(double r){
 return 3.142*r*r;
 }
 double triangleArea(double height,double base){
 return height*base/2;
 }
}
public class AbstractDemo {
 public static void main(String[] args) {
 FindArea area = new FindArea();
 System.out.println("Area of rectangle : "+area.rectArea(15.0, 4.0));
 System.out.println("Area of Circle : "+area.circleArea(4.0));
 System.out.println("Area of Triangle : "+area.triangleArea(5.0, 4.0));
 }
}
```

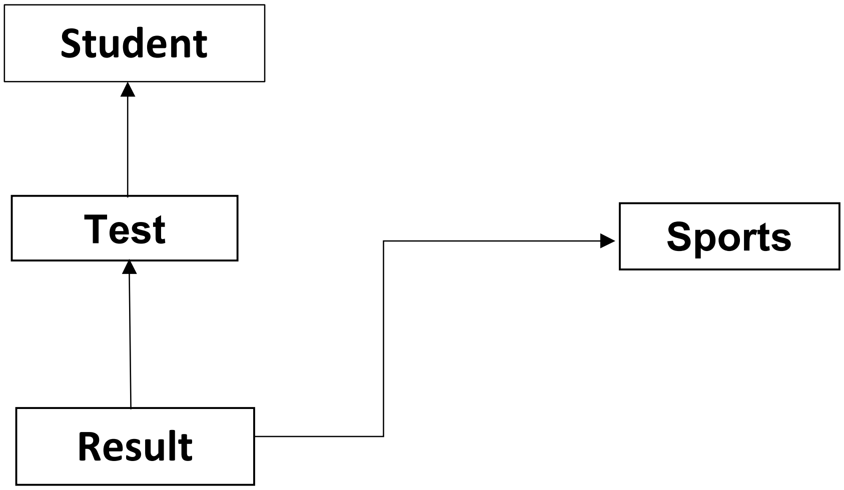
Output:

```
Area of rectangle: 60.0
Area of Circle :50.265
Area of Triangle: 10.0
```

Experiment 10

Aim: 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:

In any object-oriented programming language, Overriding is a feature that allows a subclass or child class to provide a specific implementation of a method that is already provided by one of its super-classes or parent classes

Code:

```
import java.util.Scanner;
class Student
{
 int i,a;
 Scanner scan=new Scanner(System.in);
 void roll()
 {
 System.out.print("Enter the student's roll_no.:");
 a=scan.nextInt();
 }
 void call()
 {
 System.out.println("The student with roll_no:"+a);
 }
}
class Test extends Student
{
 int sem1,sem2;
 Scanner scan=new Scanner(System.in);
 void enter()
 {
 System.out.print("Sem 1 Marks(out of 600):");
 sem1=scan.nextInt();
 System.out.print("Sem 2 Marks(out of 600):");
 sem2=scan.nextInt();
 }
}
interface Sports
{
 int score=45;
 void disp();
}
class Result extends Test implements Sports
{
 public void disp()
 {
 System.out.println("Sports Marks(out of 50):"+score);
 }
 void output()
 {
 int total=sem1+sem2+score;
 System.out.println("Total marks scored:"+total);
 }
}
class Multiple
{
 public static void main(String args[])
 {
 Result r=new Result();
 r.roll();
 r.enter();
 r.disp();
 r.call();
 r.output();
 }
}
```

Output:

```
Enter the student's roll_no.:9
Sem 1 Marks(out of 600):565
Sem 2 Marks(out of 600):526
```

```
Sports Marks(out of 50):45
The student with roll_no:9
Total marks scored:1136
```

____

Experiment 11

Aim: To implement Package

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

Theory:

Packages in Java are a mechanism to encapsulate a group of classes, interfaces, and sub-packages. In other words, we can say a package is a container of a group of related classes where some of the classes are accessible are exposed, and others are kept for internal purposes.

Code:

```
package maths;
public class Calulator {
public int fact(int x){
if(x==0 || x==1)
return 1;
else
return x*fact(x-1);
}
public int power(int x,int n){
if(n==0)
return 1;
else
return x*power(x,n-1); } }
import maths.Calulator;;
public class PackagesExp {
public static void main(String[] args) {
Calulator c = new Calulator();
System.out.println("Factorial of 5 : "+c.fact(5));
System.out.println("Power of 5 raised to 3 : "+c.power(5,3));
}
}
```

Output:

```
Factorial of 5 : 120
Power of 5 raised to 3 : 125
```

Experiment 12

Aim: 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:

An exception is an unwanted or unexpected event, which occurs during the execution of a program i.e at run time, that disrupts the normal flow of the program's instructions. To create the exception object**,** the program uses the throw keyword followed by the instantiation of the exception object

Code:

```
package maths;
public class Calulator {
class Main{
public static void main(String z[])
{
int i=0,n,a,k=0;
for(n=0;n<z.length;n++){
try{
a=Integer.parseInt(z[n]);
}
catch(NumberFormatException e){i++;}
}
System.out.println("There are "+i+" invalid and "+(z.length-i)+" valid integers in input");
}
}
```

Output:

```
There are 1 invalid and 3 valid integers in input
```

____

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:

An exception is an unwanted or unexpected event, which occurs during the execution of a program i.e at run time, that disrupts the normal flow of the program's instructions. To create the exception object**,** the program uses the throw keyword followed by the instantiation of the exception object

Code:

```
import java.util.*;
import java.io.*;
class MarksOutOfBoundsException extends Exception
{
public MarksOutOfBoundsException(String str)
{
System.out.println(str);
}
}
class stresult{
Scanner s=new Scanner(System.in);
static int
roll_no,maths,de,dm,ds,dbms,Total;
static String name;
public stresult(String n,int r){
roll_no=r;
name=n;
}
void input(){
System.out.println("Enter sem 3 marks:");
System.out.println("Maths marks:"); maths=s.nextInt();
System.out.println("Digital Electronics marks:"); de=s.nextInt();
System.out.println("Discrete Maths marks:"); dm=s.nextInt();
System.out.println("Data Structures marks:"); ds=s.nextInt();
System.out.println("DBMS marks:"); dbms=s.nextInt();
try{
```

```
if((maths<0||mat
hs>100))
throw new MarksOutOfBoundsException("Invalid Maths marks entered");
}
catch(MarksOutOfBoundsException e){
System.out.println(e+"Enter maths marks again between 0 and 100: ");
maths=s.nextInt();
}
try{
if((de<0||
de>100))
throw new MarksOutOfBoundsException("Invalid de marks entered");
}
catch(MarksOutOfBoundsException e){
System.out.println(e+"Enter de marks again between 0 and 100: ");
de=s.nextInt();
}
try{
if((dm<0||dm>100))
throw new MarksOutOfBoundsException("Invalid dm marks entered");
}
catch(MarksOutOfBoundsException e){
System.out.println(e+"Enter dm marks again between 0 and 100: ");
dm=s.nextInt();
}
try{
if((ds<0||
ds>100))
throw new MarksOutOfBoundsException("Invalid ds marks entered");
}
catch(MarksOutOfBoundsException e){
System.out.println(e+"Enter ds marks again between 0 and 100: ");
ds=s.nextInt();
}
try{
if((dbms<0||db
ms>100))
throw new MarksOutOfBoundsException("Invalid dbms marks entered");
}
catch(MarksOutOfBoundsException e){
System.out.println(e+"Enter dbms marks again between 0 and 100: ");
dbms=s.nextInt();
}
}
void total(){
Total=dbms+de+ds+d
m+maths;
Total=Total/5;
}
void display(){
System.out.pri
ntln(" ");
System.out.println("Student name:"+name);
System.out.println("Student roll no:"+roll_no);
System.out.println("Maths marks:"+maths);
System.out.println("Digital Electronics marks:"+de);
System.out.println("Data Structures marks:"+ds);
System.out.println("Discrete structures marks:"+dm);
System.out.println("DBMS marks:"+dbms);
System.out.println("Total percentage:"+Total);
if(Total>35){
System.out.println("Pass");
}
else if(Total<35){
System.out.printl
n("Fail");
}
}
}
class User{
static Scanner l=new
Scanner(System.in); public static void main(String args[]){
String n;int r;
System.out.println("Enter Student name:");
n=l.nextLine();
System.out.println("Enter roll no:");
r=l.nextInt();
s1=new stresult(n,r);
s1.input();
s1.total();
System.out.println("Result is as follows:");
s1.display();
}
}
```

Output:

```
Enter Student name:
Earth
Enter roll no:
32
Enter sem 3 marks:
Maths marks:
69
Digital Electronics marks:
78
Discrete Maths marks:
102
Data Structures marks:
-2
DBMS marks:
63
Invalid dm marks entered
MarksOutOfBoundsException
```

```
Enter dm marks again between 0 and 100: 69
Invalid ds marks entered
MarksOutOfBoundsException
Enter ds marks again between 0 and 100: 89
Result is as follows:
Student name:Earth
Student roll no:32
Maths marks:69
Digital Electronics marks:78
Data Structures marks:89
Discrete structures marks:69
DBMS marks:63
Total percentage: 73 Pass
```

Experiment 13

Aim: To implement Multithreading

13.1) 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:

Multithreading in Java is a process of executing two or more threads simultaneously to maximum utilization of CPU. Multithreaded applications execute two or more threads run concurrently.

Code:

```
import java.util.*;
class Five implements
Runnable{ public void run(){
long
start=System.currentTimeMilli
s(); int i;
for(i=1;i<11;i+
+){
System.out.pri
ntln(5*i);
}
long time=System.currentTimeMillis()-start;
System.out.println(Thread.currentThread()+" took "+time+" milliseconds to
execute");
}
}
class Seven implements
Runnable{ public
void run(){
int i;long start=System.nanoTime();
for(i=1;i<11;i+
+){
System.out.pri
ntln(7*i);
}long time=System.nanoTime()-start;
System.out.println(Thread.currentThread()+" took "+time+" nanoseconds to
execute");
}
}
class Thirteen implements
Runnable{ public
void run(){
int i;long
start=System.currentTimeMillis();
for(i=1;i<11;i++){
System.out.println(13*i);
}long time=System.currentTimeMillis()-start;
System.out.println(Thread.currentThread()+" took "+time+" milliseconds to
execute");
}
}
class Table{
public static void
main(String str[]){ Five
a=new Five();
Thirteen c=new
Thirteen(); Seven
b=new Seven();
Thread t=new
Thread(a); Thread
t1=new Thread(b);
Thread t2=new
Thread(c);
t.start();
t1.start();
t2.start();
}
}
```

Output:

```
5
10
15
```

```
20
25
30
35
40
13
26
39
52
65
78
91
104
117
130
7
14
21
28
35
42
49
56
63
70
Thread[Thread-2,5,main] took 36 milliseconds to execute 45 50
Thread[Thread-1,5,main] took 60021200 nanoseconds to execute
Thread[Thread-0,5,main] took 70 milliseconds to execute
```

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

Theory:

Synchronization in Java is the capability to control the access of multiple threads to any shared resource

Code:

```
import java.util.*;
class syncexample extends Thread{
static String msg[]={"hi","there","this","is ,a","syncronized","message"};
public syncexample(String name){
super(name);
}
public synchronized void run(){
for (int i = 0; i < msg.length; i++) {
syncexample t=(syncexample)Thread.currentThread();
System.out.println(getName() + msg[i]);
}
}
public static void main(String str[]){
syncexample t1=new
syncexample("Thread A: ");
syncexample t2=new syncexample("Thread B: ");
t1.start();
t2.start();
}
}
```

Output:

```
Thread A: hi

Thread A: there

Thread A: this

Thread A: is a

Thread A: syncronized

Thread A: message

Thread B: hi

Thread B: there

Thread B: this

Thread B: is a

Thread B: syncronized

Thread B: message
```

Experiment 14

Aim: To implement Applets

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

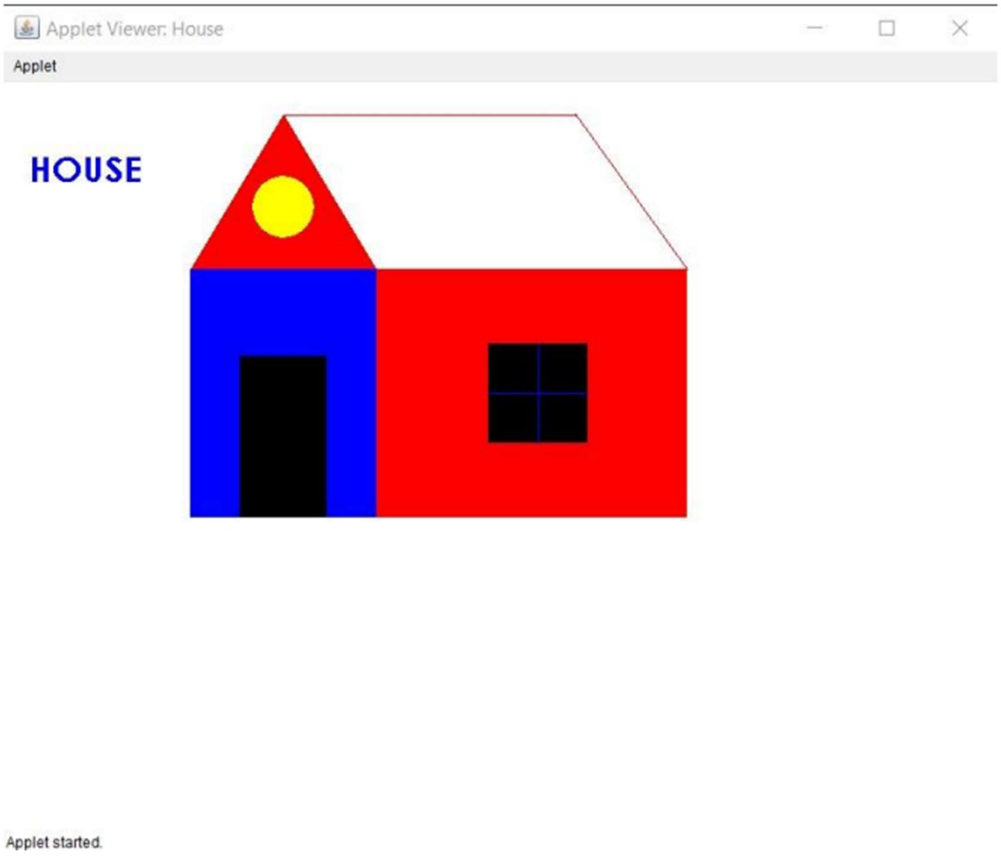
Theory:

Java applet is a java programming language that can be used to create a design, it is a bit unique people usually make a home design using Autocad, Corel Draw and other partya, but this time I made a minimalist home design using java applet.

Code:

```
import java.applet.*;
import java.awt.*;
/*<applet code="House" height="600" width="800"></applet>*/
public class House extends Applet
{
public void paint(Graphics g)
{
 Font home=new Font("Century Gothic",Font.BOLD,28);
 g.setFont(home);
 g.setColor(Color.blue);
 g.drawString("HOUSE",20,80);
 int a[]={150,300,225};
 int b[]={150,150,25};
 g.setColor(Color.blue);
 g.fillRect(150,150,150,200);

 g.setColor(Color.black);
 g.fillRect(190,220,70,130);

 g.setColor(Color.red);
 g.fillPolygon(a,b,3);

 g.setColor(Color.yellow);
 g.fillOval(200,75,50,50);
 g.setColor(Color.red);
 g.fillRect(300,150,250,200);

 g.setColor(Color.black);
 g.fillRect(390,210,80,80);
 g.setColor(Color.blue);
 g.drawLine(430,210,430,290);
 g.setColor(Color.blue);
 g.drawLine(470,250,390,250);
 g.setColor(Color.red);
 g.drawLine(460,26,226,26);
 g.setColor(Color.red);
 g.drawLine(460,25,550,150);
}
}
```

Output:



---

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

Theory:

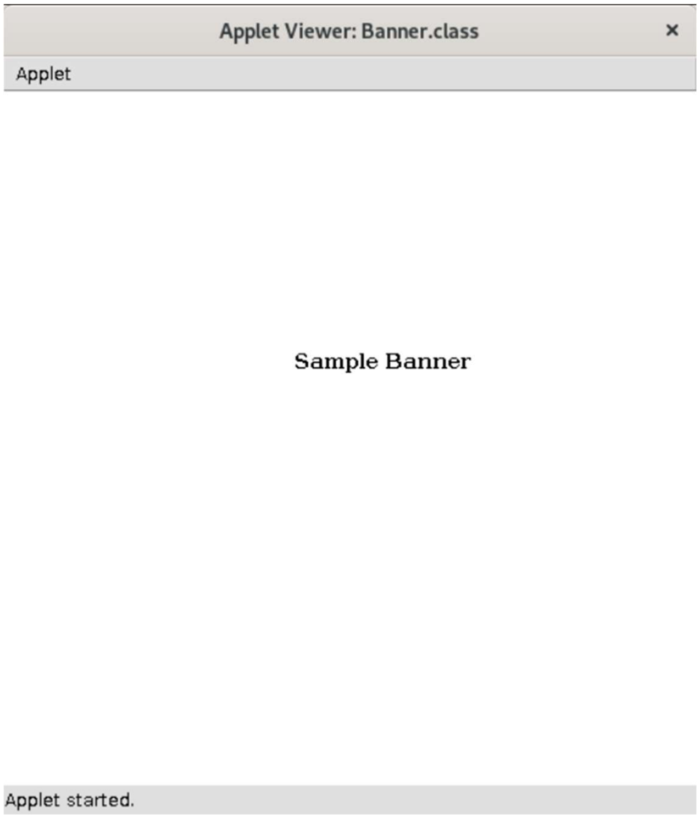
- Applets are small Java applications which can be accessed on an Internet server, transported over the Internet, and can be installed and run automatically as part of a web document.

- The applet can create a graphical user interface after a user gets an applet. It has restricted access to resources so that complicated computations can be carried out without adding the danger of viruses or infringing data integrity.

Code:

```
import java.applet.*;
import java.awt.*;
public class Banner extends Applet implements Runnable
{
    String text = " Sample Banner ";
    Thread t;
    //Initialize the applet
    public void init()
    {
        setBackground(Color.white);
    }
    //Function to start the thread
    public void start()
    {
            t = new Thread(this);
            t.start();
    }
    //Function to execute the thread
    public void run()
    {
        while(true)
            {
                try
                {
                    repaint();
                         //Delay each thread by 1000ms or 1 seconds
                Thread.sleep(1000);
                         //Shift the first character of banner text to the last postion
                text = text.substring(1) + text.charAt(0);
                }
                catch(Exception e)
                {
                }
            }
    }
    //Function to draw text
    public void paint(Graphics g)
    {
            g.setFont(new Font("TimesRoman",Font.BOLD,15));
            g.drawString(text,200,200);
    }
}
```

Output:



```
Sample Banner
```

Applet started.

Experiment 15

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

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

Theory:

Building GUI in Java requires the use of Swings. **Swing Framework** contains a large set of components which allow a high level of customization and provide rich functionalities, and is used to create window-based applications. Java swing components are lightweight, platform-independent, provide powerful components like tables, scroll panels, buttons, list, colour chooser, etc.

Code:

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

class MyFrame
    extends JFrame
    implements ActionListener {

    // Components of the Form
    private Container c;
    private JLabel title;
    private JLabel name;
    private JTextField tname;
    private JLabel mno;
    private JTextField tmno;
    private JLabel gender;
    private JRadioButton male;
    private JRadioButton female;
    private ButtonGroup gengp;
    private JLabel dob;
    private JComboBox date;
    private JComboBox month;
    private JComboBox year;
    private JLabel add;
    private JTextArea tadd;
    private JCheckBox term;
    private JButton sub;
    private JButton reset;
    private JTextArea tout;
    private JLabel res;
    private JTextArea resadd;

    private String dates[]
        = { "1", "2", "3", "4", "5",
            "6", "7", "8", "9", "10",
            "11", "12", "13", "14", "15",
            "16", "17", "18", "19", "20",
            "21", "22", "23", "24", "25",
            "26", "27", "28", "29", "30",
            "31" };
    private String months[]
        = { "Jan", "feb", "Mar", "Apr",
            "May", "Jun", "July", "Aug",
            "Sup", "Oct", "Nov", "Dec" };
    private String years[]
        = { "1995", "1996", "1997", "1998",
        "1999", "2000", "2001", "2002",
        "2003", "2004", "2005", "2006",
        "2007", "2008", "2009", "2010",
        "2011", "2012", "2013", "2014",
        "2015", "2016", "2017", "2018",
        "2019" };

// constructor, to initialize the components
// with default values.
public MyFrame()
{
    setTitle("Registration Form");
    setBounds(300, 90, 900, 600);
    setDefaultCloseOperation(EXIT_ON_CLOSE);
    setResizable(false);

    c = getContentPane();
    c.setLayout(null);

    title = new JLabel("Registration Form");
    title.setFont(new Font("Arial", Font.PLAIN, 30));
    title.setSize(300, 30);
    title.setLocation(300, 30);
    c.add(title);

    name = new JLabel("Name");
    name.setFont(new Font("Arial", Font.PLAIN, 20));
    name.setSize(100, 20);
    name.setLocation(100, 100);
    c.add(name);

    tname = new JTextField();
    tname.setFont(new Font("Arial", Font.PLAIN, 15));
    tname.setSize(190, 20);
    tname.setLocation(200, 100);
    c.add(tname);

    mno = new JLabel("Mobile");
    mno.setFont(new Font("Arial", Font.PLAIN, 20));
    mno.setSize(100, 20);
    mno.setLocation(100, 150);
    c.add(mno);

    tmno = new JTextField();
    tmno.setFont(new Font("Arial", Font.PLAIN, 15));
    tmno.setSize(150, 20);
    tmno.setLocation(200, 150);
    c.add(tmno);
```

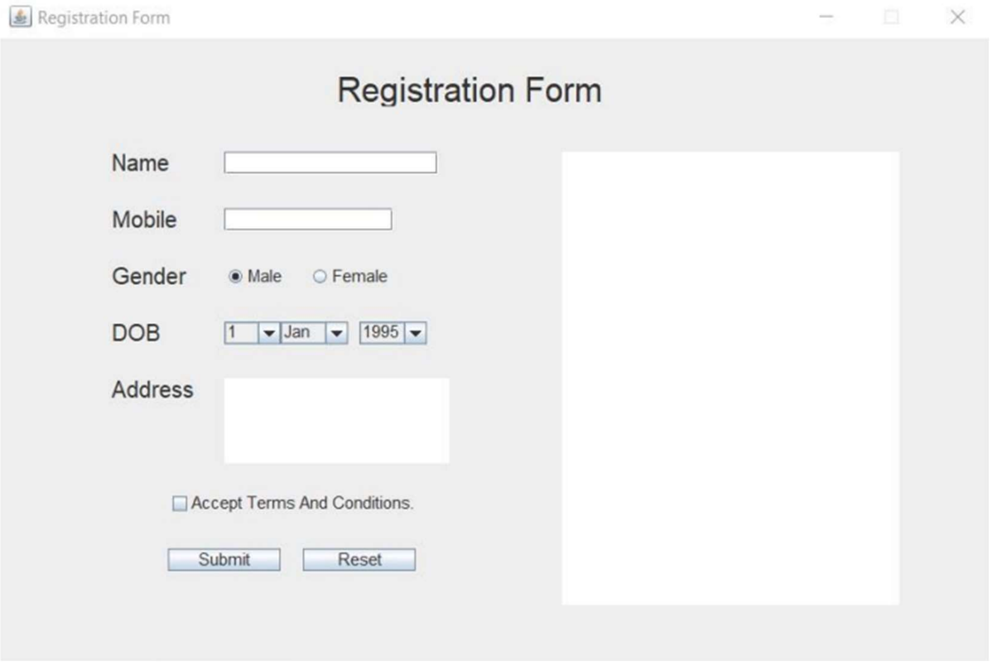
```
gender = new JLabel("Gender");
gender.setFont(new Font("Arial", Font.PLAIN, 20));
gender.setSize(100, 20);
gender.setLocation(100, 200);
c.add(gender);

male = new JRadioButton("Male");
male.setFont(new Font("Arial", Font.PLAIN, 15));
male.setSelected(true);
male.setSize(75, 20);
male.setLocation(200, 200);
c.add(male);

female = new JRadioButton("Female");
female.setFont(new Font("Arial", Font.PLAIN, 15));
female.setSelected(false);
female.setSize(80, 20);
female.setLocation(275, 200);
c.add(female);

gengp = new ButtonGroup();
gengp.add(male);
gengp.add(female);

dob = new JLabel("DOB");
dob.setFont(new Font("Arial", Font.PLAIN, 20));
dob.setSize(100, 20);
dob.setLocation(100, 250);
c.add(dob);

date = new JComboBox(dates);
date.setFont(new Font("Arial", Font.PLAIN, 15));
date.setSize(50, 20);
date.setLocation(200, 250);
c.add(date);

month = new JComboBox(months);
month.setFont(new Font("Arial", Font.PLAIN, 15));
month.setSize(60, 20);
month.setLocation(250, 250);
c.add(month);

year = new JComboBox(years);
year.setFont(new Font("Arial", Font.PLAIN, 15));
year.setSize(60, 20);
year.setLocation(320, 250);
c.add(year);
```

```
add = new JLabel("Address");
add.setFont(new Font("Arial", Font.PLAIN, 20));
add.setSize(100, 20);
add.setLocation(100, 300);
c.add(add);

tadd = new JTextArea();
tadd.setFont(new Font("Arial", Font.PLAIN, 15));
tadd.setSize(200, 75);
tadd.setLocation(200, 300);
tadd.setLineWrap(true);
c.add(tadd);

term = new JCheckBox("Accept Terms And Conditions.");
term.setFont(new Font("Arial", Font.PLAIN, 15));
term.setSize(250, 20);
term.setLocation(150, 400);
c.add(term);

sub = new JButton("Submit");
sub.setFont(new Font("Arial", Font.PLAIN, 15));
sub.setSize(100, 20);
sub.setLocation(150, 450);
sub.addActionListener(this);
c.add(sub);

reset = new JButton("Reset");
reset.setFont(new Font("Arial", Font.PLAIN, 15));
reset.setSize(100, 20);
reset.setLocation(270, 450);
reset.addActionListener(this);
c.add(reset);

tout = new JTextArea();
tout.setFont(new Font("Arial", Font.PLAIN, 15));
tout.setSize(300, 400);
tout.setLocation(500, 100);
tout.setLineWrap(true);
tout.setEditable(false);
c.add(tout);

res = new JLabel("");
res.setFont(new Font("Arial", Font.PLAIN, 20));
res.setSize(500, 25);
res.setLocation(100, 500);
c.add(res);
```

```
    resadd = new JTextArea();
    resadd.setFont(new Font("Arial", Font.PLAIN, 15));
    resadd.setSize(200, 75);
    resadd.setLocation(580, 175);
    resadd.setLineWrap(true);
    c.add(resadd);

    setVisible(true);
}

// method actionPerformed()
// to get the action performed
// by the user and act accordingly
public void actionPerformed(ActionEvent e)
{
    if (e.getSource() == sub) {
        if (term.isSelected()) {
            String data1;
            String data
                = "Name : "
                  + tname.getText() + "\n"
                  + "Mobile : "
                  + tmno.getText() + "\n";
            if (male.isSelected())
                data1 = "Gender : Male"
                        + "\n";
            else
                data1 = "Gender : Female"
                        + "\n";
            String data2
                = "DOB : "
                  + (String)date.getSelectedItem()
                  + "/" + (String)month.getSelectedItem()
                  + "/" + (String)year.getSelectedItem()
                  + "\n";

            String data3 = "Address : " + tadd.getText();
            tout.setText(data + data1 + data2 + data3);
            tout.setEditable(false);
            res.setText("Registration Successfully..");
        }
        else {
            tout.setText("");
            resadd.setText("");
            res.setText("Please accept the"
                        + " terms & conditions..");
        }
    }
```

```
        else if (e.getSource() == reset) {
            String def = "";
            tname.setText(def);
            tadd.setText(def);
            tmno.setText(def);
            res.setText(def);
            tout.setText(def);
            term.setSelected(false);
            date.setSelectedIndex(0);
            month.setSelectedIndex(0);
            year.setSelectedIndex(0);
            resadd.setText(def);
        }
    }
}

class Registration {

    public static void main(String[] args) throws Exception
    {
        MyFrame f = new MyFrame();
    }
}
```

Output:



Registration Form

Name

Mobile

Gender ● Male ○ Female

DOB 1 Jan 1995

Address

□ Accept Terms And Conditions.

Experiment 16

Aim: 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:

Java Swing tutorial is a part of Java Foundation Classes (JFC) that is *used to create window-based applications*. It is built on the top of AWT (Abstract Windowing Toolkit) API and entirely written in java.

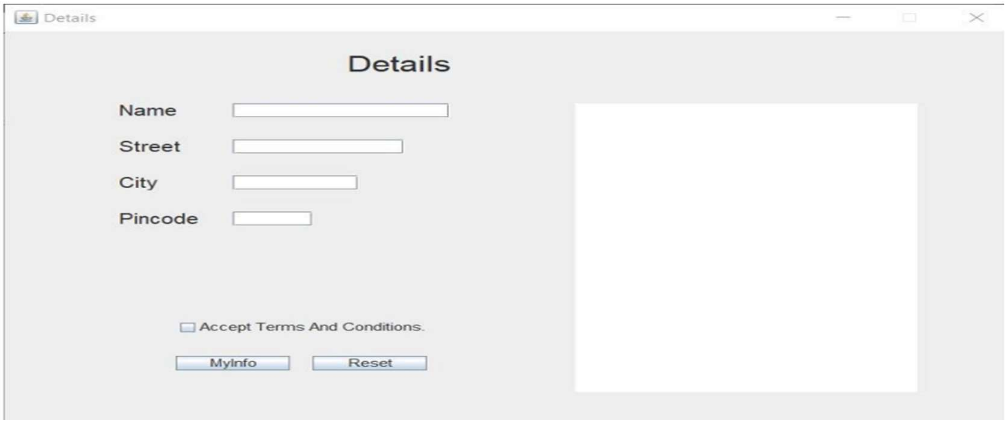
Unlike AWT, Java Swing provides platform-independent and lightweight components.

The javax.swing package provides classes for java swing API such as JButton, JTextField, JTextArea, JRadioButton, JCheckbox, JMenu, JColorChooser etc.

Code:

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
class MyFrame
 extends JFrame
 implements ActionListener {

 private Container c;
 private JLabel title;
 private JLabel name;
 private JTextField tname;
 private JLabel street;
 private JTextField tstreet;
 private JLabel city;
 private JTextField tcity;
 private JLabel pincode;
 private JTextField tpincode;
 private JCheckBox term;
 private JButton myinfo;
 private JButton reset;
 private JTextArea tout;
 private JLabel res;
 private JTextArea resadd;

public MyFrame()
{
 setTitle("Details");
 setBounds(300, 90, 900, 600);
 setDefaultCloseOperation(EXIT_ON_CLOSE);
 setResizable(false);
 c = getContentPane();
 c.setLayout(null);
 title = new JLabel("Details");
 title.setFont(new Font("Arial", Font.PLAIN, 30));
 title.setSize(300, 30);
 title.setLocation(300, 30);
 c.add(title);

 name = new JLabel("Name");
 name.setFont(new Font("Arial", Font.PLAIN, 20));
 name.setSize(100, 20);
 name.setLocation(100, 100);
 c.add(name);
 tname = new JTextField();
 tname.setFont(new Font("Arial", Font.PLAIN, 15));
 tname.setSize(190, 20);
 tname.setLocation(200, 100);
 c.add(tname);
 street = new JLabel("Street");
 street.setFont(new Font("Arial", Font.PLAIN, 20));
 street.setSize(100, 20);
 street.setLocation(100, 150);
 c.add(street);
 tstreet = new JTextField();
 tstreet.setFont(new Font("Arial", Font.PLAIN, 15));
 tstreet.setSize(150, 20);
 tstreet.setLocation(200, 150);
 c.add(tstreet);
 city = new JLabel("City");
 city.setFont(new Font("Arial", Font.PLAIN, 20));
 city.setSize(100, 20);
 city.setLocation(100, 200);
 c.add(city);
 tcity = new JTextField();
 tcity.setFont(new Font("Arial", Font.PLAIN, 15));
 tcity.setSize(110, 20);
 tcity.setLocation(200, 200);
 c.add(tcity);
 pincode = new JLabel("Pincode");
 pincode.setFont(new Font("Arial", Font.PLAIN, 20));
 pincode.setSize(100, 20);
 pincode.setLocation(100, 250);
 c.add(pincode);
 tpincode = new JTextField();
 tpincode.setFont(new Font("Arial", Font.PLAIN, 15));
 tpincode.setSize(70, 20);
 tpincode.setLocation(200, 250);
 c.add(tpincode);
 term = new JCheckBox("Accept Terms And Conditions.");
 term.setFont(new Font("Arial", Font.PLAIN, 15));
 term.setSize(250, 20);
 term.setLocation(150, 400);
 c.add(term);

 myinfo = new JButton("MyInfo");
 myinfo.setFont(new Font("Arial", Font.PLAIN, 15));
 myinfo.setSize(100, 20);
 myinfo.setLocation(150, 450);
 myinfo.addActionListener(this);
 c.add(myinfo);
 reset = new JButton("Reset");
 reset.setFont(new Font("Arial", Font.PLAIN, 15));
 reset.setSize(100, 20);
 reset.setLocation(270, 450);
 reset.addActionListener(this);
 c.add(reset);
 tout = new JTextArea();
 tout.setFont(new Font("Arial", Font.ITALIC, 32));
 tout.setSize(300, 400);
 tout.setLocation(500, 100);
 tout.setLineWrap(true);
 tout.setEditable(false);
 c.add(tout);
 res = new JLabel("");
 res.setFont(new Font("Arial", Font.PLAIN, 20));
 res.setSize(500, 25);
 res.setLocation(100, 500);
 c.add(res);

 resadd = new JTextArea();
 resadd.setFont(new Font("Arial", Font.PLAIN, 15));
 resadd.setSize(200, 75);
 resadd.setLocation(580, 175);
 resadd.setLineWrap(true);
 c.add(resadd);

 setVisible(true);
 }
public void actionPerformed(ActionEvent e)
 {
 if (e.getSource() == myinfo) {
 if (term.isSelected()) {
 String data1;
 String data
 = "Name : "
 + tname.getText() + "\n"
 + "Street : "
 + tstreet.getText() + "\n"
 + "City : "
 + tcity.getText() + "\n"
 + "Pincode : "
 + tpincode.getText() + "\n";
 tout.setText(data);
 tout.setEditable(false);
 res.setText("Details entered successfully");
 }
 else {
 tout.setText("");
 resadd.setText("");
 res.setText("Please accept the"
 + " terms & conditions..");
 }
 }
else if (e.getSource() == reset) {
 String def = "";
 tname.setText(def);
 tstreet.setText(def);
 tcity.setText(def);
 tpincode.setText(def);
 res.setText(def);
 tout.setText(def);
 term.setSelected(false);
 resadd.setText(def);

 }
 }
}
class Details {

 public static void main(String[] args) throws Exception
 {
 MyFrame f = new MyFrame();
 }
}
```

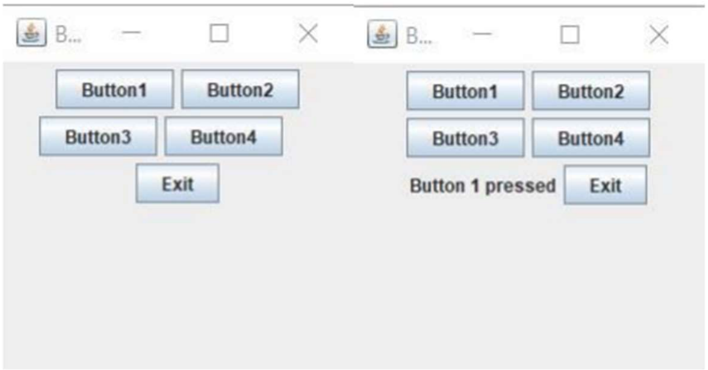
Output:



16.2)WAP 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

Code:

```
import java.awt.*;
import javax.swing.*;
import java.awt.event.*;
class btnPress extends JFrame implements ActionListener
{
Container c;
JLabel L1;
JButton B1,B2,B3,B4,exit;
btnPress()
{
```

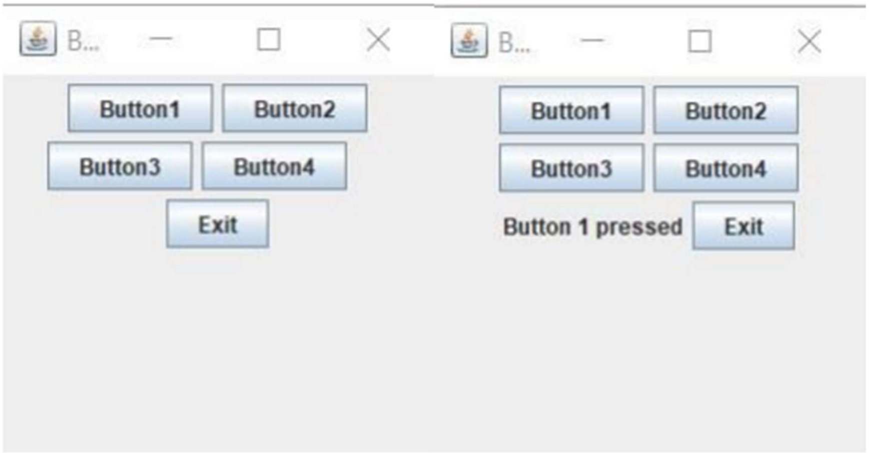


Button 1 pressed

```
c=getContentPane();
c.setLayout(new FlowLayout());
B1=new JButton("Button1");
B2=new JButton("Button2");
B3=new JButton("Button3");
B4=new JButton("Button4");
L1=new JLabel(" ");
exit=new JButton("Exit");
c.add(B1);
c.add(B2);
c.add(B3);
c.add(B4);
c.add(L1);
c.add(exit);
B1.addActionListener(this);
B2.addActionListener(this);
B3.addActionListener(this);
B4.addActionListener(this);
exit.addActionListener(this);
}
public void actionPerformed(ActionEvent ae)
{
if(ae.getSource()==B1)
```

```
{
L1.setText("Button 1 pressed");
}
else if(ae.getSource()==B2)
{
L1.setText("Button 2 pressed");
}
else if(ae.getSource()==B3)
{
L1.setText("Button 3 pressed");
}
else if(ae.getSource()==B4)
{
L1.setText("Button 4 pressed");
}
else if(ae.getSource()==exit)
{
System.exit(0);
}
else {}
}
public static void main(String z[]){
btnPress frm=new btnPress();
frm.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
frm.setBounds(200,200,250,250);
frm.setVisible(true);
frm.setTitle("Button Event");
}
}
```

Output:



```
Button 1 pressed
```