Aim: A program to demonstrate about local variables , instance variables and static variables

Program:

```
class      VariablesDemo
{
          int     a=10,b=20;     //instance variables
          static  int  e=600,f=900;    //static variables
          void  sum()
          {
                    int c=100,d=400;    // local   variable
                    System.out.println(c+d);
          }
          public static void main(String[]  args)
          {
                    VariablesDemo     t   =  new  VariablesDemo();
                              t.sum();
          System.out.println(VariablesDemo.e+VariablesDemo.f);
          }
}

/*   output
500
1500    */
```

Aim : To Demonstrate about data types size , minimum value and maximum value

```
class      DataTypes
{


          public  static  void main(String[]     args)
          {

System.out.println(Byte.SIZE+" "+Byte.MIN_VALUE+" "+Byte.MAX_VALUE);
System.out.println(Short.SIZE+" "+Short.MIN_VALUE+" "+Short.MAX_VALUE);
System.out.println(Integer.SIZE+" "+Integer.MIN_VALUE+" "+Integer.MAX_VALUE);
System.out.println(Long.SIZE+" "+Long.MIN_VALUE+" "+Long.MAX_VALUE);
System.out.println(Float.SIZE+" "+Float.MIN_VALUE+" "+Float.MAX_VALUE);
System.out.println(Double.SIZE+" "+Double.MIN_VALUE+" "+Double.MAX_VALUE);
System.out.println(Character.SIZE+" "+Character.MIN_VALUE+"
"+Character.MAX_VALUE);

        }
}
/*    Output

E:\AI>javac DataTypes.java

E:\AI>java DataTypes
8 -128 127
16 -32768 32767
32 -2147483648 2147483647
64 -9223372036854775808 9223372036854775807
32 1.4E-45 3.4028235E38
64 4.9E-324 1.7976931348623157E308
16   ?      */
```

Aim: A Program to find factorial of given number

Program:

```
import java.util.Scanner;
class    FactorialOfNumber
{
        void fact(int num)
        {
                int fact=1;
                for(int i=1;i<=num;i++)
                {
                        fact=fact*i;
                }
                System.out.println(fact);
        }
}
public class     Factorial
{
        public   static  void main(String[]  args)
        {
                int num;
```

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

          System.out.println("Enter a number ");
          num=scan.nextInt();

          FactorialOfNumber      f   =   new  FactorialOfNumber();
                    f.fact(num);

          }

}
/*   Output
E:\AI>javac Factorial.java

E:\AI>java Factorial
Enter a number
5
120

E:\AI>java Factorial
Enter a number
6
720      */
```

Object Oriented Programming Through Java Lab -- (20A05302P)

Week 1

A)

Installation of Java software, study of any Integrated development environment, Use Eclipse or Netbeans platform and acquaint with the various menus.

Create a test project, add a test class and runit.

See how you can use auto suggestions, auto fill. Try code formatter and code refactoring like renaming variables, methods and classes. Try debug step by step with java program to find prime numbers between 1 to n.

1) Installation of Java software

Step 1 - Download JDK for windows using below link

Downloading Java JDK 19.0.0.0 (64-bit) from FileHorse.com

Step -2 Install by clicking on executable file

Step-3 After installation set the path to environment variables

C:\Program Files (x86)\Java\jdk1.8.0_60\bin;

Steps to install Eclipse IDE:

1. Download and install JDK(it is a pre-requisite essential step)
2. Go to eclipse website and download choosing the version as operating system and bits requirement.
3. Open the downloaded file extension and follow the standard software installation process.
4. Choose package as per developer language needs.
5. A new window will be relaunch and if not relaunch eclipse.
6. Go to the new project and create classes inside which java applications(or programs) are good to go.

Aim : Program to find all Prime Numbers between 1 to N

Description:

- Prime number in Java: Prime number is a number that is greater than 1 and divided by 1 or itself only.
- In other words, prime numbers can't be divided by other numbers than itself or 1.
- For example 2, 3, 5, 7, 11, 13, 17.... are the prime numbers.

Algorithm:

- Get the upper limit from the user and store it in the variable **"N"**
- Start the loop from **2** to **N**, for each iteration increment the loop by **1**
- In the **checkPrime()** method, we have used a boolean flag. It will be set to **false** when the number is less than **1** or if the number is divisible by **number/2.**
- Validate the boolean returned by the **checkPrime()** and print the **number** if the **boolean** returned is **true**.

Program:

```
import java.util.Scanner;

public class PrimeNumber
{
        public static void main(String[] args)
        {
                Scanner scanner = new Scanner(System.in);
                System.out.println("Enter the Upper limit :");
                int N = scanner.nextInt();

                System.out.println("*** Prime Numbers between 1 to N ***");

                for (int i = 2; i <= N; i++)
                {
                    if (checkPrime(i))
                    {
                        System.out.print(i+"  ");
                    }
                }
        }

        public static boolean checkPrime(int n)
        {
                boolean flag = true;

                if(n <= 1)
                    flag = false;

                for(int i=2; i<= n/2; i++)
                {
                    if((n % i) == 0)
                      {
                        flag = false;
                          break;
                    }
                }

                    return flag;
        }
}
```

Output:

```
E:\AI>javac  PrimeNumber.java

E:\AI>java PrimeNumber
Enter the Upper limit :
30
*** Prime Numbers between 1 to N ***
2  3  5  7  11  13  17  19  23  29
```

B)

Aim: A Java program that prints all real solutions to the quadratic equation $ax^2+bx+c=0$.

Read in a,b, c and use the quadratic formula.

Description:

- The standard form of a quadratic equation is $\mathbf{ax^2+bx+c=0}$. It is also known as the second-degree equation.
- In the equation $ax^2+bx+c=0$, a, b, and c are unknown values and a cannot be 0. x is an unknown variable.
- The formula to find the roots of the quadratic equation is known as the quadratic formula.

$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

- A quadratic equation has two roots and the roots depend on the discriminant.
- In the above formula, ($\sqrt{\boxed{b^2\text{-}4ac}}$) is called **discriminant (d)**.

- The value of d may be positive, negative, or zero.

If d is positive (d>0), the root will be:

If the value of d is **positive**, both roots are real and different. It means there are two real solutions.

$$x1 = \frac{-b + \sqrt{b^2 - 4ac}}{2a}$$

$$x2 = \frac{-b - \sqrt{b^2 - 4ac}}{2a}$$

If d is zero (d=0), the root will be:

If the value of d is **zero**, both roots are real and the same. It means we get one real solution.

$$x1 = x2 = \frac{-b}{2a}$$

If d is negative (d<0), the root will be:

If the value of d is **negative**, both roots are distinct and imaginary or complex.
It means that there are two complex solutions.

$$x1 = \frac{-b}{2a} + i\frac{\sqrt{-(b^2 - 4ac)}}{2a}$$

$$x2 = \frac{-b}{2a} - i\frac{\sqrt{-(b^2 - 4ac)}}{2a}$$

Algorithm :

Step 1: Start

Step 2: Read a, b, c

Step 3: initialize d<-(b*b)-(4*a*c)

Step 4: initialize r<- b/2*a

Step 5: if d>0 go to **Step 6**, else go to **Step 8**

Step 6: r1=r+(sqrt(d)/2*a) and r2=r-(sqrt(d)/2*a)

Step 7: prints roots are real and distinct, first root r1 second root r2

Step 8: if d=0 go to **Step 9**, else go to **Step 10**

Step 9: print roots are real and equal, -r

Step 10: d=-d

Step 11: im=sqrt(d)/2*a

Step 12: print roots are imaginary, first root is r+i im, and the second root is r-i im

Step 13: Stop

Program:

```
 import java.util.Scanner;


public class QuadraticEquationExample
{
        public static void main(String[] Strings)
        {
              double a,b,c,d;
              Scanner input = new Scanner(System.in);

              System.out.print("Enter the value of a: ");
               a = input.nextDouble();

              System.out.print("Enter the value of b: ");
               b = input.nextDouble();
```

```
            System.out.print("Enter the value of c: ");
             c = input.nextDouble();


             d= b * b - 4.0 * a * c;

        if (d> 0.0)
        {
            double r1 = (-b + Math.pow(d, 0.5)) / (2.0 * a);
            double r2 = (-b - Math.pow(d, 0.5)) / (2.0 * a);
            System.out.println("The roots are " + r1 + " and " + r2);
        }
        else
        if (d == 0.0)
        {
            double r1 = -b / (2.0 * a);
            System.out.println("The root is " + r1);
        }
        else
        {
            System.out.println("Roots are not real.");
        }
    }

}
```

Output:

```
    E:\AI>java QuadraticEquationExample


    Enter the value of a: 1

    Enter the value of b: 2

    Enter the value of c: 3

    Roots are not real.

E:\AI>java QuadraticEquationExample


Enter the value of a: 1

Enter the value of b: 5

Enter the value of c: 2

The roots are -0.4384471871911697 and -4.561552812808831
```

C)

Aim: Develop a Java application to generate Electricity bills.

Create a class with the following members:

Consumer no , consumer name, previous month reading, current month reading, type of EB connection (i.e domestic or commercial).

If the type of the EB connection is domestic, calculate the amount to be paid as follows:

- First 100 units - Rs. 1 per unit
- 101-200 units - Rs. 2.50 per unit
- 201 -500 units - Rs. 4 per unit
- > 501 units - Rs. 6 per unit

If the type of the EB connection is commercial, calculate the amount to be paid as follows:

- First 100 units - Rs. 2 per unit
- 101-200 units - Rs. 4.50 per unit
- 201 -500 units - Rs. 6 per unit
- > 501 units - Rs. 7 per unit

Description:

Step 1: Declare all variables as per our requirement
Step 2: Read all variables data
Step 3: Calculate bill amount based on commercial or domestic
Step 4: Display bill amount

Program:

```
import java.util.*;

public class ComputeElectricityBill
{
        int        consumerno;
        String     consumername;
        int        prev_reading;
        int        curr_reading;
        String     Ebconn;
        double    bill;
```

```
void   readData()
{
        Scanner scan   =  new Scanner(System.in);

        System.out.println(" Enter  consumer  number ");
        consumerno  =   scan.nextInt();

        System.out.println(" Enter  consumer  name ");
        consumername  =   scan.next();

        System.out.println(" Enter  previous reading ");
        prev_reading   =   scan.nextInt();

        System.out.println(" Enter  current reading ");
        curr_reading =  scan.nextInt();

        System.out.println(" Enter  the type of connection  ");
        Ebconn  =   scan.next();
    }


void caluclate_bill_amount()
{
        if(Ebconn.equals("domestic"))
        {
              int units;

               units=curr_reading-prev_reading;

                if(units<=100)
                {
                        bill=units*1;
                }
                else
                if(units>=101   &&  units<=200)
                {
                        bill=units*2.50;
                }
                else
                if(units>=201   &&  units<=500)
                {
                        bill=units*4;
                }
                else
                if(units>=501)
                {
                        bill=units*6;
                }

        }
        else
        {
                int units;

                units=curr_reading-prev_reading;

                if(units<=100)
                {
                        bill=units*2;
                }
                else
                if(units>=101   &&  units<=200)
                {
                        bill=units*4.50;
                }
                else
                if(units>=201   &&  units<=500)
                {
                        bill=units*6;
                }
                else
                if(units>=501)
                {
                        bill=units*7;
                }

        }

}
```

```
         void displayData()
        {
            System.out.println(" consumer  number = "+consumerno);
            System.out.println(" consumer  name = "+consumername);
            System.out.println(" previous reading = "+prev_reading);
            System.out.println(" current reading = "+curr_reading);
            System.out.println(" Type of connection  = "+ Ebconn);
            System.out.println(" Bill amount  = "+bill);
         }

        public static void main(String[] args)
        {
             ComputeElectricityBill     b    =   new  ComputeElectricityBill();
                                b.readData();
                                b.caluclate_bill_amount();
                                b.displayData();
        }
}
```

Output:

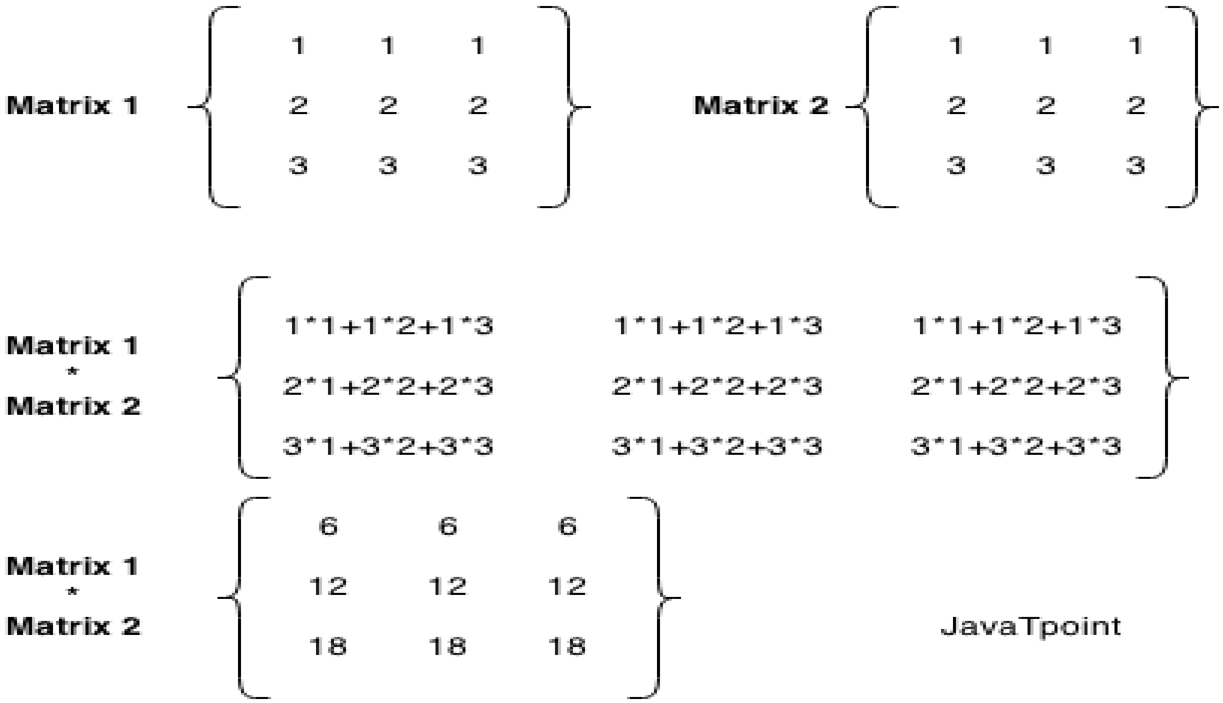
```
            E:\AI>java ComputeElectricityBill

 Enter  consumer  number
102
 Enter  consumer  name
mass
 Enter  previous reading
100
 Enter  current reading
300
 Enter  the type of connection
Commertial
 consumer  number = 102
 consumer  name = mass
 previous reading = 100
 current reading = 300
 Type of connection  = Commertial
 Bill amount  = 900.0
```

D)

Aim: Write a Java program to multiply two given matrices

Description:

- We can multiply two matrices in java using binary * operator and executing another loop.
- A matrix is also known as array of arrays. We can add, subtract and multiply matrices.
- In case of matrix multiplication, one row element of first matrix is multiplied by all columns of second matrix.



$$\text{Matrix 1} \left\{ \begin{matrix} 1 & 1 & 1 \\ 2 & 2 & 2 \\ 3 & 3 & 3 \end{matrix} \right\} \qquad \text{Matrix 2} \left\{ \begin{matrix} 1 & 1 & 1 \\ 2 & 2 & 2 \\ 3 & 3 & 3 \end{matrix} \right\}$$

$$\begin{matrix}\text{Matrix 1} \\ * \\ \text{Matrix 2}\end{matrix} \left\{ \begin{matrix} 1*1+1*2+1*3 & 1*1+1*2+1*3 & 1*1+1*2+1*3 \\ 2*1+2*2+2*3 & 2*1+2*2+2*3 & 2*1+2*2+2*3 \\ 3*1+3*2+3*3 & 3*1+3*2+3*3 & 3*1+3*2+3*3 \end{matrix} \right\}$$

$$\begin{matrix}\text{Matrix 1} \\ * \\ \text{Matrix 2}\end{matrix} \left\{ \begin{matrix} 6 & 6 & 6 \\ 12 & 12 & 12 \\ 18 & 18 & 18 \end{matrix} \right\}$$

Algorithm:

Step 1: Declare two matrices and assign values and declare third matrix to store product

Step 2: Multiply two matrices and display output

Step 3: use inner loops to multiply matrices like below

```
for(int i=0;i<3;i++)
{
        for(int j=0;j<3;j++)
        {
                        c[i][j]=0;

                for(int k=0;k<3;k++)
                {
                        c[i][j]+=a[i][k]*b[k][j];
                }
                System.out.print(c[i][j]+" ");  //printing matrix element
        }
                System.out.println();     //new line
}
```

Program :

```
public   class    MatrixMultiplicationExample
{

        public static void main(String args[])
        {
                //creating two matrices

                int      a[][]     =     {     {1,1,1},{2,2,2},{3,3,3}      };
                int      b[][]     =     {      {1,1,1},{2,2,2},{3,3,3}     };

                int    c[][]    =     new    int[3][3];     //3x3  matrix to store multiplication

        //multiplying and printing multiplication of 2 matrices

          for(int i=0;i<3;i++)
          {
                for(int j=0;j<3;j++)
                {
                        c[i][j]=0;

                        for(int k=0;k<3;k++)
                        {
                                c[i][j]+=a[i][k]*b[k][j];
                        }
                        System.out.print(c[i][j]+" ");  //printing matrix element
                }
                System.out.println();     //new line
        }
      }
}
```

Output :

```
E:\AI>javac  MatrixMultiplicationExample.java

E:\AI>java  MatrixMultiplicationExample

6 6 6
12 12 12
18 18 18
```