EXPERIMENT: 1

AIM: Basic programs using data types, operators, and control statements in Java.

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, and type of EB connection (i.e. domestic or commercial). Compute the bill amount using the following tariff.

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

PROGRAM:

```
import java.util.*;
public class Ebill
{
        public static void main (String args[])
        {
        Customerdata ob = new Customerdata();
        ob.getdata();
        ob.calc();
        ob.display();
        }
}
        class Customerdata
```

```
{
        Scanner in = new Scanner(System.in);
        Scanner ins = new Scanner(System.in);
        String cname,type;
        int bn;
        double current,previous,tbill,units;
        void getdata()
        {
        System.out.print ("\n\t Enter consumer number ");
        bn = in.nextInt();
        System.out.print ("\n\t Enter Type of connection (D for Domestic or C for
        Commercial) ");
        type = ins.nextLine();
        System.out.print ("\n\t Enter consumer name ");
        cname = ins.nextLine();
        System.out.print ("\n\t Enter previous month reading ");
        previous= in.nextDouble();
        System.out.print ("\n\t Enter current month reading ");
        current= in.nextDouble();
        }
        void calc()
        {
        units=current-previous;
        if(type.equals("D"))
        {
        if (units<=100)
        tbill=1 * units;
        else if (units>100 && units<=200)
        tbill=2.50*units;
        else if(units>200 && units<=500)
        tbill= 4*units;
        else
        tbill= 6*units;
        }
        else
        {
        if (units<=100)
        tbill= 2 * units;
        else if(units>100 && units<=200)
        tbill=4.50*units;
        else if(units>200 && units<=500)
        tbill= 6*units;
```

```
        else
        tbill= 7*units;
        }
}
        void display()
{
        System.out.println("\n\t Consumer number = "+bn);
        System.out.println ("\n\t Consumer name = "+cname);
        if(type.equals("D"))
        System.out.println ("\n\t type of connection = DOMESTIC ");
        else
        System.out.println ("\n\t type of connection = COMMERCIAL ");
        System.out.println ("\n\t Current Month Reading = "+current);
        System.out.println ("\n\t Previous Month Reading = "+previous);
        System.out.println ("\n\t Total units = "+units);
        System.out.println ("\n\t Total bill = RS "+tbill);

}
}
```

INPUT DATA:

OUTPUT DATA:

EXPERIMENT: 2

AIM: Objective: Basic programs using Arrays

Write a Java program to move all 0's to the end of an array. Maintain the relative order of the other (non-zero) array elements.

PROGRAM:

```
import java.util.*;
 public class Exercise26
 {
    public static void main(String[] args) throws Exception
 {
        int[] array_nums = {0,0,1,0,3,0,5,0,6};
        int i = 0;
        System.out.print("\nOriginal array: \n");
        for (int n : array_nums)
        System.out.print(n+"  ");

        for(int j = 0, l = array_nums.length; j < l;)
        {
        if(array_nums[j] == 0)
                j++;
        else
         {
                int temp = array_nums[i];
                array_nums[i] = array_nums[j];
                array_nums[j] = temp;
                i ++;
                j ++;
         }
        }
        while (i < array_nums.length)
        array_nums[i++] = 0;
        System.out.print("\nAfter moving 0's to the end of the array: \n");
        for (int n : array_nums)
        System.out.print(n+"  ");
                System.out.print("\n");
   }
}
```

INPUT DATA:

OUTPUT DATA;

EXPERIMENT: 3

AIM: Basic programs using Strings

Write a Java program to find the first non-repeating character in a string

PROGRAM:

```
import java.util.*;
public class Main
{
        public static void main(String[] args)
{
        String str1 = "gibblegabbler";
        System.out.println("The given string is: " + str1);
        for (int i = 0; i < str1.length(); i++)
        {
        boolean unique = true;
        for (int j = 0; j < str1.length(); j++)
        {
        if (i != j && str1.charAt(i) == str1.charAt(j))
        {
        unique = false;
        break;
         }
        }
  if (unique)
         {
         System.out.println("The first non repeated character in String is: " +
        str1.charAt(i));
   break;
        }
        }
}
}
```

INPUT DATA:

OUTPUT DATA:

EXPERIMENT: 4

AIM: Object Oriented Programming Concepts: Problem on the use of constructors, inheritance

Declare a class employee having emp_id and empname as members. Extend class employee (inheritance) to have a subclass called salary having designation and monthly_salary as members. Define the following:

– Required constructors.

– A method to find and display all details of employee drawing salary more than 20000/-.

– Method main for creating an array for storing these details given as command line argument and showing usage of above methods.text.

PROGRAM:

```
import java.util.Scanner;
class Employee
{
 String[] employee_id;
 String[] employee_name;
}

class Salary extends Employee
{
 String[] Designation;
 double[] monthly_salary;

 Salary(int j)
 {
  /*initialization of array */
  employee_name=new String[j];
  employee_id=new String[j];
  Designation=new String[j];
  monthly_salary= new double[j];

 }
```

```
 void display(int j)
 {


   System.out.println("-----------------------------------------------------------------");
   System.out.println("-----------------------------------------------------------------");
   System.out.println("\t Details of employee who have salary above 20000");
   System.out.println("-----------------------------------------------------------------");
   System.out.println("-----------------------------------------------------------------\n
\n");
  System.out.format("%-15s %-15s %-25s %-10s %n","employee id","employee
  name","employee Designation","Monthly Salary");
   System.out.println("-------------------------------------------------------------------------
------");
  for(int i=0;i<j;i++)
  {

   if(monthly_salary[i]>=20000)
   {
    System.out.format("%-15s %-15s %-25s %-10s
%n",employee_id[i],employee_name[i],Designation[i],monthly_salary[i]);

   }
  }
 }
 public static void main(String [] args)
 {
  Scanner jaimin=new Scanner(System.in);
  int length=args.length;

  Salary obj = new Salary(length);


  if(length==0)
  {
   System.out.println("please enter employee id");
  }

  for(int i=0;i<length;i++)
  {
    obj.employee_id[i]=args[i];
```

```
    System.out.println("\n\n enter the details of \""+args[i]+"\" employee id");

    System.out.print("\n name of employee -->");
    obj.employee_name[i]=jaimin.next();

    System.out.print("\n Designation of employee -->");
    obj.Designation[i]=jaimin.next();

    System.out.print("\nMonthly salary of employee -->");
    obj.monthly_salary[i]=jaimin.nextDouble();


  }

  obj.display(length);
 }
}
```

INPUT DATA:

OUTPUT DATA:

EXPERIMENT: 5

AIM: Object Oriented Programming Concepts: Problem on the use of Method Overloading and Overriding

Write a JAVA program to represent Method Overloading and Overriding.

PROGRAM:

```
package com.techvidvan.methodoverriding;
public class Addition
{
int add(int a, int b)
{
return (a + b);
}
int add(int a , int b , int c)
{
return (a + b + c) ;
}
double add(double a , double b)
{
return (a + b);
}
double add(int a , double b)
{
return (a + b);
}
public static void main( String args[])
{
Addition ob = new Addition();
System.out.println("Calling add method with two int parameters: " +ob.add(17,
25));
System.out.println("Calling add method with three int parameters: "
+ob.add(55, 27, 35));
System.out.println("Calling add method with two double parameters: "
+ob.add(36.5, 42.8));
System.out.println("Calling add method with one int and one double
parameter: " +ob.add(11, 24.5));
}
```

```
package com.techvidvan. methodoverriding;
//Base Class
class Parent
{
        void view()
{
        System.out.println("This is a parent class method");
}
}
class Child extends Parent
{
        @Override
        void view()
{
        System.out.println("This is a child class method");
}
}
        //Driver class
        public class MethodOverriding
{
         public static void main(String args[])
{
         Parent obj = new Parent();
         obj.view();
         Parent obj1 = new Child();
         obj1.view();
}
}
```

INPUT DATA:

OUTPUT DATA:

EXPERIMENT: 6

AIM: Object Oriented Programming Concepts: Problem on the use of Garbage collection

Write a JAVA program to represent Garbage Collection

PROGRAM:

```
class Employee
{

    private int ID;
    private String name;
    private int age;
    private static int nextId = 1;

    // it is made static because it
    // is keep common among all and
    // shared by all objects
    public Employee(String name, int age)
    {
        this.name = name;
        this.age = age;
        this.ID = nextId++;
    }
    public void show()
    {
        System.out.println("Id=" + ID + "\nName=" + name
                           + "\nAge=" + age);
    }
    public void showNextId()
    {
        System.out.println("Next employee id will be="
                           + nextId);
    }
    protected void finalize()
    {
        --nextId;
        // In this case,
        // gc will call finalize()
        // for 2 times for 2 objects.
```

```
    }
}

public class UseEmployee
 {
    public static void main(String[] args)
    {
        Employee E = new Employee("GFG1", 56);
        Employee F = new Employee("GFG2", 45);
        Employee G = new Employee("GFG3", 25);
        E.show();
        F.show();
        G.show();
        E.showNextId();
        F.showNextId();
        G.showNextId();

        {
            // It is sub block to keep
            // all those interns.
            Employee X = new Employee("GFG4", 23);
            Employee Y = new Employee("GFG5", 21);
            X.show();
            Y.show();
            X.showNextId();
            Y.showNextId();
            X = Y = null;
            System.gc();
            System.runFinalization();
        }
        E.showNextId();
    }
}
```

INPUT DATA:

OUTPUT DATA

EXPERIMENT: 7

AIM: Object Oriented Programming Concepts: Problem on the use of Polymorphism

Write a JAVA program to represent the concept of polymorphism.

PROGRAM:

```
import java.util.*;
public class ExceptionDemo
{
    static void func(int a,int b) throws ArithmeticException,
    ArrayIndexOutOfBoundsException
    {
    System.out.println(10/a);
     int[] arr={1,2,3};
    System.out.println(arr[b]);
    }
public static void main (String[] args)
{
            Scanner in=new Scanner(System.in);
            for(int i=0;i<3;i++)
            {
            Try
            {
                func(in.nextInt(),in.nextInt());
            }
            catch(ArithmeticException e)
            {
               System.out.println("can't divide by zero");
            }
                catch(ArrayIndexOutOfBoundsException e)
            {
```

```
                System.out.println("Out of bounds!");
            }
            }
        }
    }
```

INPUT DATA:

OUTPUT DATA:

EXPERIMENT: 8

AIM: Programs involving: Exception handling

Write a Java program to create multiple Exceptions.

PROGRAM:

```
import java.util.*;
public class ExceptionDemo
{
        static void func(int a,int b) throws ArithmeticException,
        ArrayIndexOutOfBoundsException
        {
                System.out.println(10/a);
                int[] arr={1,2,3};
                System.out.println(arr[b]);
        }
        public static void main (String[] args)
 {
                Scanner in=new Scanner(System.in);
                for(int i=0;i<3;i++){
                try
                {
                    func(in.nextInt(),in.nextInt());
                }
                catch(ArithmeticException e)
                {
                  System.out.println("can't divide by zero");
                }
                catch(ArrayIndexOutOfBoundsException e)
                {
                  System.out.println("Out of bounds!");
```

```
                }
                }
        }
}
```

INPUT DATA:

OUTPUT DATA:

EXPERIMENT: 9

AIM: Programs involving: Threads and Multiple threads

Write a Java program to create multiple threads in Java. Explain all thread methods with examples.

PROGRAM:

```
class ThreadTest extends Thread
{
    private Thread thread;
    private String threadName;

    ThreadTest( String msg)
    {
        threadName = msg;
        System.out.println("Creating thread: " +  threadName );
    }
    public void run()
    {
        System.out.println("Running thread: " +  threadName );
        try
        {
            for(int i = 0; i < 5; i++)
            {
                System.out.println("Thread: " + threadName + ", " + i);
                Thread.sleep(50);
            }
        }
        catch (InterruptedException e)
        {
            System.out.println("Exception in thread: " +  threadName);
        }
        System.out.println("Thread " +  threadName + " continue...");
    }
    public void start ()
    {
        System.out.println("Start method " +  threadName );
        if (thread == null)
```

```
            {
                thread = new Thread (this, threadName);
                thread.start ();
            }
        }
}
public class MultipleThread
{
    public static void main(String args[])
    {
        ThreadTest thread1 = new ThreadTest( "First Thread");
        thread1.start();

        ThreadTest thread2 = new ThreadTest( "Second Thread");
        thread2.start();
    }
}
```

INPUT DATA:

OUTPUT DATA:

Experiment 10

AIM: Programs involving: Packages in Java

Program 1

```
import java.io.File;
import java.io.IOException;

import java.util.Scanner;
class Progpackage
{
	public static void main(String[] args)
{
		try
		{
			File r=new
			File("C:\\Users\\LENOVO\\Desktop\\DSU\\DSU
			data\\scanner.txt");
			Scanner sc=new Scanner(r);
			while(sc.hasNextLine())
			{
				System.out.println("this is my first \" \"program");
				System.out.println(sc.nextLine());
			}
		}
		catch(IOException e)
		{
			System.out.println(e);
		}
	}
}
```

INPUT DATA:

OUTPUT DATA:

Program 2

```
package OODJ;

class A
{
	void show()
	{
		System.out.println("Java Programming");
	}
}

class proguserdefinepackage
{
	public static void main(String[] args)
	{
		A r=new A();
		r.show();
	}
}
```

INPUT DATA:

OUTPUT DATA: