**1st program: Basic programs using data types, operators, and control statements**

**in Java.**

**Code:** 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);

}

}

**Output:**

**2nd program:** **Basic programs using Arrays**

**Code:**

import java.util.\*;

public class Exercise26

{

public static void main(String[] args) throws Exception

{

int[] array\_nums = {1,6,0,3,8,9,0,2};

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");

}

}

Output:1,6,3,8,9,0,0,0

**3rd program: Write a Java program to find the first non-repeating character in a string**

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;

}

}

}

}

Output:

The given string is: gibblegabbler

The first non-repeating character is: i

**4th program:** **Oriented Programming Concepts: Problem on the use of**

**constructors, inheritance**

**Code:**

import java.util.Scanner;

class Employee

{

String emp\_id,emp\_name;

Employee(String emp\_id, String emp\_name)

{

this.emp\_id = emp\_id;

this.emp\_name = emp\_name;

}

}

class Salary extends Employee

{

String designation;

double monthly\_salary;

Salary(String emp\_id, String emp\_name, String designation, double monthly\_salary)

{

super(emp\_id, emp\_name);

this.designation = designation;

this.monthly\_salary = monthly\_salary;

}

void displayDetails()

{

System.out.format("%-10s %-15s %-20s %-15s %n", emp\_id, emp\_name, designation, monthly\_salary);

}

public static void main(String[] args)

{

Scanner scanner = new Scanner(System.in);

System.out.print("Enter the number of employees: ");

int numberOfEmployees = scanner.nextInt();

Salary[] employees = new Salary[numberOfEmployees];

for (int i = 0; i < numberOfEmployees; i++) {

System.out.println("\nEnter details for Employee " + (i + 1));

System.out.print("Employee ID: ");

String emp\_id = scanner.next();

System.out.print("Employee Name: ");

String emp\_name = scanner.next();

System.out.print("Designation: ");

String designation = scanner.next();

System.out.print("Monthly Salary: ");

double monthly\_salary = scanner.nextDouble();

employees[i] = new Salary(emp\_id, emp\_name, designation, monthly\_salary);

}

System.out.format("%-10s %-15s %-20s %-15s %n", "Emp ID", "Emp Name", "Designation", "Monthly Salary");

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

for (Salary employee : employees) {

employee.displayDetails();

}

}

}

**Output:**

Enter 2 employee entry

Enter employee name: shankar

Enter employee designation: trader

Enter employee salary: 100000

Enter employee id: 01

Enter employee name: manjunath

Enter employee designation: driver

Enter employee salary: 100000

Enter employee id: 02

Employee name: shanakar

Employee designation:trader

**5th program: Write a JAVA program to represent Method Overloading and Overriding.**

**Code:**

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();

}

}

Output:

Calling add method with 2 input parameters:42

Calling add method with 3 input parameters:117

Calling add method with 2 double parameters:79.3

Calling add method with 1 Int and 1 double parameter:35.5

This is a parent class method

This is child class method

**6th program: Write a JAVA program to represent Garbage Collection**

**Code:**

class Employee

{

private int ID;

private String name;

private int age;

private static int nextId = 1;

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;

}

}

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();

{

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();

}

}

Output:

Id=1

Name=GFG1

Age=56

Id=2

Name=GFG2

Age=45

Id=3

Name=GFG3

Age=25

Id=4

Name=GFG4

Age=23

Id=5

Name=GFG5

Age=21

**7th program: Write a JAVA program to represent the concept of polymorphism.**

**Code:**

class Animal {

public void makeSound() {

System.out.println("Some generic sound");

}

}

class Dog extends Animal {

@Override

public void makeSound() {

System.out.println("Bark! Bark!");

}

public void fetch() {

System.out.println("Fetching the ball");

}

}

class Cat extends Animal {

@Override

public void makeSound() {

System.out.println("Meow! Meow!");

}

public void scratch() {

System.out.println("Scratching furniture");

}

}

public class Polymorphism {

public static void main(String[] args) {

Animal dog = new Dog();

Animal cat = new Cat();

dog.makeSound();

cat.makeSound();

((Dog) dog).fetch();

((Cat) cat).scratch();

}

}

Output: Bark! Bark!

Meow! Meow!

Fetching the ball

Scratching the Furniture

**8th program: Exception handling**

**Code:**

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!");

}

}

}

}

Output:

Error: Division by zero is not allowed

**9th program: Write a Java program to create multiple threads in Java. Explain all thread**

**methods with examples.**

**Code:**

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();

}

}

Output: Output:

Creating thread: First Thread

Creating thread: Second Thread

Start Method: First Method

Start Method: Second Method

Running Thread: First Thread

Thread: First Thread,0

Thread first Thread continue

Running thread: Second Thread

Thread: Second Thread

Thread first thread continues

**10th program: 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);

}

}

}

Output:

**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();

}

}

Output:

Java Programming

**11th program:**

interface Customer

{

int amt=5; //public+static+final

void purchase(); //public+abstract

}

class seller implements Customer

{

@Override

public void purchase()

{

System.out.println("final amount"+""+amt);

}

}

class Intefacevariable

{

public static void main(String[] args)

{

Customer c=new seller();

c.purchase();

System.out.println(Customer.amt);

}

}

Output:

Final amount 5

**Program 2:**

interface A

{

void add();

}

interface B extends A

{

void sub();

}

class java implements B

{

//@override

public void add()

{

int a=10, b=20,c;

c=a+b;

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

}

//@override

public void sub()

{

int a=10, b=20,c;

c=a-b;

System.out.println("Subtraction"+c);

}

class main

{

public static void main(String[] args) {

B r=new java();

r.add();

r.sub();

}

}

}

Output:

Addition 30

Subtraction 10