R.M.K. ENGINEERING COLLEGE

R.S.M. Nagar, Kavaraipettai - 601 206.

CS8383

Object Oriented Programming Laboratory

NAME : NAVEEN KUMAR S

REG.NO : 111719107023

Branch : EIE

Semester : III

CS8383 - OBJECT ORIENTED PROGRAMMING Laboratory

Exp. No:	Date	List of Exercises	Pg. No:
1.	26.08.20	Electricity Bill Generation	1
2.	01.09.20	Currency Conversion, Distance conversion & time conversion.	4
3.	01.09.20	Pay Slip Generation	11
4.	12.09.20	Stack ADT	14
5.	12.09.20	String Operation using ArrayList	16
6.	01.10.20	Abstract class	19
7.	05.10.20	User defined Exceptions	21
8.	05.10.20	File OPerations	23
9.	06.10.20	Multi-threading	25
10.	09.10.20	Generic Method	27
11.	10.10.20	Calculator for Decimal Manipulation and Scientific Manipulation	29

Exp.no: 1

Date: 26.08.20

Title: Electricity Bill

Aim: To develop java application for calculating Electricity bill.

Algorithm:

1. Start
2. Declare the class with the following members:

customer_name, customer_num, previous month reading (pmr), current month reading (cmr), type_of_eb_connection, units_consumed, billamount.

3. Define method to input details.
4. Get customer_name, customer_num, previous month reading (pmr), current month reading (cmr), type_of_eb connection all the inputs from user.
5. Define another method to calculate units and bill amount.

units_consumed = cmr - pmr

5.1 if the type of connection is domestic then calculate

a) if (units >= 100), billamount = (units * 1)

b) else if (units >= 101 && units <= 200), billamount = ((100*1) + (units-100)*2.5

c) else if (units >= 201 && units <= 500), billamount = (100*1) + (100*2.5) + (units - 500)*(4))

d) else if (units >500), billamount = billamount = ((100 * 1) + (100 * 2.5) + 3100 * 4) + (units - 500) * (6))

S-2 : if the type of connection is domestic then calculate,

a) if (units <= 100), billamount = (units * 2).

b) else if (units >= 101) && units <= 200), billamount = ((100 * 2) + (units - 100) * (4.5)).

c) else if (units >= 201 && units <= 500), billamount = ((100 * 2) + (100 * 4.5) + (units - 200) * (6))

d) else if (units >500), billamount = ((100 * 2) + (100 * 4.5) + (300 * 6) + (units - 500) * (7))

6. Define the method outputdetail to display the customer name, consumer number, unit consumed and bill amount details.

7. Create a new class and add main method in it.

8. Create an object of previous class

9. By using the object, call the input details method, calculate billamount method and output details method.

10. End.

If the type of the EB connection is commercial, then 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
> 500 units – Rs. 7 per unit

Example Input/Output 1:
Input:
20 Ram 1245 1350 domestic

Output:
112.50

Show My Solution

```
import java.util.*;
class ElectricityBill{
    int prevMonth;
    int currMonth;
    int units;
    String calBill;
    float billAmount;
    public void setConsumerNumber(int consNo) {
        int ConsumerNumber = consNo;

    }
    public void setConsumerName(String consName){
        String ConsumerName=consName;
    }
    public int setPreviousMonthReading(int readPrev){
        prevMonth=readPrev;
        return prevMonth;
    }
    public int setCurrentMonthReading(int readCurr){
        currMonth=readCurr;
        return currMonth;
    }
    public String setIsDomestic(boolean bool){
        if(bool==true){
            calBill="isDom";
        }
        else if(bool==false){
            calBill="isCur";
        }
        return calBill;
    }
    public float calculateBillAmount(){
        if("isDom"==calBill){
            units=currMonth-prevMonth;
            if(units<100){
                billAmount=units*1.00f;
            }
            else if(units<=200){
                billAmount=(100f*1.00f)+(units-100f)*2.50f;
            }
            else if(units<=500){
                billAmount=(100f*1.00f)+(100f*2.50f)+(units-200f)*4.00f;
            }
            else if(units>500){
                billAmount=(100f*1.00f)+(100f*2.50f)+(300f*4f)+(units-500f)*6.00f;
            }
        }
        else if("isCur"==calBill){
            units=currMonth-prevMonth;
            if(units<100){
                billAmount=units*2.00f;
            }
            else if(units<=200){
                billAmount=(100f*2.00f)+(units-100f)*4.50f;
            }
            else if(units<=500){
                billAmount=(100f*2.00f)+(100f*4.50f)+(units-200f)*6.00f;
            }
            else if(units>500){
                billAmount=(100f*2.00f)+(100f*4.50f)+(300f*6f)+(units-500f)*7.00f;
            }
        }
    return billAmount;

    }
}

public class Hello {

    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        ElectricityBill eBill = new ElectricityBill();
        eBill.setConsumerNumber(sc.nextInt());
        eBill.setConsumerName(sc.next());
        eBill.setPreviousMonthReading(sc.nextInt());
        eBill.setCurrentMonthReading(sc.nextInt());
        eBill.setIsDomestic(sc.next().equals("domestic"));
        System.out.printf("%.2f", eBill.calculateBillAmount());
    }
}
```

</> EX001 - Electricity Bill Amount (Id-8867)

Develop a Java application to generate Electricity bill. Create a class ElectricityBill with the following members: consumerNumber, consumerName, previousMonthReading, currentMonthReading, isDomestic (type of EB connection i.e domestic or commercial). Create a method calculateBillAmount to compute and return the bill amount using the following tariff.

If the type of the EB connection is domestic, then 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
> 500 units – Rs. 6 per unit

If the type of the EB connection is commercial, then 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
> 500 units – Rs. 7 per unit

Example Input/Output 1:
Input:
20 Ram 1245 1350 domestic

Output:
112.50

Show My Solution

Result:

the java application to generate electricity bill is executed successfully and output is verified.

Exp.No: 2(a)

Date: 01.09.20

Title: Currency Convertor

Aim: To develop a java application to implement currency convertor using packages.

Algorithm:

1. Start
2. Declare a package for currency
3. Declare a class "currencyconvert" with in the currency package.
4. Define the following methods.
   - 4.1. Read value to convert dollor to inr calculate rate = value * (66.66)
   - 4.2 Read value to convert euro to inr calculate rate = value * (79.84)
   - 4.3 Read value to convert yen to inr calculate rate = value * (0.61)
   - 4.4 Read value to convert inr to dollar calculate rate = value * (0.015)
   - 4.5 Read value to convert inr to euro calculate rate = value * (0.013)
   - 4.6 Read value to convert inr to yen calculate rate = value * (1.645)
5. Create another class and add main method & import currency package.
6. Create an object for currencyconvert class.
7. Using switch case, call various method from currency convert class
8. End.

Result:

the java application to implement currency convertor using package is executed successfully and output verified.

Exp.NO: 2(b)

Date: 01.09.20

Title: Time convertor

Aim: To develop a java application to implement time convertor using packages.

Algorithm:

1. Start
2. Declare a package for time convertor package.
3. Declare a class within the time convertor package.

4.1. Define the method "hours_to_minutes", to convert hours into minutes then calculate = (no. of hours * 60) minutes.

4.2 Define the method "hours_to_seconds", to convert hours into seconds then calculate = (no. of hours * 360) seconds.

4.3 Define the method "minutes_to_hours", to convert minutes to hours then calculate = (no. of minutes/60) hours + (no. of minutes % 360)

5. Create a class and add main method & import time convertor package.

6. Create an object for previous class.

7.1) if user choose conversion of hours to minutes then call method "hours_to_minutes"

7.2) if user choose conversion of hours to seconds then call method "hours_to_seconds".

7.3: if user choose conversion of seconds to hours then call method "seconds_to_hours".

7.4) if user choose conversion of minutes to hours then call method "minuts_to_hours"

8) calculate the time

9. Display the result

10. End.

Result:

the java application to implement time converter using pakage is executed successfully and output is verified.

Exp.No: 2(c)

Date: 01.09.20

Title: Distance Convertor

Aim: To develop a java application to implement distance convertor using packages.

Algorithm:

1. Start
2. Declare a package for distance convertor
3. Declare a class with in the distance convertor package.

4.1: Define the method "meters_to_km", to convert meters to km then calculate meter = (meter / 100) km.

4.2 Define the method "miles_to_km", to convert miles to kmm then calculate miles = (miles * 1.6) km.

4.3 Define the method "km_to_meters", to convert km to meters then calculate km = (km / 1.6) miles.

4.5 Create a class and add main method.

6. Create an object for previous class.

7.1. if user choose meter to km then call method "meters_to_km".

7.2. if user choose miles to km then call method "miles_to_km".

7.3. If user choose km to meters then call method "km_to_meters".

7.4. If user choose km to miles then call method "km_to_miles".

8. Calculate the distance

9. Display the result

10. End.

```
import java.util.*;
import com.skillrack.converter.*;

public class Hello {

    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        CurrencyConverter currencyConverter = new CurrencyConverter();
        DistanceConverter distanceConverter = new DistanceConverter();
        TimeConverter timeConverter = new TimeConverter();
        int T = sc.nextInt();
        while(T-- > 0){
            int option = sc.nextInt();
            double fromVal = sc.nextDouble();
            double toVal = 0;
            switch(option){
                case 1:
                    toVal = currencyConverter.dollarToRupee(fromVal);
                    break;
                case 2:
                    toVal = currencyConverter.rupeeToDollar(fromVal);
                    break;
                case 3:
                    toVal = currencyConverter.euroToRupee(fromVal);
                    break;
                case 4:
                    toVal = currencyConverter.rupeeToEuro(fromVal);
                    break;
                case 5:
                    toVal = currencyConverter.yenToRupee(fromVal);
                    break;
                case 6:
                    toVal = currencyConverter.rupeeToYen(fromVal);
                    break;
                case 7:
                    toVal = distanceConverter.meterToKilometer(fromVal);
                    break;
                case 8:
                    toVal = distanceConverter.kilometerToMeter(fromVal);
                    break;
                case 9:
                    toVal = distanceConverter.mileToKilometer(fromVal);
                    break;
                case 10:
                    toVal = distanceConverter.kilometerToMile(fromVal);
                    break;
                case 11:
                    toVal = timeConverter.hourToMinute(fromVal);
                    break;
                case 12:
                    toVal = timeConverter.hourToSecond(fromVal);
                    break;
                case 13:
                    toVal = timeConverter.secondToHour(fromVal);
                    break;
                case 14:
                    toVal = timeConverter.minuteToHour(fromVal);
                    break;
            }
            System.out.println(toVal);
        }

    }
}

package com.skillrack.converter;
import java.util.*;
public class CurrencyConverter
{
    public  double dollarToRupee(double fromVal)
    {
        return fromVal*69.00;
    }
    public  double rupeeToDollar(double fromVal)
    {
        return fromVal/69.00;
    }
    public  double euroToRupee(double fromVal)
    {
        return fromVal*78.00;
    }
    public double rupeeToEuro(double fromVal)
    {
        return fromVal/78.00;
    }
    public  double yenToRupee(double fromVal)
    {
        return fromVal*0.64;
    }
    public  double rupeeToYen(double fromVal)
    {
        return fromVal/0.64;
    }
}
```

```java
package com.skillrack.converter;
import java.util.*;
public class DistanceConverter
{
    public double meterToKilometer(double meters)
    {
        return meters/1000;
    }
    public double kilometerToMeter(double kilometer)
    {
        return kilometer*1000;
    }
    public double mileToKilometer(double mile)
    {
        return mile*1.609;
    }
    public double kilometerToMile(double kilometer)
    {
        return kilometer/1.609;
    }

}
```

```java
package com.skillrack.converter;
import java.util.*;
public class TimeConverter
{
    public double hourToMinute(double hours)
    {
        return hours*60;

    }
    public double hourToSecond(double hour)
    {
        return hour*3600;

    }
    public double secondToHour(double seconds)
    {
        return seconds/3600;
    }
    public double minuteToHour(double minutes)
    {
        return minutes/60;
    }
}
```

Completed Programs - Lab - OOP - Java - CS8383 - Converter - PART 002

</> EX002 Conversion (Id-8872)

Develop a Java application to implement currency converter (Dollar to INR, Euro to INR, Yen to INR and vice
KM, miles to KM and vice versa), time converter (hours to minutes, seconds and vice versa) using packages

Create the following classes with appropriate methods:
com.skillrack.converter.CurrencyConverter
com.skillrack.converter.DistanceConverter
com.skillrack.converter.TimeConverter

Note: The files will be saved in the appropriate directory.

The input is given based on the following command to convert the values.

1. Dollar to rupee
2. Rupee to dollar
3. Euro to rupee
4. Rupee to euro
5. Yen to rupee
6. Rupee to yen
7. Meter to kilometer
8. Kilometer to meter
9. Mile to kilometer
10. Kilometer to mile
11. Hour to minute
12. Hour to second
13. Second to hour
14. Minute to hour

All the methods must return the converted values.

Assume the following conversion values,
1 dollar is equal to 69 rupees.
1 euro is equal to 78 rupees.
1 yen is equal to 0.64 rupees.
1 mile is equal to 1.609

Example Input/Output 1:
Input:
5
1 20
5 15
7 2
11 4
14 126

Output:
1380.0
9.6
0.002
240.0
2.1

The main method class is given below

Result:

the java application to implement distance converter using package is executed successfully and output is verified.

Exp.No: 3

Date: 01.09.20

Title: Employee Details (Pay Slip Generation)

Aim: To develop a java application with Employee class with Emp_name, Emp_id, Address, Mail_id, Mobile_no as members. Inherit the classes, programmer, Assistant professor, Associate professor and professor from employee class. Add basic pay as the member of all the inherited classes with 97% of BP as DA, 10% of BP as HRA, 12% of BP as PF, 0.1% of BP for staff club fund, Generate pay slips for the employees with their gross and net salary.

Algorithm:

1. Start
2. Create a class employee with employee name, employee id, address, mail id, mobile num as members with in this employee class
3. Define constructor for initializing the variables
4. Define method payslip with basicpay of double data type as parameter with in this method.

5.1 Find dearness allowance (da), calculate da = basicpay * 97/100.

5.2 find house rent allowance (hra), calculate hra = basic pay * 10/100

5.3 find provident fund (pf), calculate pf = basic pay * 12/100

5.4. find staff club fund, calculate club = basic pay * 0.1/100.

5.5. find gross salary, calculate gross_salary = basic pay + da + hra.

5.6. find net salary, calculate net_salary = (gross_salary) - (pf) - (club)

6.1. Create a class programmer and inherit the class from employee class add basic pay as the member of this inherited class.

6.2. Create a class assistant proffessor and inherit the class from employee class add basic pay as the member of this inherited class.

6.3. Create a class associate proffesor and inherit the class from employee class add basic pay as the member of this inherited class.

7. declare a class employeedemo add main method in it.

8.1. create objects for all inherited classes and pass values as parameters.

8.2. Call the method payslip with the objects of inherited classes.

89. Calculate and display the result.

10. End.

```
import java.util.*;
class Employee{
    String Name,Id,Address,Mail,Mobile;
    double BP,DA,HRA,PF,StaffClubFund,GrossSalary,NetSalary;
    public void getEmployeeDetails(Scanner input){
        Name = input.nextLine();
        Id = input.nextLine();
        Address = input.nextLine();
        Mail = input.nextLine();
        Mobile = input.nextLine();
    }
    public void getBasicPay(Scanner payment){
        BP = Double.parseDouble(payment.nextLine());
        DA = (BP * 97) / 100;
        HRA = (BP * 10) / 100;
        PF = (BP * 12) / 100;
        StaffClubFund = (BP * 0.1) / 100;
        GrossSalary = (BP + DA + HRA);
        NetSalary = (GrossSalary - PF - StaffClubFund);
    }
    public void employeeDetails(){
        System.out.println("EMPLOYEE DETAILS");
        System.out.println("Name: " +Name);
        System.out.println("Id: " +Id);
        System.out.println("Address: " +Address);
        System.out.println("Mail: " +Mail);
        System.out.println("Mobile: " +Mobile);

    }
    public void employeePaySlip(){
        System.out.printf("BP: Rs. "+"%.1f",BP);
        System.out.printf("\nDA: Rs. "+"%.1f",DA);
        System.out.printf("\nHRA: Rs. "+"%.1f",HRA);
        System.out.printf("\nPF: Rs. "+"%.1f",PF);
        System.out.printf("\nStaff Club Fund: Rs. "+"%.1f",StaffClubFund);
        System.out.printf("\nGross Salary: Rs. "+"%.1f",GrossSalary);
        System.out.printf("\nNet Salary: Rs. "+"%.1f\n",NetSalary);
    }
}
class Programmer extends Employee{
    public void printPaySlip(){
        super.employeeDetails();
        System.out.println("PAY SLIP FOR PROGRAMMER");
        super.employeePaySlip();
    }
}
class AssistantProfessor extends Employee{
    public void printPaySlip(){
        super.employeeDetails();
        System.out.println("PAY SLIP FOR ASSISTANT PROFESSOR");
        super.employeePaySlip();
    }
}
class AssociateProfessor extends Employee{
    public void printPaySlip(){
        super.employeeDetails();
        System.out.println("PAY SLIP FOR ASSOCIATE PROFESSOR");
        super.employeePaySlip();
    }
}
class Professor extends Employee{
    public void printPaySlip(){
        super.employeeDetails();
        System.out.println("PAY SLIP FOR PROFESSOR");
        super.employeePaySlip();
    }
}


public class Hello {

    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int N = Integer.parseInt(sc.nextLine());
        for (int ctr = 1; ctr <= N; ctr++) {
            int employeeType = Integer.parseInt(sc.nextLine());
            switch (employeeType) {
                case 1:
                    Programmer programmer = new Programmer();
                    if (!(programmer instanceof Employee)) {
                        System.out.println("Employee Not Inherited");
                    }
                    programmer.getEmployeeDetails(sc);
                    programmer.getBasicPay(sc);
                    programmer.printPaySlip();
                    break;

                case 2:
                    AssistantProfessor assistantProfessor = new AssistantProfessor();
                    if (!(assistantProfessor instanceof Employee)) {
                        System.out.println("Employee Not Inherited");
                    }
                    assistantProfessor.getEmployeeDetails(sc);
                    assistantProfessor.getBasicPay(sc);
                    assistantProfessor.printPaySlip();
                    break;
                case 3:
```

```
        System.out.printf("\nStaff Club Fund: Rs. "+"%.1f",StaffClubFund);
        System.out.printf("\nGross Salary: Rs. "+"%.1f",GrossSalary);
        System.out.printf("\nNet Salary: Rs. "+"%.1f\n",NetSalary);
    }
}
class Programmer extends Employee{
    public void printPaySlip(){
        super.employeeDetails();
        System.out.println("PAY SLIP FOR PROGRAMMER");
        super.employeePaySlip();
    }
}
class AssistantProfessor extends Employee{
    public void printPaySlip(){
        super.employeeDetails();
        System.out.println("PAY SLIP FOR ASSISTANT PROFESSOR");
        super.employeePaySlip();
    }
}
class AssociateProfessor extends Employee{
    public void printPaySlip(){
        super.employeeDetails();
        System.out.println("PAY SLIP FOR ASSOCIATE PROFESSOR");
        super.employeePaySlip();
    }
}
class Professor extends Employee{
    public void printPaySlip(){
        super.employeeDetails();
        System.out.println("PAY SLIP FOR PROFESSOR");
        super.employeePaySlip();
    }
}


public class Hello {

    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int N = Integer.parseInt(sc.nextLine());
        for (int ctr = 1; ctr <= N; ctr++) {
            int employeeType = Integer.parseInt(sc.nextLine());
            switch (employeeType) {
                case 1:
                    Programmer programmer = new Programmer();
                    if (!(programmer instanceof Employee)) {
                        System.out.println("Employee Not Inherited");
                    }
                    programmer.getEmployeeDetails(sc);
                    programmer.getBasicPay(sc);
                    programmer.printPaySlip();
                    break;

                case 2:
                    AssistantProfessor assistantProfessor = new AssistantProfessor();
                    if (!(assistantProfessor instanceof Employee)) {
                        System.out.println("Employee Not Inherited");
                    }
                    assistantProfessor.getEmployeeDetails(sc);
                    assistantProfessor.getBasicPay(sc);
                    assistantProfessor.printPaySlip();
                    break;
                case 3:
                    AssociateProfessor associateProfessor = new AssociateProfessor();
                    if (!(associateProfessor instanceof Employee)) {
                        System.out.println("Employee Not Inherited");
                    }
                    associateProfessor.getEmployeeDetails(sc);
                    associateProfessor.getBasicPay(sc);
                    associateProfessor.printPaySlip();
                    break;
                case 4:
                    Professor professor = new Professor();
                    if (!(professor instanceof Employee)) {
                        System.out.println("Employee Not Inherited");
                    }
                    professor.getEmployeeDetails(sc);
                    professor.getBasicPay(sc);
                    professor.printPaySlip();
                    break;
            }
        }
    }
}
```

Completed Programs - Lab - OOP - Java - CS8383 - Payroll - PART 003

</> EX003 Generate Pay Slip using Inheritance (Id-8868)

Develop a Java application with **Employee** class with name, id, address, mail, mobile as members. Inherit the classes, **Programmer**, **AssistantProfessor**, **AssociateProfessor** and **Professor** from Employee class. Add Basic Pay (**BP**) as the member of all the inherited classes with 97% of BP as **DA**, 10 % of BP as **HRA**, 12% of BP as **PF**, 0.1% of BP for **staffClubFund**. Generate pay slips for the employees with their gross and net salary.
Note: All the values in rupees must be printed with the precision up to 1 decimal place.

Formulae:
Gross Salary = BP + DA + HRA
Net Salary = Gross Salary - PF - Staff Club Fund

Boundary Condition(s):
1 <= N <= 10
1 <= Employee Type <= 4
1 <= Length of employee name <= 100
1 <= Employee id <= 10^8
1 <= Employee address <= 100
1 <= Employee mail id <= 50
10^5 <= Employee mobile number < 10^12
1 <= Employee basic pay <= 10^7

Input Format:
The first line contains the value of N (where N is number of employees).
The next N set of 7 lines contain the employee details.
The first line in each set contains the employee type. (1 for Programmer, 2 for Assistant Professor, 3 for Associate Professor and 4 for Professor).
The second line in each set contains the employee name.
The third line in each set contains the employee id.
The fourth line in each set contains the employee address.
The fifth line in each set contains the employee mail id.
The sixth line in each set contains the employee mobile number.
The seventh line in each set contains the employee basic pay.

Output Format:
The first N set of 14 lines contain the details of employee and the details of his/her pay slip as shown in the Example Input/Output section.

Example Input/Output 1:
Input:
1
John
10
51/A, XYZ street, ABC city - 654 879.
john@gmail.com
9873216540
150000
2
Bruce
55
No. 6, 2nd Floor, PQR Apartments, GGG city- 987 654.
bruce47@gmail.com
6549807321
20000

Output:
EMPLOYEE DETAILS
Name: John
Id: 10
Address: 51/A, XYZ street, ABC city - 654 879.
Mail: john@gmail.com
Mobile: 9873216540
PAY SLIP FOR PROGRAMMER
BP: Rs. 150000.0
DA: Rs. 145500.0
HRA: Rs. 15000.0
PF: Rs. 18000.0
Staff Club Fund: Rs. 150.0
Gross Salary: Rs. 310500.0
Net Salary: Rs. 292350.0
EMPLOYEE DETAILS
Name: Bruce
Id: 55
Address: No. 6, 2nd Floor, PQR Apartments, GGG city- 987 654.
Mail: bruce47@gmail.com
Mobile: 6549807321
PAY SLIP FOR ASSISTANT PROFESSOR
BP: Rs. 20000.0
DA: Rs. 19400.0
HRA: Rs. 2000.0
PF: Rs. 2400.0
Staff Club Fund: Rs. 20.0
Gross Salary: Rs. 41400.0
Net Salary: Rs. 38980.0

Show My Solution

Result:

Thus the java application to generate payslips for the employee is executed successfully.

Exp.no: 4

Date: 12.09.20

Title: Stack Implementation

Aim: To design a java interface for ADT stack. Implement the interface using array. provide necessary exception handling in both the implementations.

Algorithm:

1. Start
2. Define an interface stack with two method push and pop
3. initialize stack with maximum size N, top = -1, N = 20
4. Define a class that implement an interface stack.
5. Define a method push.
   5.1. If (top+1 >= size) throw an overflow exception.
   5.2. else add an element to stack.
6. Define a method pop
   6.1. if (top == -1) then throw underflow exception.
   6.2. else pop element from stack.
7. Define a method to display the content of stack.
8. Define a main class and include try block.
9. Within try block call the push, pop, display methods.
10. Include the necessary catch block.
11. Stop.

Show My Solution

```
import java.util.*;
import java.lang.*;
interface Stack {
    void push(int data) throws Exception;
    int pop()throws Exception;
    int peek()throws Exception;
    boolean isEmpty();
    void display()throws Exception;

}
class ArrayStack implements Stack {
    int top = -1;
    int [] arr;
    public ArrayStack(int SIZE) {
        arr = new int[SIZE];
        return;
    }
    public void push(int element) throws Exception {
        if(top == arr.length - 1) {
            throw new Exception();
        }
        else {
            arr[++top] = element;
        }
    }
    public int peek()throws Exception {
        if(top == -1){
            throw new Exception();
        }
        else {
            return arr[top];
        }
    }
    public int pop()throws Exception {
        if(top == -1) {
            throw new Exception("Stack Empty");

        }
        else {
            int popper = arr[top];
            top--;
            return popper;
        }
    }
    public boolean isEmpty() {
        return top == -1;
    }
    public void display()throws Exception{
        if(top == -1) {
            throw new Exception();
        }
        else {
            for(int i=0;i<=top;i++) {
                System.out.print(arr[i]+" ");
            }
        }
    }
}
public class Hello {

    static final int SIZE = 100;
```



```
    public void display()throws Exception{
        if(top == -1) {
            throw new Exception();
        }
        else {
            for(int i=0;i<=top;i++) {
                System.out.print(arr[i]+" ");
            }
        }
    }
}
public class Hello {

    static final int SIZE = 100;

    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        Stack stack = new ArrayStack(SIZE);
        int N = sc.nextInt();
        for (int query = 1; query <= N; query++) {
            int queryType = sc.nextInt();
            switch (queryType) {
                case 1:
                    try {
                        stack.push(sc.nextInt());
                    } catch (Exception e) {
                        System.out.println("Stack Overflow");
                    }
                    break;
                case 2:
                    try {
                        System.out.println("Popped Element: " + stack.pop());
                    } catch (Exception e) {
                        System.out.println("Stack Underflow");
                    }
                    break;
                case 3:
                    try {
                        System.out.println("Top Element: " + stack.peek());
                    } catch (Exception e) {
                        System.out.println("Stack Empty");
                    }
                    break;
                case 4:
                    try {
                        System.out.print("Stack Elements: ");
                        stack.display();
                        System.out.println();
                    } catch (Exception e) {
                        System.out.println("Stack Empty");
                    }
                    break;
                case 5:
                    if (stack.isEmpty()) {
                        System.out.println("TRUE");
                    } else {
                        System.out.println("FALSE");
                    }
            }
        }
    }
}
```



Webinar /Other Programming Track(s)

College/University LeaderBoard | Global LeaderBoard

Completed Programs - Lab - OOP - Java - CS8383 - Stack ADT - PART 004

</> EX004 Stack ADT Using Interface (Id-8871)

Design a Java interface for ADT **Stack**. Implement this interface using an array. Provide necessary exception handling in both the implementations. The interface Stack must have the following method signatures.

- **void push(int data)**
- **int pop()**
- **int peek()**
- **boolean isEmpty()**
- **void display()**

Note: The display() method must print all the elements of the stack separated by a space in the order of insertion if the stack is not empty. Else it throws necessary exception.

The query type can be any one of the following types.
1 - Push
2 - Pop
3 - Peek
4 - Display
5 - isEmpty

Example Input/Output 1:
Input:
19
1 10
1 20
1 30
1 40
4
2
4
2
3
4
1 50
4
5
2
2
2
2
3
5

Output:
Stack Elements: 10 20 30 40
Popped Element: 40
Stack Elements: 10 20 30
Popped Element: 30
Top Element: 20
Stack Elements: 10 20
Stack Elements: 10 20 50
FALSE
Popped Element: 50
Popped Element: 20
Popped Element: 10
Stack Empty
Stack Empty
TRUE

Show My Solution

Result:

Thus, the java interface for ADT stack is created and stack is implemented using array with necessary exception handling. The program is executed successfully and output is verified.

Exp.No: 5

Date: 12.09.20

Title: String Operations Using Array list

Aim: Develop a program to perform string operations using Arraylist. Write functions for the following.

a. Append - add at end

b. Insert - add at particular index

c. Search.

d. List all string starts with given letter.

Algorithm:

1. Start
2. declare the variables. And create an object of Array List with string type.
3. define append() method
   3.1. read n value.
   3.2. Set for loop from 0 to n.
      3.2.1. enter the string you want to append.
      3.2.2. by using scanner class taking strings as input from user.
4. define insert() method.
   4.1. Get index value and element to be inserted in array list.
   4.2. by using add method, add the given element at particular position
5. Define search() method.
   5.1. enter string to be searched
   5.2. get input from user

5.3. if str.equals(al.get(i))

found=1

string found in the array list

then print the string found in index.

5.4 if found==0 then

String not found in the array list

6. Define display() method

6.1. enter character

6.2. char c = sc.next().charAt(0);

6.3. For(i=0; i<al.size(); i++)

String test = al.get(i)

if(test.charAt(0)==c) then, display string.

7. Stop.

```
import java.util.*;

public class Hello {

    public static void display(List<String> strList) {
        for (String S : strList) {
            System.out.print(S + " ");
        }
    }

    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int N = Integer.parseInt(sc.nextLine());
        List<String> strList = new ArrayList<>();
        for (int query = 1; query <= N; query++) {
            int queryType = sc.nextInt();
            String str = "";
            switch (queryType) {
case 1:
    strList.add(sc.nextLine().trim());
    break;
case 2:
    int index=sc.nextInt();
    strList.add(index,sc.nextLine().trim());
    break;
case 3:
    System.out.println(strList.indexOf(sc.nextLine().trim()));
    break;
case 4:
    int n=0;
    String startChar=sc.nextLine().trim();
    for (String element:strList)
    {
        if(element.startsWith(startChar))
        {
            System.out.print(element+" ");
            n=n+1;

        }
    }
    if(n==0)
    {
        System.out.println("No such string");

    }
    else{
        System.out.println();
    }
    break;
case 5:
    String element=sc.nextLine().trim();
    if (strList.contains(element))
    {
        strList.remove(strList.indexOf(element));

    }
    else{
        System.out.println("No such string");
    }
    break;
case 6:
    Collections.sort(strList);
    break;
case 7:
    System.out.println(strList.size());
    break;

                case 8:
                    if (!strList.isEmpty()) {
                        display(strList);
                        System.out.println();
                    } else {
                        System.out.println("List is empty");
                    }
                    break;
            }
        }
    }
}
```



Completed Programs - Lab - OOP - Java - CS8383 - String - ArrayList - PART 005

</> EX005 String - ArrayList (Id-8936)

The program must accept **N** queries as the input. The queries can be any one of the following types.
1 -> Query type followed by a string to perform the **Append** operation.
2 -> Query type followed by an integer and a string to perform the **Insert** operation.
3 -> Query type followed by a string to perform **Search** operation.
4 -> Query type followed by a character to perform **Starts with** operation.
5 -> Query type followed by a string to perform **Remove** operation.
6 -> Query type to perform **Sort** operation.
7 -> Query type to perform **Size** operation.
8 -> Query type to perform **Display** operation.

The string operations using ArrayList for the functions given below.
1. Append - Appends the given string to the end of this list.
2. Insert - Inserts the given string at the given index in this list. (0 <= index <= size of the ArrayList)
3. Search - Returns the index of the first occurrence of the given string in this list, or **-1** if this list does not contain the given string.
4. Starts with - Print all the string values starts with the given character. If there is no such string then print **No such string**.
5. Remove - Removes the first occurrence of the given string from this list, if it is present. If it is not present then print **No such string**.
6. Sort - Sort all the string values in the list in alphabetical order.
7. Size - Print the size of the list.
8. Display - If the list is not empty then print all the string values in the list. Else print **List is empty.**

You must fill in the missing lines of code so that runs successfully.

Note: Do not implement the Display operation, as it is implemented.

Example Input/Output 1:
Input:
17
1 cricket
1 hockey
8
7
2 0 football
8
7
6
8
1 chess
3 hockey
3 kabadi
4 c
4 a
5 cricket
5 rummy
8

Output:
cricket hockey
2
football cricket hockey
3
cricket football hockey
2
-1
cricket chess
No such string
No such string
football hockey chess

Show My Solution

Result:

Thus string operation is implemented using array List. The output is verified.

Exp.no: 6

Date: 01-10-20

Title: Abstract class

Aim: To develop a java program to create an abstract class named shape that contains two integers and an empty method named print area(). provide three classes named Rectangle, triangle and circle such that each one of the classes extends the class shape. Each one of the classes contains only the method print area() that prints the area of the given shape.

Algorithm:

1. Start
2. Create an abstract class named shape that contains 2 integers, constructors & create an empty abstract method named printarea().
3. Design the class named rectangle that extends the class shape and implement the print area() method.
   3.1. area = dim1 * dim2
   3.2. print the area of rectangle.
4. Design the class named circle that extends the class shape and implement the printarea() method.
   4.1 area = 3.14 * dim1 * dim1
   4.2 print the area of rectangle.
6. Design a main class and create an object of three classes rectangle, triangle and circle. through object call the printarea()
7. Stop.

```
    }
}
```

Show My Solution

```
import com.skillrack.lab.Circle;
import com.skillrack.lab.Rectangle;
import com.skillrack.lab.RightTriangle;
import com.skillrack.lab.Shape;
import java.util.Scanner;

public class Hello {

    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        int length = scanner.nextInt();
        int breadth = scanner.nextInt();
        Shape rectangle = new Rectangle(length, breadth);

        int base = scanner.nextInt();
        int height = scanner.nextInt();
        Shape rightAngledTriangle = new RightTriangle(base, height);

        int radius = scanner.nextInt();
        Shape circle = new Circle(radius, radius);

        rectangle.printArea();
        rightAngledTriangle.printArea();
        circle.printArea();

    }
}

package com.skillrack.lab;
public class Rectangle extends Shape{
    public Rectangle(int x,int y){
        super(x,y);
    }
    public void printArea(){
        System.out.println(getX()*getY());
    }
}

package com.skillrack.lab;
public class RightTriangle extends Shape
{
    double area;
    public RightTriangle(int x,int y){
        super(x,y);
    }
    public void printArea(){
        area=(0.5)*super.getX()*super.getY();
        System.out.format("%.2f \n",area);
    }
}

package com.skillrack.lab;
public class Circle extends Shape
{
    double area;
    public Circle(int x,int y){
        super(x,y);

    }
    public void printArea()
    {
        area=(3.142857143)*getX()*getY();
        double aproxArea=Math.round(area*100.0)/100.0;
        System.out.format("%.2f \n",aproxArea);
    }
}
```



College/University LeaderBoard | Global LeaderBoard

ompleted Programs - Lab - OOP - Java - CS8383 - Abstract Class - PART 006

</> Shape - printArea (Abstract Class) (Id-8934)

Define Java classes Rectangle, Triangle and Circle to extend the abstract class Shape. These three classes must implement the method printArea provided in the abstract class Shape. Also these three classes must provide the appropriate constructors.

Note:
- The formula to calculate the area for a rectangle is length*breadth
- The formula to calculate the area for a right angled triangle is 1/2 * base * height
- The formula to calculate the area for a cirlcle is 22/7 * radius * radius
- The area for right angled triangle and the circle must be printed with precision upto 2 decimal places

Example Input/Output 1:

Input:
10 20
9 5
11

Output:
200
22.50
380.29

The abstract class **Shape.java** is given below.

```
package com.skillrack.lab;

public abstract class Shape {
    //For rectangle x is length and y is breadth
    //For right angled triangle x is base and y is height
    //For circle both x and y denotes radius
    int x;
    int y;

    public Shape(int x, int y){
        this.x = x;
        this.y = y;
    }

    public abstract void printArea();


    public int getX() {
        return x;
    }

    public void setX(int x) {
        this.x = x;
    }

    public int getY() {
        return y;
    }

    public void setY(int y) {
        this.y = y;
    }
}
```

The class **Hello.java** containing the main method is given below.

```
import com.skillrack.lab.Circle;
import com.skillrack.lab.Rectangle;
import com.skillrack.lab.RightTriangle;
import com.skillrack.lab.Shape;
import java.util.Scanner;

public class Hello {

    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        int length = scanner.nextInt();
        int breadth = scanner.nextInt();
        Shape rectangle = new Rectangle(length, breadth);

        int base = scanner.nextInt();
        int height = scanner.nextInt();
        Shape rightAngledTriangle = new RightTriangle(base, height);

        int radius = scanner.nextInt();
        Shape circle = new Circle(radius, radius);

        rectangle.printArea();
        rightAngledTriangle.printArea();
        circle.printArea();

    }
}
```

Show My Solution

Result: thus the java program for finding area of Rectangle, triangle and circle using abstract method is executed successfully and output is verified.

Exp no: 7

Date: 05.10.20

title: User Defined exception

Aim: To develop a java program to implement user defined exception handling.

Algorithm:

1. Start
2. Create a class that represents user defined exception named as my exception.
3. Call constructor of parent exception within the class.
4. Create a main class that uses above my exception class.
5. Add main method with in the main class.
6. Throw an object of user defined exception within the try block.
7. Print the message from my exception object.
8. Display the result.
9. Stop.

```java
private static void withdrawMoney(int amount) throws InvalidAmountException, LimitExceededException
{
    if(amount<=0)
    {
        throw new InvalidAmountException("Amount to withdraw must be positive");
    }
    else if(amount>50000)
    {
        throw new LimitExceededException("Maximum withdrawl amount is 50000");
    }
    else
    {
        System.out.println("Valid amount");
    }

}
}//end of class Hello
```

skillrack

lab

InvalidAmountException.java

LimitExceededException.java

```java
package com.skillrack.lab;
public class InvalidAmountException extends Exception
{
    public InvalidAmountException(String message)
    {
        super(message);
    }
}
```

Save Run

skillrack

lab

InvalidAmountException.java

LimitExceededException.java

```java
package com.skillrack.lab;
public class LimitExceededException extends Exception
{
    public LimitExceededException(String message)
    {
        super(message);
    }
}
```

Save Run

Input:
3
1000
-99
9992929

Output:
Valid amount
Amount to withdraw must be positive
Maximum withdrawal amount is 50000
1 1 1

Example Input/Output 2:
Input:
7
1000
-99
9992929
5000
0
45000
90000

Output:
Valid amount
Amount to withdraw must be positive
Maximum withdrawal amount is 50000
Valid amount
Amount to withdraw must be positive
Valid amount
Maximum withdrawal amount is 50000

Result:

thus, the java program for user defined exception handling is executed succesfully and output is verified.

Exp no: 8

Date: 05.10.20

Title: File operations

Aim: TO develop a java program that reads the file name from the user, displays information about whether the file exists, whether the file is readable, or writable, the the type of file and the length of the file in bytes.

Algorithm:

1. Start
2. get the file name from user.
3. check whether is exist or not. if the given file exist then,
3.1. find whether the file is readable or not, then print result.
3.2. find whether the file is writeable or not, then print result.
3.3. find length of file in n bytes.

a) if the file name ends with .jpg, .png, .gif extensions then, print as image file.

b) if the file name ends with .exe then print executable file.

c) if the file name ends with ".java" extension then, print file as java file.

d) if the file name ends with ".txt" extension then, print text file.

e) if the file name is did'nt mention with proper extension at end of file then file is unknown.

6. print the details of file.
7. Exit.

Show My Solution

```
package com.skillrack.lab;

import java.io.*;

public class FileInfoPrinter {
public static void printDetails(String filePath) throws Exception
{
    File f=new File(filePath);
    if(f.exists())
    {
        System.out.println("Present");
        BufferedReader br=new BufferedReader(new FileReader(f));
        System.out.println(br.readLine());
        br.close();

    }
    else{
        System.out.println("NotPresent");
    }

}
//end of FileInfoPrinter class



} // end of FileInfoPrinter class
```

</> Print File Information (Id-8961)

Fill in the lines of code for the file **FileInfoPrinter.java** implementing the static method **printDetails** whose method signature is given below.

public static void printDetails(String filePath) throws Exception

The method must perform the following.
- If a file exists in the path denoted by filePath, then print Present. Else print NotPresent
- If the file exists in the path denoted by filePath, then print the first line available in the file. (If a file exists always first line will be available)

Example Input/Output 1:

Here file1.txt is present and is writable (indicated by 1). It contains three lines
skill
rack
com

Input:
file1.txt

Output:
Present
skill

Example Input/Output 2:

Here file9.txt is not present.

Input:
file9.txt

Output:
NotPresent

Show My Solution

Result: thus the java program for reading a file name from the user, displays information about whether the file exists, whether the file is readable, or writable, the type of file and the length of file in bytes have been executed successfully and output is verified.

Exp.no: 9

Date: 06-10-20

Title: Multithreaded Application

Aim: Develop a java program that implements a multi-threaded application that has three threads. First thread generates a random integer every 1 second and if the value is even, second thread computes the square of number and prints. if the value is odd then print the value of cube of the number.

Algorithm:

1. Start
2. Creating a class which is extending from 'thread' class
3. In first thread class implement run() method.
3.1. Initialize num value to zero
3.2. Creating object of an Random class as r.
3.3. num = r.nextInt(10)
3.4. It generates the random integer numbers from 0 to 9.
If generated number is even number, create object of class and start
3.5. else create object of third class and start it.
3.6. call thread.sleep(1000)
4. In second thread class implement run() method.
4.1. print the square value of given number.
5. In third thread class implement run() method
5.1. print the cube value of given number.
6. Create main class and add main method with in class.
6.1. Creating the object of the first thread class and start it.
7. Stop.

```
import java.io.*;
import java.lang.*;
import java.util.*;
class even implements Runnable{
    public int n;
    public even(int n){
        this.n=n;

    }
    public void run(){
        System.out.println("Even Square:"+n*n);
    }
}
class odd implements Runnable{
    public int n;
    public odd(int n){
        this.n=n;
    }
    public void run(){
        System.out.println("Odd Cube:"+n*n*n);
    }
}
class MultiThread extends Thread{
    public void run(){
        int j=0;
        Random r= new Random();
        try{
            for(int i=0;i<10;i++){
                j=r.nextInt(100);
                System.out.println("Main Thread No. "+j);
                if(j%2==0){
                    Thread a=new Thread(new even(j));
                    a.start();
                }
                else{
                    Thread b=new Thread(new odd(j));
                    b.start();
                }
                Thread.sleep(1000);
            }
        }catch(Exception e){
            System.out.println(e.getMessage());
        }
    }
}
public class Main{
    public static void main(String[] args){
        MultiThread m= new MultiThread();
        m.start();
    }
}
```

input

```
Main Thread No. 40
Even Square:1600
Main Thread No. 3
Odd Cube:27
Main Thread No. 74
Even Square:5476
Main Thread No. 92
Even Square:8464
Main Thread No. 96
Even Square:9216
Main Thread No. 64
Even Square:4096
Main Thread No. 89
Odd Cube:704969
Main Thread No. 14
Even Square:196
Main Thread No. 95
Odd Cube:857375


..Program finished with exit code 0
Press ENTER to exit console.
```

Result:

thus the java program that implements a multi-threaded application that has three threads have executed successfully and output is verified.

Exp.no: 10

Date: 09.10.20

title: Generic Function

Aim: TO develop a java program to find the maximum value from the given type of elements using a generic function.

Algorithm:

1. Start
2. Create a class as generic
3. Extending type parameter T from comparable interface with access specifier as public.

   public <T extends comparable <T >>T max (T [] element).
4. Define a method max with array of element as parameter.

4.1. T max = element [0]

4.2. For (i=0; i< element. length; i++)

   If(element >max)

   Max = element [i]

4.3. return maximum element.

5. Create another class name genericdemo
6. Add main method with in the class generic demo.
7. Create an object for previous class "generic".
8. Create an array for integers and call max method.
9. Create an array for character and call max method.
10. Display the result.
11. Stop.

```java
package com.skillrack.lab;

public class Man implements Comparable<Man> {

    int age;

    public Man(int age) {
        this.age = age;
    }

    @Override
    public int compareTo(Man t) {
        return this.age - t.age;
    }

    public String toString(){
        return this.age+"";
    }
}
```

Show My Solution

```java
package com.skillrack.lab;

public class MaxFinder {
@SuppressWarnings("unchecked")
public static <T extends Comparable<T>> T max(T... elements)
{
    T max=elements[0];
    for(T element: elements)
    {
        if(element.compareTo(max)>0){
            max=element;
        }

    }
    return max;
}
} //end of class MaxFinder
```

</> Generics - Maximum of Varargs (Id-9051)

Fill in the lines of code for the file **MaxFinder.java** to implement the static method **max** which will
- Accept a varargs of any class which extends Comparable and return the maximum element in the varargs.

IMPORTANT: Use the annotation **@SuppressWarnings("unchecked")** on the method to avoid warnings during compilation

The code present in **Hello.java** is given below.

```
import com.skillrack.lab.Man;
import com.skillrack.lab.MaxFinder;
import java.util.Scanner;

public class Hello {

    public static void main(String[] args) throws Exception {
        Scanner sc = new Scanner(System.in);
        int N = sc.nextInt();
        Integer arr[] = new Integer[N];
        String strarr[] = new String[N];
        Man manarr[] = new Man[N];
        for(int index=0; index < N; index++){
            arr[index] = sc.nextInt();
            manarr[index] = new Man(arr[index]);
            strarr[index] = sc.nextLine().trim();
        }

        System.out.println(MaxFinder.max(arr));
        System.out.println(MaxFinder.max(strarr));
        System.out.println(MaxFinder.max(manarr));
    }

}
```

The code present in **Man.java** is given below.

```
package com.skillrack.lab;

public class Man implements Comparable<Man> {

    int age;

    public Man(int age) {
        this.age = age;
    }

    @Override
    public int compareTo(Man t) {
        return this.age - t.age;
    }

    public String toString(){
        return this.age+"";
    }
}
```

Show My Solution

Result:

thus, the java program to find the maximum value from the given type of elements using a generic function is executed successfully and output is verified.

Exp.no: 11(a)

Date: 10.10.20

title: Calculator for decimal manipulation

Aim: To create java application for calculator for decimal manipulation

Algorithm:

1. Start
2. import packages for graphic programming
3. Create class calculator which implements. contain frame, Button (b1 to b16), panel, textfield, Grid layout
4. Create constructor calculator and object for frame, panel and each button, textfield, gridlayout
   4.1. add (p) to frame, set size and set visible (true)
   4.2. Create void method ActionPerformed (ActionEvent) and check e.getSource() == (for each button)
5. Create main method and object to create the calculator class
6. Stop.

Main.java

Files

```
1  import java.awt.*;
2  import java.awt.event.*;
3  public class Main implements
     ActionListener
4  {
5      int c,n;
6      String s1,s2,s3,s4,s5;
7      Frame f;
8      Button b1,b2,b3,b4,b5,b6,b7,b8
         ,b9,b10,b11,b12,b13,b14,b15
         ,b16,b17;
9      Panel p;
10     TextField tf;
11     GridLayout g;
12     Main()
13     {
14         f = new Frame("My
             calculator");
15         p = new Panel();
16         f.setLayout(new FlowLayout
             ());
17         b1 = new Button("0");
18         b1.addActionListener(this);
19         b2 = new Button("1");
20         b2.addActionListener(this);
21         b3 = new Button("2");
22         b3.addActionListener(this);
23         b4 = new Button("3");
24         b4.addActionListener(this);
25         b5 = new Button("4");
26         b5.addActionListener(this);
27         b6 = new Button("5");
```

Code | Console | Output | Commands

Main.java

Files

```
28          b6.addActionListener(this);
29          b7 = new Button("6");
30          b7.addActionListener(this);
31          b8 = new Button("7");
32          b8.addActionListener(this);
33          b9 = new Button("8");
34          b9.addActionListener(this);
35          b10 = new Button("9");
36          b10.addActionListener(this
              );
37          b11 = new Button("+");
38          b11.addActionListener(this
              );
39          b12 = new Button("-");
40          b12.addActionListener(this
              );
41          b13 = new Button("*");
42          b13.addActionListener(this
              );
43          b14 = new Button("/");
44          b14.addActionListener(this
              );
45          b15 = new Button("%");
46          b15.addActionListener(this
              );
47          b16 = new Button("=");
48          b16.addActionListener(this
              );
49          b17 = new Button("C");
50          b17.addActionListener(this
              );
51          tf = new TextField(20);
```

Code | Console | Output | Commands

```
        tf = new TextField(20);
        f.add(tf);
        g = new GridLayout(4,4,10
          ,20);
        p.setLayout(g);
        p.add(b1);p.add(b2);p.add
          (b3);p.add(b4);p.add(b5
          );p.add(b6);p.add(b7);p
          .add(b8);p.add(b9);
        p.add(b10);p.add(b11);p.add
          (b12);p.add(b13);p.add
          (b14);p.add(b15);p.add
          (b16);p.add(b17);
        f.add(p);
        f.setSize(300,300);
        f.setVisible(true);
    }
    public void actionPerformed
      (ActionEvent e)
    {
        if(e.getSource()==b1)
        {
            s3 = tf.getText();
            s4 = "0";
            s5 = s3+s4;
            tf.setText(s5);
        }
        if(e.getSource()==b2)
        {
            s3 = tf.getText();
            s4 = "1";
            s5 = s3+s4;
```

Main.java

Files

```
74              s5 = s3+s4;
75              tf.setText(s5);
76          }
77          if(e.getSource()==b3)
78          {
79              s3 = tf.getText();
80              s4 = "2";
81              s5 = s3+s4;
82              tf.setText(s5);
83          }if(e.getSource()==b4)
84          {
85              s3 = tf.getText();
86              s4 = "3";
87              s5 = s3+s4;
88              tf.setText(s5);
89          }
90          if(e.getSource()==b5)
91          {
92              s3 = tf.getText();
93              s4 = "4";
94              s5 = s3+s4;
95              tf.setText(s5);
96          }
97          if(e.getSource()==b6)
98          {
99              s3 = tf.getText();
100             s4 = "5";
101             s5 = s3+s4;
102             tf.setText(s5);
103         }
104         if(e.getSource()==b7)
105         {
```

Code | Console | Output | Commands

```
05      {
06          s3 = tf.getText();
07          s4 = "6";
08          s5 = s3+s4;
09          tf.setText(s5);
10      }
11      if(e.getSource()==b8)
12      {
13          s3 = tf.getText();
14          s4 = "7";
15          s5 = s3+s4;
16          tf.setText(s5);
17      }
18      if(e.getSource()==b9)
19      {
20          s3 = tf.getText();
21          s4 = "8";
22          s5 = s3+s4;
23          tf.setText(s5);
24      }
25      if(e.getSource()==b10)
26      {
27          s3 = tf.getText();
28          s4 = "9";
29          s5 = s3+s4;
30          tf.setText(s5);
31      }
32      if(e.getSource()==b11)
33      {
34          s1 = tf.getText();
35          tf.setText("");
36          c=1;
```

```
36              c=1;
37
38          }
39          if(e.getSource()==b12)
40          {
41              s1 = tf.getText();
42              tf.setText("");
43              c=2;
44
45          }
46          if(e.getSource()==b13)
47          {
48              s1 = tf.getText();
49              tf.setText("");
50              c=3;
51
52          }
53          if(e.getSource()==b14)
54          {
55              s1 = tf.getText();
56              tf.setText("");
57              c=4;
58
59          }
60          if(e.getSource()==b15)
61          {
62              s1 = tf.getText();
63              tf.setText("");
64              c=5;
65
66          }
67          if(e.getSource()==b16)
```

Main.java

Files

```
167             if(e.getSource()==b16)
168             {
169                 s2 = tf.getText();
170                 if(c==1)
171                 {
172                     n = Integer
                        .parseInt(s1
                        )+Integer.parseInt
                        (s2);
173                     tf.setText(String
                        .valueOf(n));
174                 }
175                 else
176                 if(c==2)
177                 {
178                     n = Integer
                        .parseInt(s1
                        )-Integer.parseInt
                        (s2);
179                     tf.setText(String
                        .valueOf(n));
180                 }
181                 else
182                 if(c==3)
183                 {
184                     n = Integer
                        .parseInt(s1
                        )*Integer.parseInt
                        (s2);
185                     tf.setText(String
                        .valueOf(n));
186                 }
```

Code | Console | Output | Commands

Main.java

Files

```
187                 if(c==4)
188                 {
189                     try
190                     {
191                         int p=Integer
                        .parseInt(s2);
192                         if(p!=0)
193                         {
194
                              n = Integer
                        .parseInt(s1
                        )/Integer.parseInt
                        (s2);
195                         tf.setText
                        (String.valueOf(n
                        ));
196                          }
197                          else
198                             tf.setText
                        ("infinite");
199
200                     }
201                     catch(Exception i
                        ){}
202                 }
203                 if(c==5)
204                 {
205                     n = Integer
                        .parseInt(s1
                        )%Integer.parseInt
                        (s2);
206                     tf.setText(String
```

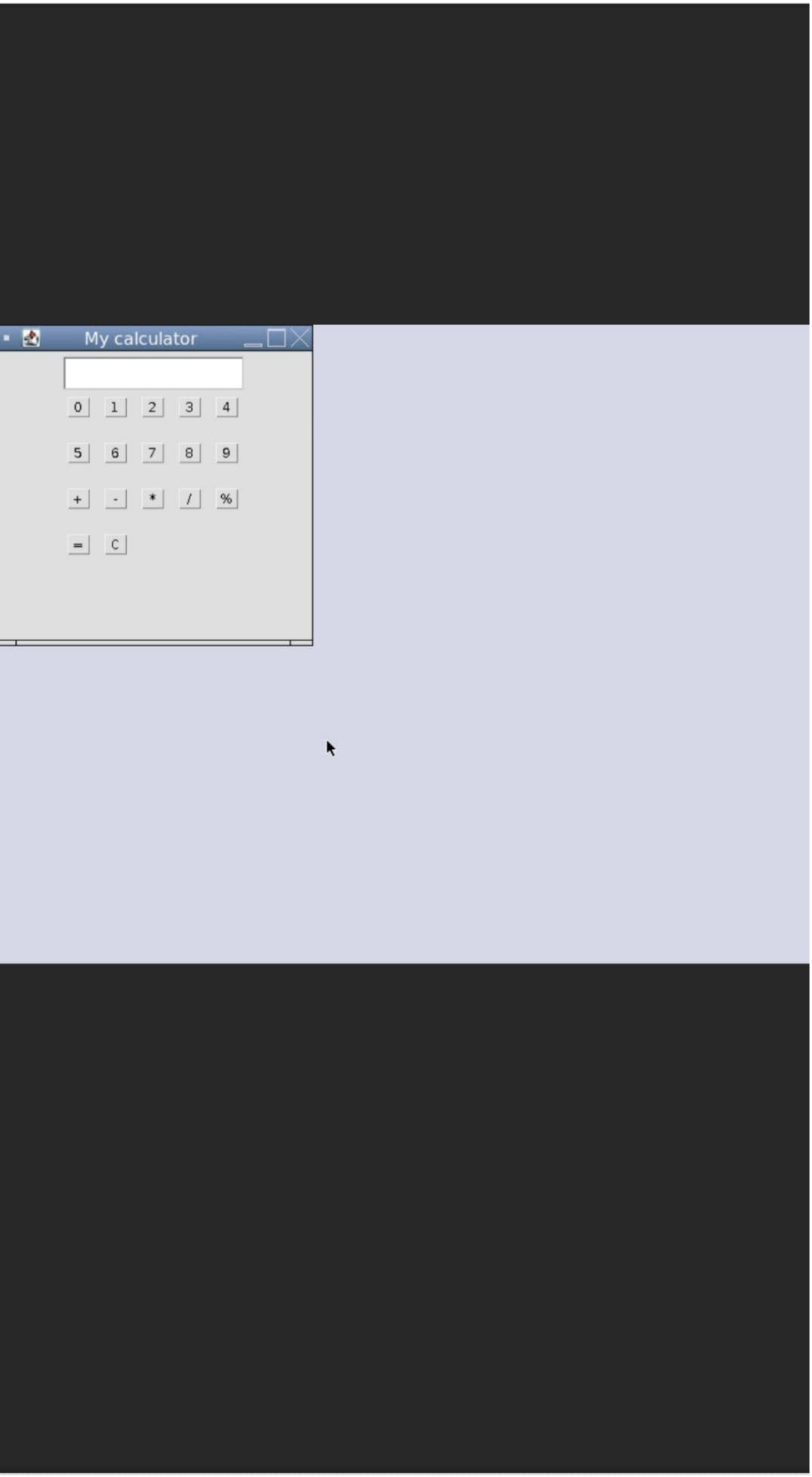
Code | Console | Output | Commands

Main.java

Files

```
                    )%Integer.parseInt
                    (s2);
206                 tf.setText(String
                    .valueOf(n));
207             }
208         }
209         if(e.getSource()==b17)
210         {
211             tf.setText("");
212         }
213     }
214
215     public static void main
          (String[] abc)
216     {
217         Main v = new Main();
218     }
219 }
```

```
OpenJDK Runtime Environment (build 11.0
.6+10-post-Ubuntu-1ubuntu118.04.1)
> java -classpath .:/run_dir/junit-4.12
.jar:target/dependency/* Main
Starting X
.....5.. 4.. 3.. 2.. 1..
```



Result:

The java application for the calculator for decimal manipulator is executed and output is verified.

Exp.No: 11(b)

Date: 10.10.20

Title: Scientific Calculator

Aim: To create java application for scientific calculator.

Algorithm:

1. Start
2. import packages for graphics programming
3. Create a class calculatorframe which is extends from jframe. class & panel object of jpane class, result of double type, last command of string data type, start of boolean type, as members of calculator frame.

3.1. Define display object of jbutton class

4. Create an constructor of that calculatorframe class.

4.1. Set size for the frame, set layout for the frame by using border layout.

4.2. set result=0, last command = "=" and start=true.

4.3. Add button to display and make display set enabled as false.

4.4. Create an object of inertaction as insert and commandaction as command.

4.5. By using object of jpanel class set layout of frame like grid layout.

4.6. Add all buttons to the panel by using add button() method.

5. Define add button method for adding label to button.

6. Define a class interaction which is implements from actionlistener interface. if input is any operation.

6.1. Define a class commandation which implements actionlistener interface.

if input is any command it check whether command == "=" and display the set text and make start as false.

Else it makes last command as command.

Or else.

It calculate the result by using command and text from true.

7. Define a class calculate of void data type.

7.1. Check the last command and command if given input is matches with last command.

8. Define a class calculator and add main method.

8.1. Add eventqueue class available in java library and add ~~static~~ method invoke later available class the implements runnable interface.

8.2 Create an object of previous class and make the frame visibility as true, set default close operation for the frame.

9. Stop.

Main.java

Files

```
1  import java.awt.*;
2  import java.awt.event.*;
3  import javax.swing.*;
4  import javax.swing.event.*;
5
6  class SwingMain extends JFrame {
7      private final Font BIGGER_FONT
         = new Font("monspaced",Font
         .PLAIN, 20);
8      private JTextField textfield;
9      private boolean number = true;
10     private String equalOp = "=";
11     private MainOp op = new MainOp
         ();
12
13     public Main() {
14         textfield = new JTextField
             ("", 12);
15         textfield
             .setHorizontalAlignment
             (JTextField.RIGHT);
16         textfield.setFont
             (BIGGER_FONT);
17         ActionListener
             numberListener = new
             NumberListener();
18         String buttonOrder =
             "1234567890 ";
19         JPanel buttonPanel = new
             JPanel();
20         buttonPanel.setLayout(new
             GridLayout(4, 4, 4, 4));
21         for (int i = 0; i <
             buttonOrder.length(); i
             ++) {
```

```
22              String key =
                  buttonOrder.substring
                  (i, i+1);
23              if (key.equals(" ")) {
24                  buttonPanel.add(new
                      JLabel(""));
25              } else {
26                  JButton button =
                      new JButton(key);
27                  button
                      .addActionListener
                      (numberListener);
28                  button.setFont
                      (BIGGER_FONT);
29                  buttonPanel.add
                      (button);
30              }
31          }
32          ActionListener
              operatorListener = new
              OperatorListener();
33          JPanel panel = new JPanel
              ();
34          panel.setLayout(new
              GridLayout(4, 4, 4, 4));
35          String[] opOrder = {"+", "
              -", "*", "/","=","C"
              ,"sin","cos","log"};
36          for (int i = 0; i < opOrder
              .length; i++) {
37              JButton button = new
                  JButton(opOrder[i]);
38              button
                  .addActionListener
                  (operatorListener);
```

```
38              button
                  .addActionListener
                  (operatorListener);
39              button.setFont
                  (BIGGER_FONT);
40              panel.add(button);
41          }
42          JPanel pan = new JPanel();
43          pan.setLayout(new
              BorderLayout(4, 4));
44          pan.add(textfield,
              BorderLayout.NORTH );
45          pan.add(buttonPanel ,
              BorderLayout.CENTER);
46          pan.add(panel ,
              BorderLayout.EAST);
47          this.setContentPane(pan);
48          this.pack();
49          this.setTitle("Calculator"
              );
50          this.setResizable(false);
51      }
52      private void action() {
53          number = true;
54          textfield.setText("");
55          equalOp = "=";
56          op.setTotal("");
57      }
58      class OperatorListener
          implements ActionListener {
59          public void actionPerformed
              (ActionEvent e) {
60              String displayText =
                  textfield.getText();
61              if (e.getActionCommand
```

```
61              if (e.getActionCommand
                  ().equals("sin"))
62              {
63                  textfield.setText
                     ("" + Math.sin
                     (Double.valueOf
                     (displayText
                     ).doubleValue()));
64
65              }else
66              if (e.getActionCommand
                  ().equals("cos"))
67              {
68                  textfield.setText
                     ("" + Math.cos
                     (Double.valueOf
                     (displayText
                     ).doubleValue()));
69
70              }
71              else
72              if (e.getActionCommand
                  ().equals("log"))
73              {
74                  textfield.setText
                     ("" + Math.log
                     (Double.valueOf
                     (displayText
                     ).doubleValue()));
75
76              }
77              else if (e
                  .getActionCommand
                  ().equals("C"))
78              {
```

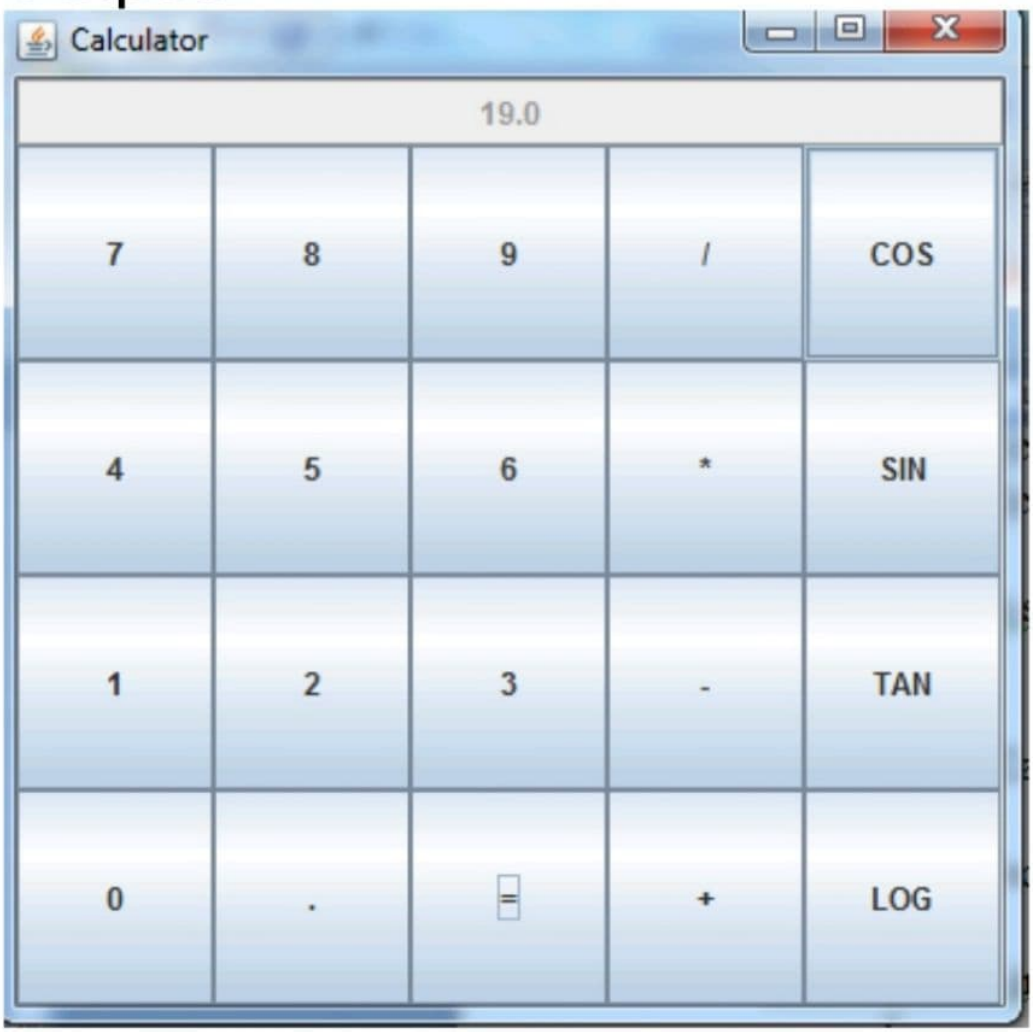
```
78                  {
79                      textfield.setText
                         ("");
80                  }
81
82                  else
83                  {
84                      if (number)
85                      {
86
87                          action();
88                          textfield
                         .setText("");
89
90                      }
91                      else
92                      {
93                          number = true;
94                          if (equalOp
                         .equals("="))
95                          {
96                              op.setTotal
                         (displayText);
97                          }else
98                          if (equalOp
                         .equals("+"))
99                          {
100                             op.add
                         (displayText);
101                         }
102                         else if
                         (equalOp.equals("
                         -"))
103                         {
104                             op.subtract
```

```
104                         op.subtract
                        (displayText);
105                     }
106                     else if
                        (equalOp.equals("
                        *"))
107                     {
108                         op.multiply
                        (displayText);
109                     }
110                     else if
                        (equalOp.equals("
                        /"))
111                     {
112                         op.divide
                        (displayText);
113                     }
114
115                     textfield
                        .setText("" + op
                        .getTotalString
                        ());
116                     equalOp = e
                        .getActionCommand
                        ();
117                 }
118             }
119         }
120     }
121     class NumberListener implements
          ActionListener {
122         public void actionPerformed
              (ActionEvent event) {
123           String digit = event
                .getActionCommand();
```

```
123             String digit = event
                  .getActionCommand();
124             if (number) {
125                 textfield.setText
                      (digit);
126                 number = false;
127             } else {
128                 textfield.setText
                      (textfield.getText
                      () + digit);
129             }
130         }
131     }
132     public class MainOp {
133         private int total;
134         public MainOp() {
135             total = 0;
136         }
137         public String
              getTotalString() {
138             return ""+total;
139         }
140         public void setTotal(String
              n) {
141             total = convertToNumber
                  (n);
142         }
143         public void add(String n) {
144             total +=
                  convertToNumber(n);
145         }
146         public void subtract(String
```

```
146         public void subtract(String
              n) {
147           total -=
                convertToNumber(n);
148         }
149         public void multiply(String
              n) {
150           total *=
                convertToNumber(n);
151         }
152         public void divide(String n
              ) {
153           total /=
                convertToNumber(n);
154         }
155         private int convertToNumber
              (String n) {
156           return Integer.parseInt
                (n);
157         }
158       }
159   }
160   class SwingMain {
161       public static void main
            (String[] args) {
162         JFrame frame = new Main();
163         frame
              .setDefaultCloseOperation
              (JFrame.EXIT_ON_CLOSE);
164         frame.setVisible(true);
165       }
166   }
```

Output:



Result:

thus the application for scientific calculator is created.