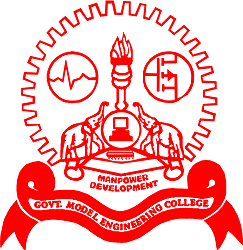
**CSL203 OBJECT ORIENTED PROGRAMMING LAB LABORATORY RECORD**

**CSU19B03 MDL19CS005 Name: ADITHYA A**

**B. Tech. Computer Science & Engineering**



**Department of Computer Engineering**

**Model Engineering College Thrikkakara, Kochi 682021 Phone: +91.484.2575370**

[**http://www.mec.ac.in**](http://www.mec.ac.in/)[**ho**](mailto:hodcs@mec.ac.in)[**dcs@mec.ac.in**](mailto:dcs@mec.ac.in)

***CERTIFIED THAT THIS IS A BONOFIDE RECORD OF THE WORKDONE IN THE LABORATORY/WORKSHOP OF MODEL ENGINEERING COLLEGE, THRIKKAKARA***

# Staff in Charge Head of the Department

**Submitted for the Practical Examination in**

**................................................................................... held on .....................................**

**Internal Examiner External Examiner**

**LIST OF EXPERIMENTS**

|  |  |  |  |
| --- | --- | --- | --- |
| **Sl No:** | **Date** | **Name of the Experiments** | **Page No** |
| 1 | 08/10/2020 | Write a Java program that checks whether a given string is a palindrome or not. Ex: MALAYALAM is palindrome. | 1 |
| 2 | 16/10/2020 | Write a Java Program to find the frequency of a given character in a string. | 3 |
| 3 | 17/09/2020 | Write a Java program to multiply two given matrices. | 5 |
| 4 | 17/09/2020 | Write a Java program to display the transpose of a given matrix | 8 |
| 5 | 16/10/2020 | Write a Java program to reverse a given string. | 10 |
| 6 | 24/09/2020 | Write a Java program to find the second smallest element in an array. | 12 |
| 7 | 24/09/2020 | Write a Java program to check whether a given number is prime or not. | 14 |
| 8 | 19/10/2020 | Exercise to understand inheritance | 16 |
| 9 | 20/10/2020 | Exercise to understand polymorphism | 20 |
| 10 | 24/09/2020 | Write a Java program to calculate the area of different shapes namely circle, rectangle, and triangle using the concept of method overloading. | 22 |
| 11 | 05/11/2020 | Two classes with one object | 24 |
| 12 | 10/11/2020 | Write a Java program to demonstrate the use of garbage collector. | 26 |
| 13 | 13/11/2020 | Write a file handling program in Java with reader/writer. | 28 |
| 14 | 17/11/2020 | Write a Java program that read from a file and write to file by handling all file related exceptions. | 32 |
| 15 | 24/11/2020 | Write a Java program that shows the usage of try, catch, throws and finally. | 35 |
| 16 | 05/12/2020 | Write a Java program to create two threads: One for displaying all odd number between 1 and 100 and second thread for displaying all even numbers between 1 and 100. | 38 |

|  |  |  |  |
| --- | --- | --- | --- |
| 17 | 16/12/2020 | Write a Java program that shows thread synchronization. | 41 |
| 18 | 08/12/2020 | Doubly Linked List | 44 |

**AIM:**

1. **PALINDROME**

Write a Java program that checks whether a given string is a palindrome or not. Ex: MALAYALAM is palindrome

# ALGORITHM:

* 1. Get the string to check for palindrome
  2. Hold the string in temporary variable
  3. Reverse the string
  4. Compare the temporary string with reversed string
  5. If both numbers are same, print "palindrome string"
  6. Else print "not palindrome string"

# PROGRAM:

/\* Program to check whether a given word is palindrome or not \*/ import java.util.\*;

class palindrome

{

public static void main(String args[])

{

String s1,s2; char ch;

s2 = "";

Scanner sc = new Scanner(System.in); System.out.print("Enter the String: "); s1 = sc.next();

for(int i=s1.length()-1;i>=0;i--)

{

ch = s1.charAt(i); s2 = s2 + ch;

}

if(s2.equalsIgnoreCase(s1))

else

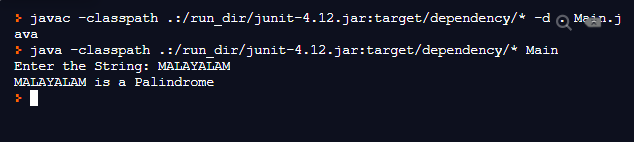
}

}

System.out.println(s1 + " is a Palindrome");

System.out.println(s1 + " is not a Palindrome");

# OUTPUT:

****

**RESULT:**

Java program to check whether the given string is a ‘Palindrome’ or not was implemented successfully.

# AIM:

1. **FREQUENCY OF A GIVEN CHARACTER IN A STRING**

Write a Java Program to find the frequency of a given character in a string.

# ALGORITHM:

* 1. Get the string from on which the operation is to be done.
  2. Get the character whose frequency is to be found out.
  3. Iterate through each character in the string and check whether it is that character
  4. If yes, increment the count.
  5. Finally, print the count obtained.

# PROGRAM:

import java.io.\*; class charcount

{

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

{

BufferedReader br = new BufferedReader(new InputStreamReader(System.in)); String s1;

System.out.print("Enter the String: "); s1 = br.readLine();

System.out.print("Enter the character to be checked: "); char ch = (char)System.in.read();

int count = 0;

for(int i=0;i<s1.length()-1;i++)

{

if(ch == s1.charAt(i))

count++;

}

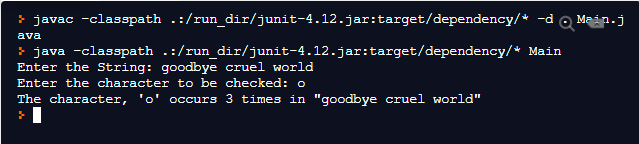
System.out.println("The character, \'" + ch + "\' occurs " + count + " times in \"" + s1 +

"\"");

}

}

# OUTPUT:

****

**RESULT:**

Program to find the frequency of a given character from a given string was implemented successfully.

# AIM:

1. **MATRIX MULTIPLICATION**

Develop a Java program to find the product of two matrices.

# ALGORITHM:

* 1. Declare 2 two dimensional arrays with some integer values.
  2. Carry out Matrix multiplication by multiplying the rows of the first matrix with the columns of the second.
  3. This can be done using three loops. Store the result in another two dimensional array.
  4. Display the resultant matrix along with the original matrices.

# PROGRAM:

class matmul

{

public static void main(String args[])

{

int c[][] = new int[3][3]; int a[][] = {

{12,4,5,2},

{9,6,3,1},

{2,15,-4,5}

};

int b[][] = {

{-5,19,4},

{7,-3,8},

{0,1,14},

{2,0,0}

};

int i,j,k; System.out.println("Matrix A:"); for(i=0;i<3;i++)

{

for(j=0;j<4;j++)

System.out.print(a[i][j]+"\t"); System.out.println();

}

System.out.println("Matrix B:"); for(i=0;i<4;i++)

{

for(j=0;j<3;j++)

System.out.print(b[i][j]+"\t"); System.out.println();

}

for(i=0;i<3;i++)

{

for(j=0;j<4;j++)

{

for(k=0;k<3;k++)

c[i][k] += a[i][j]\*b[j][k]; }

}

System.out.println("Product of the two matrices is:"); for(i=0;i<3;i++)

{

for(j=0;j<3;j++)

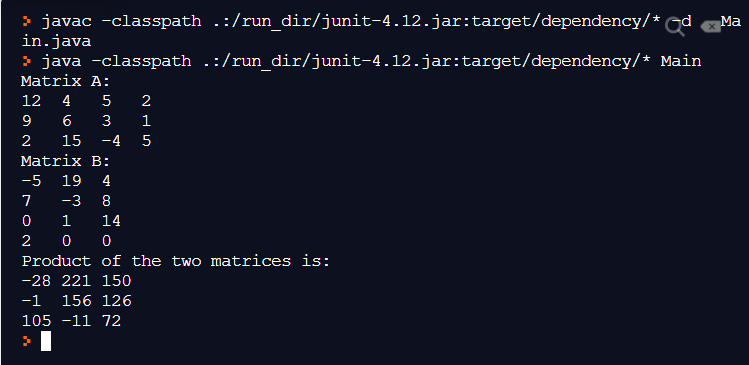
System.out.print(c[i][j]+"\t"); System.out.println();

}

}

}

# OUTPUT:

****

**RESULT:**

Program to calculate the product of two matrices was implemented successfully.

# AIM:

1. **TRANSPOSE OF A MATRIX**

Develop a JAVA program to find the transpose of a matrix.

# ALGORITHM:

* 1. Declare a two-dimensional array with some integer values and another two-dimensional array without any values.
  2. Iterate through each element in the matrix and store it in the second one by interchanging the row number and column number.
  3. Print the resultant matrix along with the original matrix.

# PROGRAM:

// Program to generate the transpose of a given matrix class transpose {

public static void main(String args[]) { int a[][] = {

{1,2,3,4},

{5,6,7,8},

{9,10,11,12}

};

int b[][] = new int[4][3];

System.out.println("The given matrix is:\n"); for(int i=0;i<3;i++)

{

for(int j=0;j<4;j++)

{

System.out.print(a[i][j]+"\t");

}

System.out.println();

}

for(int i=0;i<3;i++) {

for(int j=0;j<4;j++)

b[j][i] = a[i][j];

}

System.out.println("The transpose of the matrix is:\n"); for(int i=0;i<4;i++) {

for(int j=0;j<3;j++) {

System.out.print(b[i][j]+"\t");

}

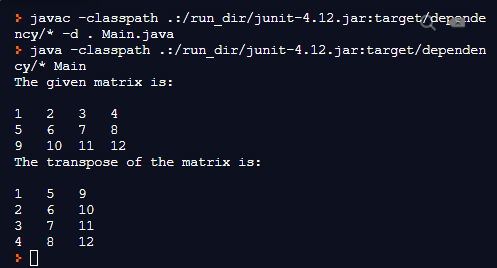
System.out.println();

}

}

}

# OUTPUT:

****

**RESULT:**

Program to find the transpose of a matrix was implemented successfully.

# AIM:

1. **REVERSING A GIVEN STRING**

Write a Java program to reverse a given string.

# ALGORITHM:

* 1. Declare two strings. One to store the input text and the other to store the reversed text.
  2. Get the string to be reversed from the user.
  3. Iterate from the last element to the first element (i.e. in the reverse order) of the input string and store each character to the end of the empty string one-by-one.
  4. Print the reversed string obtained.

# PROGRAM:

import java.util.\*; class reverse {

public static void main(String args[]) { reverse r1 = new reverse();

Scanner sc = new Scanner(System.in); String s1;

System.out.print("Enter the String: "); s1 = sc.nextLine();

String s2 = r1.strrev(s1);

System.out.println("The reversed String is: " + s2);

}

String strrev(String str) { String rev = "";

for(int i=str.length()-1;i>=0;i--)

{

rev = rev + str.charAt(i);

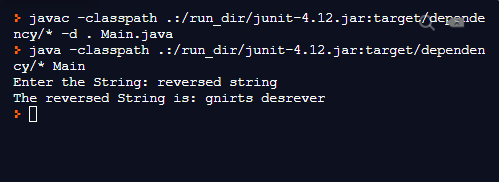
}

return (rev);

}

}

# OUTPUT:

****

**RESULT:**

Program to reverse a given string was implemented successfully.

# AIM:

1. **SECOND SMALLEST ELEMENT IN AN ARRAY**

Write a Java program to find the second smallest element in an array.

# ALGORITHM:

* 1. Declare a one-dimensional array and get the values form the user.
  2. Assume the first two elements as the smallest and the second smallest elements in the array.
  3. Traverse through the rest of the array and compare the elements with the smallest and the second smallest elements.
  4. If the current element is smaller than the smallest element, change the smallest element to the current element.
  5. Else, if the current element is larger than the smallest element, but smaller than the second smallest element, change the second smallest element to the current element.
  6. Display the second smallest element obtained after the iteration is complete.

# PROGRAM:

/\* Program to find the second smallest in an array \*/ class arrsmall

{

public static void main(String args[])

{

int a[] = {-22,59,12,334,2,-33,78,45,-67,500,-12};

System.out.println("The given array is: "); for(int i=0;i<a.length;i++)

System.out.print(a[i]+" ");

int s1,s2;//To store the smallest and the second smallest numbers respectively s1 = a[0];//assumption

s2 = a[1];

if(s2<s1)

{

s1 = a[1];

s2 = a[0];

}

for(int i=2;i<a.length;i++)

{

if(a[i]<=s1 && a[i]<s2)//ie a[i] is the smallest

{

s2 = s1; s1 = a[i];

}

else if(a[i]>s1 && a[i]<=s2)//ie a[i] is the second smallest s2 = a[i];

else

}

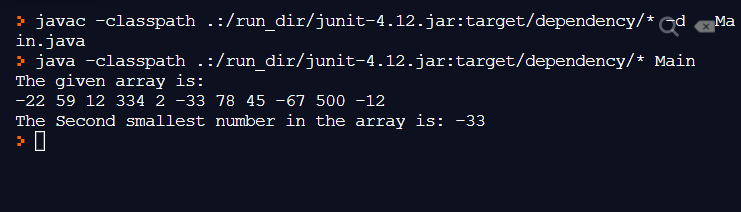
continue;

System.out.println("\nThe Second smallest number in the array is: " + s2);

}

}

# OUTPUT:

****

**RESULT:**

Program to find the second smallest element in an array was implemented successfully.

# AIM:

* 1. **CHECK WHETHER A NUMBER IS PRIME OR NOT**

Write a Java program to check whether a given number is prime or not.

# ALGORITHM:

* + 1. Get the number, n to be checked from the user.
    2. Iterate from 2 to n/2 and check whether those integers divide the number, n
    3. If a divisor is found at any point of the iteration, print “Not a prime number”.
    4. Otherwise, complete the iteration and then print “Prime number”.

# PROGRAM:

/\* Program to check whether a given number is prime or not \*/ import java.util.\*;

class prime

{

public static void main(String args[])

{

Scanner sc = new Scanner(System.in); System.out.print("Enter a number: "); int n = sc.nextInt();

if(n<0)

else

System.out.println("\nInvalid input: The number cannot be negative");

{ int flag = 0;

for(int i=2;i<n/2;i++)

{

if((n%i) == 0)

{

flag = 1; break;

}

}

number");

if(flag == 1)

System.out.println("The given number," + n + " is not a Prime

else

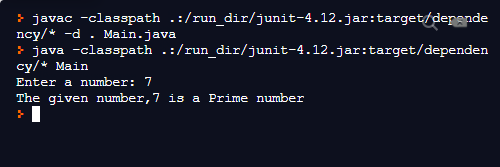
}

}

}

System.out.println("The given number," + n + " is a Prime number");

# OUTPUT:

****

**RESULT:**

Program to check whether a number is prime or not was implemented successfully.

# AIM:

* 1. **EXERCISE TO UNDERSTAND INHERITANCE**

Write a Java program which creates a class named 'Employee' having the following members: Name, Age, Phone number, Address, Salary. It also has a method named 'printSalary( )' which prints the salary of the Employee. Two classes 'Officer' and 'Manager' inherits the 'Employee' class. The 'Officer' and 'Manager' classes have data members 'specialization' and 'department' respectively. Now, assign name, age, phone number, address and salary to an officer and a manager by making an object of both of these classes and print the same.

# ALGORITHM:

* + 1. Create a class called ‘Employee’ with all the data members and the method ‘printSalary()’.
    2. Create two classes called ‘Officer’ and ‘Manager’, which inherit from ‘Employee’ and initialize the data members defined in the ‘Employee’ through the subclasses.
    3. Create a method to print the details of the Officer and the Manager. Call the ‘printSalary()’ method of the ‘Employee’ class to print the salaries.
    4. Create a main class and create the objects of both the ‘Officer’ and the ‘Manager’ classes and call the display methods of both.

# PROGRAM:

/\* Program to store the details of employees using Inheritance \*/ class Employee

{

String name,address,phone; double salary;

int age;

Employee(String n,String add,int a,double sal,String ph)

{

name = n; age = a;

address = add; salary = sal; phone = ph;

}

void printSalary()

{

System.out.println("Salary per annum: " + salary + "(AUD)");

}

}

class Officer extends Employee

{

String spec;

Officer(String n,String add,int a,double sal,String ph,String sp)

{

super(n,add,a,sal,ph); spec = sp;

}

void printDetails()

{

System.out.println("Details of Officer"); System.out.println("Name: " + name); System.out.println("Age: " + age); System.out.println("Address: " + address); System.out.println("Phone No.: " + phone); super.printSalary(); System.out.println("Specialization: " + spec);

}

}

class Manager extends Employee

{

String dept;

Manager(String n,String add,int a,double sal,String ph,String dp)

{

super(n,add,a,sal,ph);

dept = dp;

}

void printDetails()

{

System.out.println("Details of Manager"); System.out.println("Name: " + name); System.out.println("Age: " + age); System.out.println("Address: " + address); System.out.println("Phone No.: " + phone); super.printSalary(); System.out.println("Department: " + dept);

}

}

class empdetails

{

public static void main(String args[])

{

String name = "Charles Mays";

String ad = "25, Ocean Street, Sydney, Australia"; int age = 51;

String ph = "(02) 8905 1101";

double sal = 60000.00;

String spl = "MBA in Human Resources";

Officer emp1 = new Officer(name,ad,age,sal,ph,spl); emp1.printDetails();

name = "P. Sherman";

ad = "42, Wallaby Way, Sydney, Australia"; age = 38;

ph = "(02) 8571 0043";

sal = 120000.00;

String dep = "Sales";

Manager emp2 = new Manager(name,ad,age,sal,ph,dep); emp2.printDetails();

}

}

# OUTPUT:

****

**RESULT:**

The program to understand inheritance was successfully implemented.

# AIM:

* 1. **EXERCISE TO UNDERSTAND POLYMORPHISM**

Write a java program to create an abstract class named Shape that contains an empty method named numberOfSides( ). Provide three classes named Rectangle, Triangle and Hexagon such that each one of the classes extends the class Shape. Each one of the classes contains only the method numberOfSides( ) that shows the number of sides in the given geometrical structures.

# ALGORITHM:

* + 1. Create an abstract class named ‘Shape’ with and empty method ‘numberOfSides’.
    2. Create three classes named ‘Rectangle’, ‘Triangle’ and ‘Hexagon’ that inherit from the class ‘Shape’.
    3. Implement the numberOfSides() method in each of the sub-classes by printing the number of sides that the shape has. (4 for Rectangle, 3 for Triangle and 6 for Hexagon)
    4. Create a main class and create the objects of each of the subclasses.
    5. Call the numberOfSides() method of each class.

# PROGRAM:

/\*Program to implement polymorphism using an abstract class \*/ abstract class Shape {

abstract void numberOfSides();//abstract method

}

class Rectangle extends Shape { void numberOfSides() {

System.out.println("A Rectangle has 4 sides");

}

}

class Triangle extends Shape { void numberOfSides() {

System.out.println("A Triangle has 3 sides");

}

}

class Hexagon extends Shape { void numberOfSides() {

System.out.println("A Hexagon has 6 sides");

}

}

class shapedetails {

public static void main(String args[]) { Rectangle r1 = new Rectangle(); r1.numberOfSides();

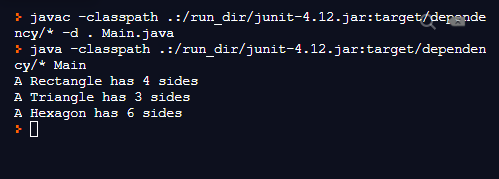
Triangle t1 = new Triangle(); t1.numberOfSides();

Hexagon h1 = new Hexagon(); h1.numberOfSides();

}

}

# OUTPUT:

****

**RESULT:**

The program to implement Polymorphism in Java was implemented successfully.

# AIM:

* 1. **AREA OF DIFFERENT SHAPES**

Write a Java program to calculate the area of different shapes namely circle, rectangle, and triangle using the concept of method overloading.

# ALGORITHM:

* + 1. Create a class with three methods such that the method name and return type, but with different parameter lists.
    2. Use the first one with just one ‘Double’ parameter to calculate the area of a circle with a given radius.
    3. Use the second one to calculate the area of a rectangle using two ‘Double’ parameters as the length and breadth.
    4. Use the third one to calculate the area of a triangle using three ‘Double’ parameters as the three sides of a triangle.
    5. Create a main class and an object of the other class.
    6. Call the three methods using their respective number of arguments.
    7. Print the area obtained.

# PROGRAM:

/\* Program to calculate the area of various shapes such as Circle rectangle and triangle using Method Overloading\*/

class Shapes

{

double area(double r)//For Area of a Circle

{

return (3.14\*r\*r);

}

double area(double a, double b)//For Area of a Rectangle

{

return (a\*b);

}

double area(double a, double b, double c)//For Area of a Triangle

{

double s = (a+b+c)/2;

return (Math.sqrt(s\*(s-a)\*(s-b)\*(s-c)));

}

}

class AreaOverload

{

public static void main(String args[])

{

Shapes s1 = new Shapes(); double a;

a = s1.area(10);//Area of circle

System.out.println("The area of the Circle with radius 10 is: " + a); a = s1.area(3.5,10);//Area of rectangle

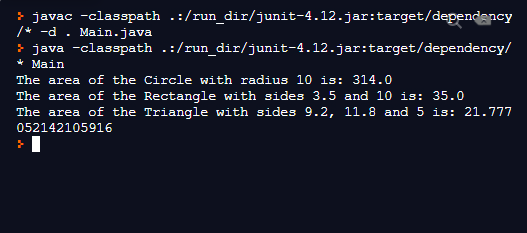
System.out.println("The area of the Rectangle with sides 3.5 and 10 is: " + a); a = s1.area(9.2,11.8,5);//Area of triangle

System.out.println("The area of the Triangle with sides 9.2, 11.8 and 5 is: " +a);

}

}

# OUTPUT:

****

**RESULT:**

The program to find the area of different shapes using the concept of method overloading was implemented successfully.

# AIM:

* 1. **TWO CLASSES WITH ONE OBJECT**

Write two Java classes Employee and Engineer. Engineer should inherit from Employee class. Employee class to have two methods display() and calcSalary(). Write a program to display the engineer salary and to display from Employee class using a single object instantiation (i.e., only one object creation is allowed).

* display() only prints the name of the class and does not return any value. Ex. “Name of class is Employee.”
* calcSalary() in Employee displays “Salary of employee is 10000” and calcSalary() in Engineer displays “Salary of employee is 20000

# ALGORITHM:

* + 1. Create a class called ‘Employee’ and another class called ‘Engineer’ such that ‘Engineer’ is a sub-class of ‘Employee’.
    2. Create two methods in ‘Employee’ class, one to display the name of the class as “Employee” and the other to print the salary of the employee as 10000.
    3. In the sub-class, ‘Engineer’, create a method to print the salary, with the same name as the one in the ‘Employee’ class.
    4. Call the calcSalary() method of the ‘Employee’ class from the one in the ‘Engineer’ class and then print the salary again as 20000.
    5. Create a main class and create an object of the sub-class, ‘Engineer’.
    6. Call the display() method of the super-class and the calcSalary() method of the sub-class to print all the required details.

# PROGRAM:

class Employee

{

void display()

{

System.out.println("Name of class is Employee");

}

void calcSalary()

{

System.out.println("Salary of Employee is 10000");

}

}

class Engineer extends Employee

{

void calcSalary()

{

super.calcSalary();

System.out.println("Salary of Employee is 20000");

}

}

class empengg

{

public static void main(String args[])

{

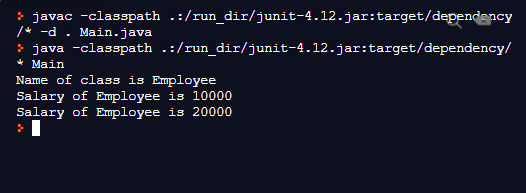
Engineer e1 = new Engineer(); e1.display();

e1.calcSalary();

}

}

# OUTPUT:

****

**RESULT:**

Program to use two classes with one object was implemented successfully.

# AIM:

* 1. **GARBAGE COLLECTION**

Write a Java program to demonstrate the use of garbage collector.

# ALGORITHM:

* + 1. Create a class with some arbitrary method and a finalize() method.
    2. In the finalize() method, print the details of the object which is going to be used by the garbage collector.
    3. Create a main class and the create an object of the other class.
    4. Call the arbitrary method defined, in order to show that the object is active.
    5. Nullify the object created.
    6. Call the garbage collector using “System.gc()” method.

# PROGRAM:

/\* Program to demonstrate garbage collection in java \*/

class Main{

public static class GarbageCollect{

public void finalize(){

System.out.println("Garbage Collected");

}

}

public static void main(String args[]){

GarbageCollect ob1=new GarbageCollect();

GarbageCollect ob2=new GarbageCollect();

ob1=null;

ob2=null;

System.gc();

}

}

# OUTPUT:



**RESULT:**

Program to demonstrate Garbage Collector in Java was implemented successfully.

# AIM:

* 1. **FILE HANDLING WITH READER/WRITER**

Write a file handling program in Java with reader/writer.

# ALGORITHM:

* + 1. Get the name of the file which is to be used, from the user.
    2. Open the file using a “FileWriter” object, by handling the exception that may occur.
    3. Get the string of text, line-by-line from the user and write them to the file, by handling the exception that may occur. Stop the process when the user inputs the word “Done”.
    4. Close the file by handling the exception that may occur.
    5. Open the file again, but using a “FileReader” object, by checking for exceptions that may occur.
    6. Read the text character-by-character until end of file is reached, checking for exceptions and print each character onto the console.
    7. Close the file by handling the exceptions that may occur.

# PROGRAM:

/\* Program for file handling in java \*/ import java.io.\*;

class filehandling

{

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

{

BufferedReader br = new BufferedReader(new InputStreamReader(System.in)); FileWriter fw;

char ch;

String s;

System.out.print("Enter the name of the file to which the content is to be written: "); String filename = br.readLine();

//Attempting to open/create the file try {

fw = new FileWriter(filename);

} catch(FileNotFoundException e) { System.out.println("Cannot open the file: " + e); return;

}

//Now the file has been opened and contents can be written

System.out.println("Enter the contents of the file. Enter \"Done\" to stop."); try {

do {

s = br.readLine(); if(s.equals("Done"))

break;

else

{

for(int i=0;i<s.length();i++)

{

ch = s.charAt(i); fw.write(ch);

}

fw.write("\n");//For newline

}

}while(true);

} catch(IOException e) {

System.out.println("Error Writing file " + e);

}

//Now we can close the file

try {

fw.close();

} catch(IOException e) {

System.out.println("Error closing the file " + e);

}

//Now the file has been closed

//It can now be used for reading FileReader fr;

try {

fr = new FileReader(filename);

} catch(FileNotFoundException e) {

System.out.println("Error opening file for reading: " + e); return;

}

//Now the file has been opened for reading int a;

System.out.println("\nThe contents stored in " + filename + " are:"); try {

do {

a = fr.read();

if(a != -1)

System.out.print((char)a);

}while(a != -1);

} catch(IOException e) {

System.out.println("Error reading the file: " + e);

}

//Now the file can be closed try {

fr.close();

} catch(IOException e) {

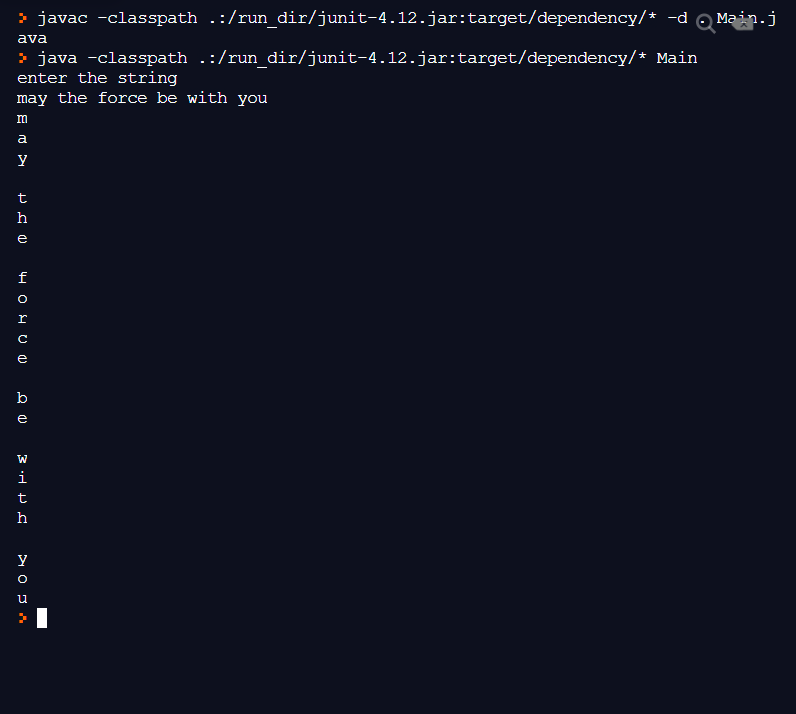
System.out.println("Error closing the file " + e);

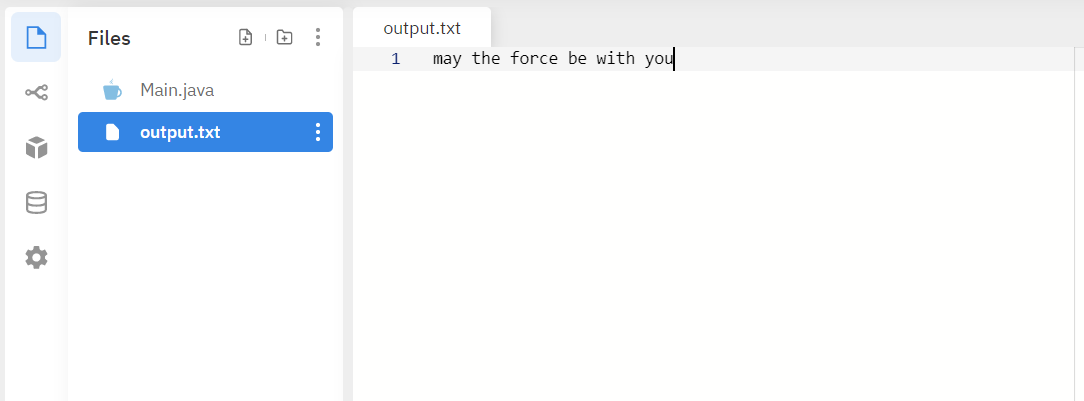
}

}

}

# OUTPUT:

****

****

**RESULT:**

The program for file handling using Reader/Writer classes was implemented successfully.

# AIM:

* 1. **FILE HANDLING WITH EXCEPTIONS**

Write a Java program that read from a file and write to file by handling all file related exceptions.

# ALGORITHM:

* + 1. Get the name of the files which are to be used, from the user.
    2. Open the file to be read using a “FileInputStream” object, by handling the FileNotFoundException that may occur.
    3. Open the file to which the content should be written using a “FileOutputStream” object, by checking for the FileNotFoundException that may occur.
    4. Get the string of text, character-by-character from the input file and write it to the output file, considering the IOException that may occur.
    5. Close both the files by handling the IOException that may occur.

# PROGRAM:

// Program to read from a file and write to a file by handling all the exceptions

import java.io.\*;

public class Main

{

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

{

int i;

FileReader fin = null;

FileWriter fout=null;

try

{

fin = new FileReader("output.txt");

fout=new FileWriter("copy.txt");

}

catch (IOException e)

{

System.out.println("cant read the file");

}

try{System.out.println("content of output.txt");

do

{

i= fin.read();

if(i != -1)

{

fout.write((char)i);

System.out.print((char)i);

}

}while(i != -1);

}

catch(IOException e)

{

System.out.println("error in reading file or writting file");

}

fin.close();

fout.close();

FileReader fcr= null;

try

{

fcr = new FileReader("copy.txt");

}

catch (IOException e)

{

System.out.println("cant read the file");

}

try

{ System.out.println("\ncontent of copy.txtfile ");

do

{

i= fcr.read();

if(i != -1)

{

System.out.print((char)i);

}

}while(i != -1);

}

catch(IOException e)

{

System.out.println("error in reading file");

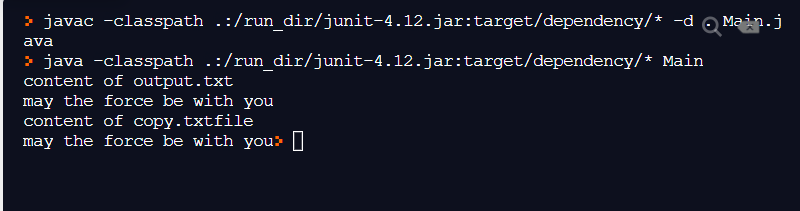
}

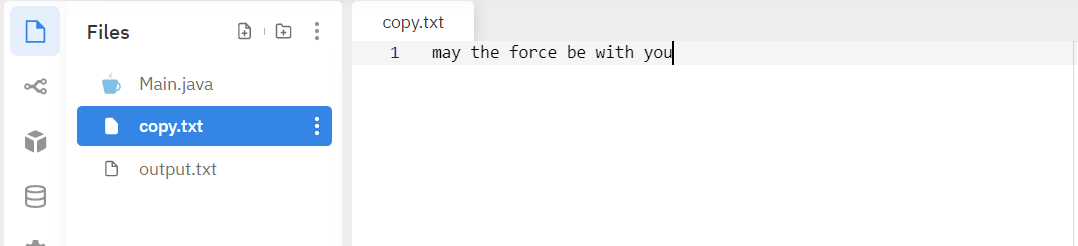
fcr.close();

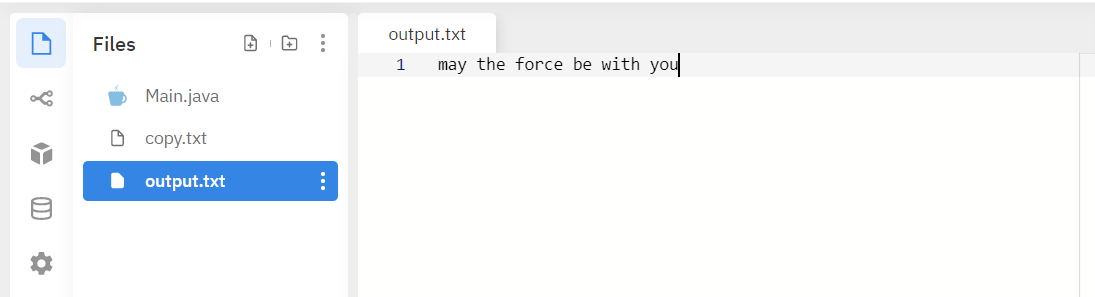
}

}

# OUTPUT:

****

****

****

**RESULT:**

The file handling program was implemented successfully with exception handling.

# AIM:

* 1. **USAGE OF TRY, CATCH, THROWS AND FINALLY**

Write a Java program that shows the usage of try, catch, throws and finally.

# ALGORITHM:

* + 1. Declare a method with the ‘throws’ keyword and throw an exception manually from it.
    2. Declare a method which contains a try block with a return statement and a finally block.
    3. Declare a method which contains a try-catch block, such that an exception will occur in the try block.
    4. Create a main() method to call all the other methods.

# PROGRAM:

/\* Program to show the use of try, catch, throws and finally \*/

class Main {

public static void main(String args[]) throws ArithmeticException {

int op1, op2;

try {

System.out.println("Start of add");

op1 = 0;

op2 = 25 / op1;

System.out.println(op2);

System.out.println("End of add");

}

catch (ArithmeticException e) {

System.out.println("Divide by Zero Error!!");

}

finally

{

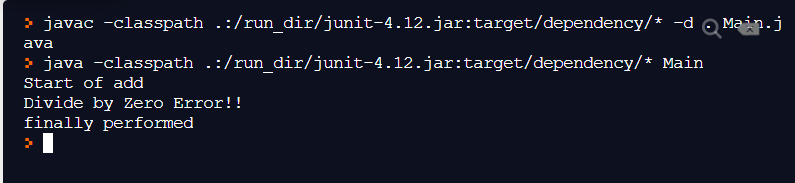
System.out.println("finally performed");

}

}

}

# OUTPUT:

****

**RESULT:**

The program to demonstrate the usage of try, catch, throws and finally keywords was implemented successfully.

# AIM:

* 1. **MULTIPLE THREADS**

Write a Java program to create two threads: One for displaying all odd number between 1 and 100 and second thread for displaying all even numbers between 1 and 100.

# ALGORITHM:

* + 1. Create two classes which implement the Runnable interface.
    2. Use one class to print the odd numbers in its run() method and the other to print even numbers in its run() method.
    3. Use sleep() method in both the run() methods so that the threads could execute one after the other.
    4. Create the main class and create the Thread objects.
    5. Start the two threads using the start() method.

# PROGRAM:

/\* Program to create two threads: One for displaying all odd number between 1 and 100 and second thread for displaying all even numbers between 1 and 100. \*/

class OddThread implements Runnable

{

    public void run()

    {

        try

        {

            for(int i=1;i<100;i=i+2)

            {

                System.out.print("Odd: " + i + " ");

                Thread.sleep(500);

            }

        } catch(InterruptedException e) {

            System.out.println("Odd Thread Interrupted "+e);

        }

        System.out.println("\nExiting Odd Thread");

    }

}

class EvenThread implements Runnable

{

    public void run()

    {

        try

        {

            for(int i=2;i<=100;i=i+2)

            {

                System.out.print("Even: " + i + " ");

                Thread.sleep(500);

            }

        } catch(InterruptedException e) {

            System.out.println("Even Thread Interrupted "+e);

        }

        System.out.println("\nExiting Even Thread");

    }

}

class Main

{

    public static void main(String args[])

    {

        Thread t1 = new Thread(new OddThread(), "odd");

        Thread t2 = new Thread(new EvenThread(), "even");

        t2.start();

        t1.start();

        try

        {

            Thread.sleep(40500);

        }catch(InterruptedException e){

            System.out.println("main Thread Interrupted "+e);

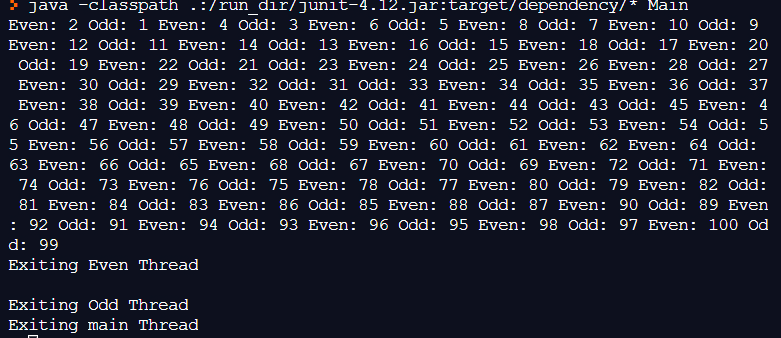
        }

        System.out.println("Exiting main Thread");

    }

}

# OUTPUT:

****

**RESULT:**

Program to implement two threads to print the odd and even numbers was implemented successfully.

# AIM:

* 1. **THREAD SYNCHRONIZATION**

Write a Java program that shows thread synchronization.

# ALGORITHM:

* + 1. Create a class with only one method which prints the natural number up to n using the ‘synchronized’ keyword.
    2. Create two classes the implement the ‘Runnable’ interface and calls the method to generate the natural numbers.
    3. Create the main class which contains the main() method and create one only object of the class which contains the method which is being called.
    4. Create the two threads by passing the above created object and start the threads.

# PROGRAM:

/\* Program to demonstrate Thread synchronization in Java \*/

class SquareOfNumber{

synchronized void printSquareOfNumber(int n){

for(int i=1;i<=10;i++){

System.out.println(n\*i);

try{

Thread.sleep(1000);

}catch(Exception e){System.out.println(e);}

}

}

}

class callingFunc1 implements Runnable{

SquareOfNumber t;

callingFunc1(SquareOfNumber t){

this.t=t;

}

public void run(){

t.printSquareOfNumber(5);

}

}

class callingFunc2 implements Runnable{

SquareOfNumber t;

callingFunc2(SquareOfNumber t){

this.t=t;

}

public void run(){

t.printSquareOfNumber(100);

}

}

class Main{

public static void main(String args[]){

SquareOfNumber obj = new SquareOfNumber();

Thread t1=new Thread(new callingFunc1(obj), "five");

Thread t2=new Thread(new callingFunc2(obj), "Hundred");

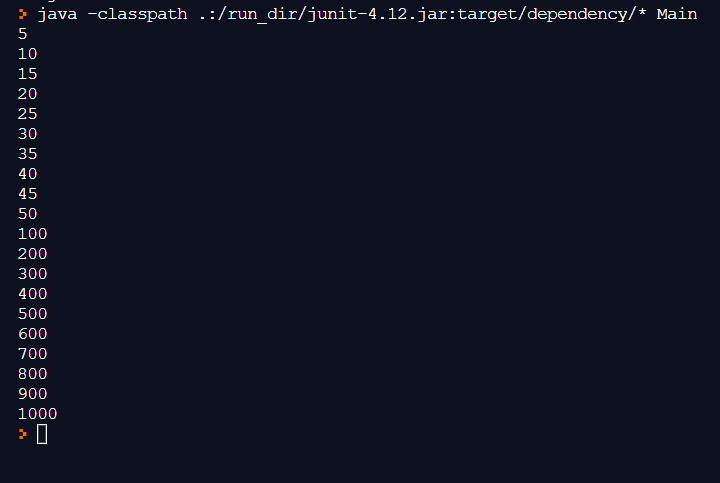
t1.start();

t2.start();

}

}

# OUTPUT:



**RESULT:**

The program to implement thread synchronization in Java was implemented successfully.

# AIM:

* 1. **DOUBLY LINKED LIST**

Write a Java program for the following:

1. Create a doubly linked list of elements.
2. Delete a given element from the above list.
3. Display the contents of the list after deletion

# ALGORITHM:

* 1. Insert the first element of the linked list.
  2. Ask the user if more nodes are required. If yes, insert the elements to the end of the linked list.
  3. After all the inputs are done, ask the user for the element to be deleted.
  4. Check whether the element is present in the linked list.
  5. If yes, delete the node containing that element. Otherwise, print an appropriate message.
  6. Display the elements left in the linked list.

# PROGRAM:

/\* Program to implement a Doubly Linked List in Java \*/

import java.io.\*;

public class Main {

class Node

{

int data;

Node previous;

Node next;

public Node(int data)

{

this.data = data;

}

}

Node head, tail = null;

public void addNode(int data) {

Node newNode = new Node(data);

if(head == null) {

head = tail = newNode;

head.previous = null;

tail.next = null;

}

else {

tail.next = newNode;

newNode.previous = tail;

tail = newNode;

tail.next = null;

}

}

public void display() {

Node current = head;

if(head == null) {

System.out.println("List is empty");

return;

}

System.out.println("Nodes of doubly linked list: ");

while(current != null) {

System.out.print(current.data + " ");

current = current.next;

}

System.out.println("\n");

}

public void delete(int data)

{

Node current = head;

if(head == null) {

System.out.println("List is empty");

return;

}

while(current != null)

{

if(current.data==data)

break;

current=current.next;

}

if(current==head)

{

head=current.next;

current.next.previous=null;

}

else

{ if(current.next==null)

{

tail=current.previous;

current.previous.next=null;

}

else

{

current.next.previous=current.previous;

current.previous.next=current.next;

}

}

}

public static void main(String[] args) throws NumberFormatException, IOException {

int num,cho;

doublink dList = new doublink();

BufferedReader br=new BufferedReader (new InputStreamReader(System.in));

do {

System.out.println(" OPTIONS: 1. insert\t2. Delete \t3.display\t4.exit\nenter the choice: ");

cho=Integer.parseInt(br.readLine()) ;

switch (cho) {

case 1:

{

System.out.println("Enter the integerto be inserted: ");

num=Integer.parseInt(br.readLine()) ;

dList.addNode(num);

break;

}

case 2:

{

System.out.println("Enter the integer to be deleted: ");

num=Integer.parseInt(br.readLine()) ;

dList.delete(num);

break;

}

case 3:

{

dList.display();

break;

}

case 4:

{

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

System.exit(0);

}

default:

{ System.out.println("wrong choice");

break;

}

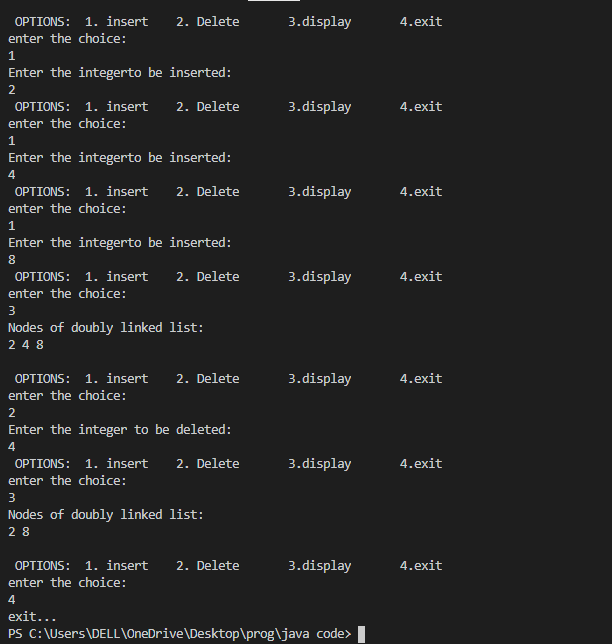
}

} while (true);

}

}

# OUTPUT:



**RESULT:**

The program to implement Doubly Linked List in Java was implemented successfully.