MRCET CAMPUS

MALLA REDDY COLLEGE OF ENGINEERING & TECHNOLOGY

(AUTONOMOUS INSTITUTION – UGC, GOVT. OF INDIA)

Department of CSE

(Emerging Technologies)

Data Science& Cyber Security

B.TECH(R-20 Regulation)

(II YEAR – II SEM)

(2024-25)

Object Oriented Programming through JAVA LAB

(R20A0585)

LAB MANUAL

MALLA REDDY COLLEGE OF ENGINEERING & TECHNOLOGY

(Autonomous Institution – UGC, Govt. of India)

Recognized under 2(f) and 12(B) of UGC ACT 1956

(Affiliated to JNTUH, Hyderabad, Approved by AICTE-Accredited by NBA & NAAC – 'A' Grade - ISO 9001:2015 Certified)

Maisammaguda, Dhulapally (Post Via. Hakimpet), Secunderabad–500100, Telangana State, India

Department of Computer Science and Engineering

EMERGING TECHNOLOGIES

Object Oriented Programming through JAVA LAB

(R20A0585)

LAB MANUAL

Prepared by

D KALPANA, ASST. PROF

**(First Time Prepared Faculty)*

On

20.02.2022

Updated by

P.LAVANYA, ASST. PROF

***(Second-time Prepared Faculty if updated the existed Lecture Notes)*

On

02.01.2025

Department of Computer Science and Engineering

EMERGING TECHNOLOGIES

Vision

- "To be at the forefront of Emerging Technologies and to evolve as a Centre of Excellence in Research, Learning and Consultancy to foster the students into globally competent professionals useful to the Society."

Mission

The department of CSE (Emerging Technologies) is committed to:

- To offer highest Professional and Academic Standards in terms of Personal growth and satisfaction.
- Make the society as the hub of emerging technologies and thereby capture opportunities in new age technologies.
- To create a benchmark in the areas of Research, Education and Public Outreach.
- To provide students a platform where independent learning and scientific study are encouraged with emphasis on latest engineering techniques.

QUALITY POLICY

- To pursue continual improvement of teaching learning process of Undergraduate and Post Graduate programs in Engineering & Management vigorously.
- To provide state of art infrastructure and expertise to impart the quality education and research environment to students for a complete learning experiences.
- Developing students with a disciplined and integrated personality.
- To offer quality relevant and cost effective programmes to produce engineers as per requirements of the industry need.

For more information: www.mrcet.ac.in

SYLLABUS

M R C E T CAMPUS | AUTONOMOUS INSTITUTION - UGC, GOVT. OF INDIA

II Year B.Tech. CSE (CYS & DS) - II Sem **L/T/P/C**
0/-/3/1.5

(R20A0585) OBJECT ORIENTED PROGRAMMING THROUGH JAVA LAB

COURSE OBJECTIVES:

1. To prepare students to become familiar with the Standard Java technologies ofJ2SE

2. To prepare students to excel in Object Oriented programming and to succeed as a Java Developer through global rigorous education.

3. To provide Students with a solid foundation in OOP fundamentals required to solve programming problems and also to learn Advanced Java topics like J2ME, J2EE, JSP, JavaScript

4. To inculcate in students professional and ethical attitude, multidisciplinary approach and an ability to relate Java programming issues to broader application context.

5. To provide student with an academic environment aware of excellence, written ethical codes and guidelines and lifelong learning needed for a successful professional career.

Week 1:

a) Write a java program to find the Fibonacci series using recursive and non-recursive functions

b) Write a program to multiply two given matrices.

c) Write a program for Method overloading and Constructor overloading

Week 2:

a) Write a program to demonstrate execution of static blocks ,static variables & static methods.

b) Write a program to display the employee details using Scanner class

c) Write a program for sorting a given list of names in ascending order

Week 3:

a) Write a program to implement single and Multi level inheritance

b) Write a program to implement Hierarchical Inheritance.

c) Write a program to implement method overriding.

Week 4:

a) Write a program to create an abstract class named Shape that contains two integers and an empty method named printArea (). 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 printArea () that prints the area of the given shape.

b) Write a program to implement Interface.

c) Write a program to implement multiple and Hybrid Inheritance

Week 5:

a) Write a program to create inner classes

b) Write a program to create user defined package and demonstrate various access modifiers.

c) Write a program to demonstrate the use of super and final keywords.

Week 6:

a) Write a program if number is less than 10 and greater than 50 it generate the exception out of range. else it displays the square of number.

b) Write a program with multiple catch Statements.

c) Write a program to implement nested try

Week 7:

a) Write a Program to implement simple Thread by extending Thread class and implementing runnable interface.

b) Write a program that implements a multi-thread application that has three threads

c) Write a program to set and print thread priorities

Week 8:

Write a program to implement following collections

a) array List b) Vector

c) Hash table d) Stack

Week 9:

a) Write a program to demonstrate lambda expressions.

b) Write a program for producer and consumer problem using Threads

Week 10:

a) Write a program to list all the files in a directory including the files present in all its subdirectories.

b) Write a Program to Read the Content of a File Line by Line

Week 11:

a) Write a program that connects to a database using JDBC display all records in a table.

b) Write a program to connect to a database using JDBC and insert values into it.

c) Write a program to connect to a database using JDBC and delete values from it

Week 12:

Write a program that works as a simple calculator. Use a Grid Layout to arrange

Buttons for digits and for the + - * % operations. Add a text field to display the result.

COURSE OUTCOMES:

Upon successful completion of this course, the students will be able to:

1. Analyze the necessity for Object Oriented Programming paradigm and over structured programming and become familiar with the fundamental concepts in OOP.

2. Demonstrate an ability to design and develop Java programs, analyze, and interpret object oriented data and report results.

3. Analyze the distinguish between various types of inheritance.

4. Demonstrate an ability to design an object oriented system, AWT components or multithreaded process as per needs and specifications.

5. Demonstrate an ability to visualize and work on laboratory and multidisciplinary tasks like console and windows applications for standalone programs.

INDEX

WEEK -1(A) **Date:**

Aim: Write a java program to find the Fibonacci series using recursive and non recursive functions

Program:

```
class Fibonascci
{
        int a,b,c;
void nonrecursive(int n)        //Non recursive function to find the Fibonacci series.
{
        a=0; b=1;
        System.out.print(a+ "" + b);
        c=a+b;
while(c<=n)
{
System.out.print(c); a=b;
        b=c;
        c=a+b;
}
}
int recursive(int n)      // Recursive function to find the Fibonacci series.
{
        if(n==0)
                return (0);
        if(n==1)
                return (1);
        else
        return(recursive(n-1)+recursive(n-2));
}
}
class Fib1
{
        public static void main(String args[])
        {
        int n=5;
        System.out.println("The Fibonacci series using non recursive is");
        // Creating object for the fib class.
         Fibonascci f=new Fibonascci();
        f.nonrecursive(n);                          // Calling non recursive function oF fib class.
        System.out.println("\n The Fibonacci series using recursive is");
        for(int i=0;i<=n;i++)
        {
        int F1=f.recursive(i);              // Calling recursive function of fib class.
        System.out.print(F1);
        }
        }
}
```

Three test outputs:

WEEK -1(B) **Date**:

Aim: Write a java program to multiply two given matrices.

Program:

```
public class MatrixEx
{
public static void main(String args[])
{
//creating two matrices
int a[][]={{1,1,1},{2,2,2},{3,3,3}};
int b[][]={{1,1,1},{2,2,2},{3,3,3}};

//creating another matrix to store the multiplication of two matrices int c[][]=new int[3][3];
        //3 rows and 3 columns

//multiplying and printing multiplication of 2 matrices for(int i=0;i<3;i++)
{
for(int j=0;j<3;j++)
{ c[i][j]=0;
for(int k=0;k<3;k++)
{
c[i][j]+=a[i][k]*b[k][j];
}       //end of k loop
System.out.print(c[i][j]+" "); //printing matrix element
}       //end of j loop System.out.println(); //new line
}
}
}
```

Three test outputs:

Signature of the faculty

WEEK : 1(C) **Date:**

Aim: Write a java program for Method overloading and Constructor overloading Method overloading:

Program:

Method overloading:

```
import java.io.*;
class MethodOverloadingEx
{
static int add(int a, int b)
{
return a + b;
}

static int add(int a, int b, int c)
{
return a + b + c;
}

public static void main(String args[])
{
System.out.println("add() with 2 parameters");
System.out.println(add(4, 6));

System.out.println("add() with 3 parameters");
System.out.println(add(4, 6, 7));
}
}
```

Output:

Constructor overloading

```
public class Student {
//instance variables of the class
int id;
String name;
Student()
{
System.out.println("this a default constructor");
}
Student(int i, String n)
{
id = i;
name = n;
}
public static void main(String[] args) {
//object creation
Student s = new Student();
System.out.println("\nDefault Constructor values: \n");
System.out.println("Student Id : "+s.id + "\nStudent Name : "+s.name);

System.out.println("\nParameterized Constructor values: \n");
Student student = new Student(10, "Kalpana");
System.out.println("Student Id : "+student.id + "\nStudent Name : "+student.name);
}
}
```

Three test outputs:

Signature of the faculty

EXERCISE:

1. Write a java program to find all even and odd integers up to a given integer.
2. Write a java program that reads a line of integers and displays each integers and the product of all integers use String Tokenizer.

WEEK: 2(A) **Date:**

Aim: Write a JAVA program to demonstrate execution of static blocks ,static variables & static methods.

Program:

```
public class StaticDemo
 {
   static int x = 10;
   static int y;

   static void func(int z) {
      System.out.println("x = " + x);
      System.out.println("y = " + y);
      System.out.println("z = " + z);
   }

   static {
      System.out.println("Running static initialization block.");
      y = x + 5;
   }

   public static void main(String args[]) {
      func(8);
   }
}
```

Three test Outputs:

WEEK: 2(B) **Date:**

Aim: Write a JAVA program to display the employee details using Scanner class
Program:

```
import java.util.*;

class EmployeeDetails
{
public static void main(String args[])
{
System.out.println("enter name,id,age,salary");
Scanner sc=new Scanner(System.in);
String n=sc.next();
int i=sc.nextInt();
int a=sc.nextInt();
float s=sc.nextFloat();

System.out.println("name is"+n+"idis"+i+"ageis"+a+"salaryis"+s);
}
}
```

Three test Outputs:

WEEK: 2(C) **Dates:**

Aim: Write a JAVA program for sorting a given list of names in ascending order
Program:

```
// Java Program to Sort Names in an ascending Order
import java.io.*;

class AscendingNames {
	public static void main(String[] args)
	{
		// storing input in variable
		int n = 4;

		// create string array called names
		String names[]
			= { "Rani", "Akshaya", "Guru", "Roshan" };
		String temp;
		for (int i = 0; i < n; i++)
		{
			for (int j = i + 1; j < n; j++)
			{

				// to compare one string with other strings
				if (names[i].compareTo(names[j]) > 0)
				{
					// swapping
					temp = names[i];
					names[i] = names[j];
					names[j] = temp;
				}
			}
		}

		// print output array
		System.out.println(
			"The names in alphabetical order are: ");
		for (int i = 0; i < n; i++)
			{
			System.out.println(names[i]);
			 }
	}
}
```

Three test Outputs:

Signature of the faculty

WEEK: 3(A) **Date:**

Aim: Write a java program to implement single and Multi level inheritance
Program:
Single inheritance:

```
class Employee
{
 float salary=40000;
}
class Programmer extends Employee
{
        int bonus=10000;
        public static void main(String args[])
        {
                Programmer p=new Programmer();
               System.out.println("Programmer salary is:"+p.salary);
               System.out.println("Bonus of Programmer is:"+p.bonus);
        }
}
```

Three test Outputs:

Signature of the faculty

Multi level inheritance:

```
class Shape {
  public void display() {
    System.out.println("Inside display");
  }
}
class Rectangle extends Shape {
  public void area() {
    System.out.println("Inside  area");
  }
}
class Cube extends Rectangle {
  public void volume() {
    System.out.println("Inside  volume");
  }
}
public class Tester {
  public static void main(String[] arguments) {
    Cube cube = new Cube();
    cube.display();
    cube.area();
    cube.volume();
  }
}
```

Three test Outputs:

Signature of the faculty

WEEK: 3(B) **Date:**

Aim: Write a JAVA program to implement Hierarchical Inheritance

Program:

```
class A {
	public void print_A() { System.out.println("Class A"); }
}

class B extends A {
	public void print_B() { System.out.println("Class B"); }
}

class C extends A {
	public void print_C() { System.out.println("Class C"); }
}

class D extends A {
	public void print_D() { System.out.println("Class D"); }
}

// Driver Class
public class Test {
	public static void main(String[] args)
	{
		B obj_B = new B();
		obj_B.print_A();
		obj_B.print_B();

		C obj_C = new C();
		obj_C.print_A();
		obj_C.print_C();

		D obj_D = new D();
		obj_D.print_A();
		obj_D.print_D();
	}
}
```

Three test Outputs:

Signature of the faculty

EXERCISE:

WEEK: 3(C) **Date:**

AIM: Write aJAVA program to implement method overriding
Program:

Signature of the faculty

WEEK: 4(A) **Date:**

AIM: Write a JAVA program to create an abstract class named Shape that contains two integers and an empty method named printArea (). 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 printArea () that prints the area of the given shape.

Program:

```
package project1;
import java.util.*;
public class AbstactDDemo
{
public static void main(String[] args)
{
Rectangle r=new Rectangle(); r.area(2,5);
Circle c=new Circle(); c.area(5,5);
Triangle t=new Triangle(); t.area(2,5);
}
}

abstract class shape
{
int x,y;
abstract void area(double x,double y);
}

class Rectangle extends shape
{
void area(double x,double y)
{
System.out.println("Area of rectangle is :"+(x*y));
}
}

class Circle extends shape
{
void area(double x,double y)
{
System.out.println("Area of circle is :"+(3.14*x*x));
}
```

```
}

class Triangle extends shape
{
void area(double x,double y)
{
System.out.println("Area of triangle is :"+(0.5*x*y));
}
}
```

OUTPUT:

Signature of the faculty

WEEK: 4(B) **Date:**

AIM: Write a JAVA program to implement Interface.

Program:

```
interface Bank{
float rateOfInterest();
}
class SBI implements Bank{
public float rateOfInterest()
{
return 9.15f;
}
}
class PNB implements Bank{
public float rateOfInterest(){
return 9.7f;
}
}
class TestInterface2{
public static void main(String[] args){
Bank b=new SBI();
System.out.println("ROI: "+b.rateOfInterest());
}}
```

OUTPUT:

Signature of the faculty

EXERCISE:

WEEK: 4(C) **Date:**

AIM: Write a JAVA program to implement multiple and Hybrid Inheritance

WEEK: 5(A) **Date:**

AIM: Write a JAVA program to create inner classes

Program:

```
class Outer_Demo {
  int num;

  // inner class
  private class Inner_Demo {
    public void print() {
      System.out.println("This is an inner class");
    }
  }

  // Accessing he inner class from the method within
  void display_Inner() {
    Inner_Demo inner = new Inner_Demo();
    inner.print();
  }
}

public class My_class {

  public static void main(String args[]) {
    // Instantiating the outer class
    Outer_Demo outer = new Outer_Demo();

    // Accessing the display_Inner() method.
    outer.display_Inner();
  }
}
```

OUTPUT:

Signature of the faculty

WEEK: 5(B) **Date:**

AIM: Write a JAVA program to create user defined package and demonstrate various access modifiers.

Program:

```
// private access modifier
class A{
private int data=40;
private void msg(){System.out.println("Hello java");}
}
  public class Simple{
 public static void main(String args[]){
   A obj=new A();
   System.out.println(obj.data);//Compile Time Error
   obj.msg();//Compile Time Error
   }
}
```

OUTPUT:

//default access modifier

```
//save by A.java
package pack;
class A{
  void msg(){System.out.println("Hello");}
}

//save by B.java
package mypack;
import pack.*;

class B{
  public static void main(String args[]){
   A obj = new A();//Compile Time Error
   obj.msg();//Compile Time Error
  }
}
```

OUTPUT:

// protected access modifier

```
//save by A.java
package pack;
public class A{
protected void msg(){System.out.println("Hello");}
}

//save by B.java
package mypack;
import pack.*;

class B extends A{
  public static void main(String args[]){
   B obj = new B();
   obj.msg();
  }
}
```

OUTPUT:

// public access modifier

```
//save by A.java
package pack;
public class A{
public void msg()
{
System.out.println("Hello");}
}

//save by B.java

package mypack;
import pack.*;

class B{
  public static void main(String args[])
{
   A obj = new A();
   obj.msg();
  }
}
```

OUTPUT:

Signature of the faculty

EXERCISE:

WEEK: 5(C) **Date:**

AIM: Write a program to demonstrate the use of super and final keywords.

Program:

WEEK: 6(A) **Date:**

AIM: Write a java program with multiple catch Statements

Program:

```
public class MultiCatchEx1
{
public static void main(String[] args)
{
try
{
 int arr[] = new int[6];
 arr[3] = 20/0; // Exception occurred.
 System.out.println("I am in try block");
}
catch(ArithmeticException  ae)
{
 System.out.println("A number cannot be divided by zero, Illegal operation in java");
}
catch(ArrayIndexOutOfBoundsException  e)
{
 System.out.println("Accessing array element outside of specified limit");
}
catch(Exception e)
{
 System.out.println(e.getMessage());
}
System.out.println("I am out of try-catch block");
 }
}
```

OUTPUT:

Signature of the faculty

WEEK: 6(B) **Date:**

AIM: Write a JAVA program to implement nested try.

Program:

```
public class NestedTryBlock {

   public static void main(String args[])
   {
      // outer (main) try block
      try {

         //inner try block 1
         try {

            // inner try block 2
            try {
               int arr[] = { 1, 2, 3, 4 };

               //printing the array element out of its bounds
               System.out.println(arr[10]);
            }

            // to handles ArithmeticException
            catch (ArithmeticException e) {
               System.out.println("Arithmetic exception");
               System.out.println(" inner try block 2");
            }
         }

         // to handle ArithmeticException
         catch (ArithmeticException e) {
            System.out.println("Arithmetic exception");
            System.out.println("inner try block 1");
         }
      }
```

```
// to handle ArrayIndexOutOfBoundsException
        catch (ArrayIndexOutOfBoundsException e4) {
            System.out.print(e4);
            System.out.println(" outer (main) try block");
        }
        catch (Exception e5) {
            System.out.print("Exception");
            System.out.println(" handled in main try-block");
        }
    }
}
```

OUTPUT:

Signature of the faculty

EXERCISE:

WEEK: 6(C) **Date:**

AIM: Write a JAVA program if number is less than 10 and greater than 50 it generate the exception out of range. else it displays the square of number..

Program:

WEEK: 7(A) **Date:**

AIM: Write a JAVA program that implements a multi-thread application that has three threads

Program:

```
import java.util.Random;
class Square extends Thread
{
          int x;
         Square(int n)
          {
          x = n;
          }
 public void run()
 {
         int sqr = x * x;
          System.out.println("Square of " + x + " = " + sqr );
 }
}

class Cube extends Thread
{
 int x;
          Cube(int n)
          {
         x = n;
          }
 public void run()
 {
          int cub = x * x * x;
          System.out.println("Cube of " + x + " = " + cub );
 }
}

class Number extends Thread
{
 public void run()
 {
         Random random = new Random();
         for(int i =0; i<10; i++){
```

```
            int randomInteger = random.nextInt(100);
            System.out.println("Random Integer generated : " + randomInteger);
            Square s = new Square(randomInteger);
            s.start();
 Cube c = new Cube(randomInteger);
            c.start();
 try {
           Thread.sleep(1000);
```

//This thread generates random number 10 times between 1 to 100 for every 1 second. The generated random number is then passed as argument to Square and Cube threads.
Output varies each time a program is executed.

```
}

catch (InterruptedException ex)
   {
           System.out.println(ex);
   }
 }
}
}

public class Threading {
 public static void main(String args[])
 {
 Number n = new Number();
 n.start();
 }
}
```

Three test outputs:

Signature of the faculty

WEEK: 7(B) **Date:**

AIM: Write a JAVA program to set and print thread priorities

Program:

```
import java.lang.*;

class ThreadDemo extends Thread {

        public void run()
        {
                System.out.println("Inside run method");
        }
        public static void main(String[] args)
        {
                ThreadDemo t1 = new ThreadDemo();
                ThreadDemo t2 = new ThreadDemo();
                ThreadDemo t3 = new ThreadDemo();
        System.out.println("t1 thread priority : "+ t1.getPriority());
        System.out.println("t2 thread priority : "+ t2.getPriority());
        System.out.println("t3 thread priority : "+ t3.getPriority());
                // Setting priorities of above threads by passing integer arguments
                t1.setPriority(2);
                t2.setPriority(5);
                t3.setPriority(8);          // t3.setPriority(21); will throw IllegalArgumentException

                System.out.println("t1 thread priority : "+ t1.getPriority());
                System.out.println("t2 thread priority : "+ t2.getPriority());
                System.out.println("t3 thread priority : "+ t3.getPriority());

        System.out.println("Currently Executing Thread : "+ Thread.currentThread().getName());
        System.out.println("Main thread priority : "+ Thread.currentThread().getPriority());

        Thread.currentThread().setPriority(10);
        System.out.println("Main thread priority : "+ Thread.currentThread().getPriority());
        }
}
```

OUTPUT:

Signature of the faculty

EXERCISE:

WEEK: 7(C) **Date:**

AIM: Write a Program to implement simple Thread by extending Thread class and implementing runnable interface.

Program:

WEEK: 8(A) **Date:**

AIM: Write a JAVA program to implement following collections

a)array List b) Vector

c)Hash table d)Stack

Program:

a) array List

```
import java.util.*;
 public class ArrayListExample1{
 public static void main(String args[]){
  ArrayList<String> list=new ArrayList<String>();//Creating arraylist
      list.add("Mango");//Adding object in arraylist
      list.add("Apple");
      list.add("Banana");
      list.add("Grapes");
      //Printing the arraylist object
      System.out.println(list);
 }
}
```

OUTPUT:

Signature of the faculty

b) Vector

```
import java.io.*;
import java.util.*;

class  VectorDemo {

		public static void main(String[] args)
	{
		int n = 5;
		Vector<Integer> v = new Vector<Integer>(n);
		for (int i = 1; i <= n; i++)
			v.add(i);
			System.out.println(v);
			v.remove(3);
			System.out.println(v);
		for (int i = 0; i < v.size(); i++)
			System.out.print(v.get(i) + " ");
	}
}
```

OUTPUT:

Signature of the faculty

EXERCISE:

WEEK: 8(B) **Date:**

AIM: Write a JAVA program to implement following collections

c) Hash table d)Stack

Program:

WEEK: 9(A) **Date:**

AIM: Write a JAVA program for producer and consumer problem using Threads

Program:

```
public class ProducerConsumerTest {
  public static void main(String[] args) {
    CubbyHole c = new CubbyHole();
    Producer p1 = new Producer(c, 1);
    Consumer c1 = new Consumer(c, 1);
    p1.start();
    c1.start();
  }
}
class CubbyHole {
  private int contents;
  private boolean available = false;

  public synchronized int get() {
    while (available == false) {
      try {
        wait();
      } catch (InterruptedException e) {}
    }
    available = false;
    notifyAll();
    return contents;
```

```
  }
  public synchronized void put(int value) {
    while (available == true) {
      try {
        wait();
      } catch (InterruptedException e) { }
    }
    contents = value;
    available = true;
    notifyAll();
  }
}
class Consumer extends Thread {
  private CubbyHole cubbyhole;
  private int number;

  public Consumer(CubbyHole c, int number) {
    cubbyhole = c;
    this.number = number;
  }
  public void run() {
    int value = 0;
    for (int i = 0; i < 10; i++) {
      value = cubbyhole.get();
      System.out.println("Consumer #" + this.number + " got: " + value);
```

```
    }
  }
}
class Producer extends Thread {
  private CubbyHole cubbyhole;
  private int number;
  public Producer(CubbyHole c, int number) {
    cubbyhole = c;
    this.number = number;
  }
  public void run() {
    for (int i = 0; i < 10; i++) {
      cubbyhole.put(i);
      System.out.println("Producer #" + this.number + " put: " + i);
      try {
        sleep((int)(Math.random() * 100));
      } catch (InterruptedException e) { }
    }
  }
}
```

OUTPUT:

Signature of the faculty

EXERCISE:

WEEK: 9(B) **Date:**

AIM: Write a JAVA program to demonstrate lambda expressions.

Program:

WEEK: 10(A) **Date:**

AIM: Write a JAVA program to list all the files in a directory including the files present in all its subdirectories.

Program:

Step 1: Create a File Object for the directory.
Step 2: Obtain the array of files and subdirectory of that directory.
Step 3: If array[j] is a file, then display the file name and recursively go to the next element of the array[j].
Step 4: If array[j] is a directory, then display then directory name, and repeat from step 2.

```
import java.io.File;
public class DisplayFileExample1
{
public void printFileNames(File[] a, int i, int lvl)
{
// base case of the recursion  i == a.length means the directory has  no more files. Hence, the recursion has to stop
if(i == a.length)
{
return;
}
// tabs for providing the indentation for the files of sub-directory
for (int j = 0; j < lvl; j++)
{
System.out.print("\t");
}
// checking if the encountered object is a file or not
if(a[i].isFile())
{
System.out.println(a[i].getName());
}
// for sub-directories
else if(a[i].isDirectory())
{
System.out.println("[" + a[i].getName() + "]");
// recursion for sub-directories
printFileNames(a[i].listFiles(), 0, lvl + 1);
}
```

```
// recursively printing files from the directory    i + 1 means look for the next file
printFileNames(a, i + 1, lvl);
}
public static void main(String[] argvs)
{
// Providing the full path for the directory
String path = "E:\\Documents";
File fObj = new File(path);                // creating a file object
DisplayFileExample1 obj = new DisplayFileExample1();
if(fObj.exists() && fObj.isDirectory())
{
// array for the files of the directory pointed by fObj
File a[] = fObj.listFiles();

System.out.println("= = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = =");
System.out.println("Displaying Files from the directory: " + fObj);
System.out.println("= = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = =");
// Calling the method
obj.printFileNames(a, 0, 0);
}
}
}
```

OUTPUT:

Signature of the faculty

EXERCISE:

WEEK: 10(B) **Date:**

AIM: Write a Program to Read the Content of a File Line by Line

Program:

WEEK: 11(A) **Date:**

AIM: Write a JAVA program that connects to a database using JDBC display all records in a table

Program:

```
// This program will fetch all the records of emp table.


import java.sql.*;

class MysqlCon
{
public static void main(String args[]){

try{
Class.forName("com.mysql.jdbc.Driver");

Connection con= DriverManager .getConnection ("jdbc:mysql://localhost:3306 /sonoo", "root","root");
//here sonoo is the database name, root is the username and root is the password

Statement stmt=con.createStatement();

ResultSet rs=stmt.executeQuery("select * from emp");

while(rs.next())
System.out.println(rs.getInt(1)+"  "+rs.getString(2)+"  "+rs.getString(3));

con.close();

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

}
}
```

// To connect java application with the mysql database, **mysqlconnector.jar** file is required to be loaded.

Two ways to load the jar file:

Paste the mysqlconnector.jar file in jre/lib/ext folder

Set classpath

1) Paste the mysqlconnector.jar file in JRE/lib/ext folder:

Download the mysqlconnector.jar file. Go to jre/lib/ext folder and paste the jar file here.

2) Set classpath:

There are two ways to set the classpath:

1. temporary
2. permanent

set the temporary classpath

open command prompt and write:

C:>set classpath=c:\folder\mysql-connector-java-5.0.8-bin.jar;.;

set the permanent classpath

Go to environment variable then click on new tab. In variable name write classpath and in variable value paste the path to the mysqlconnector.jar file by appending mysqlconnector.jar;.; as C:\folder\mysql-connector-java-5.0.8-bin.jar;.;

WEEK: 11(B) **Date:**

AIM: Write a program to connect to a database using JDBC and insert values into it.

Program:

```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;
import java.sql.Statement;

public class JDBCExample {
  static final String DB_URL = "jdbc:mysql://localhost/TUTORIALSPOINT";
  static final String USER = "guest";
  static final String PASS = "guest123";

  public static void main(String[] args) {
    // Open a connection
    try(Connection conn = DriverManager.getConnection(DB_URL, USER, PASS);
      Statement stmt = conn.createStatement();  )
   {
      // Execute a query
      System.out.println("Inserting records into the table...");
      String sql = "INSERT INTO Registration VALUES (100, 'Zara', 'Ali', 18)";
      stmt.executeUpdate(sql);
      sql = "INSERT INTO Registration VALUES (101, 'Mahnaz', 'Fatma', 25)";
      stmt.executeUpdate(sql);
      sql = "INSERT INTO Registration VALUES (102, 'Zaid', 'Khan', 30)";
      stmt.executeUpdate(sql);
      sql = "INSERT INTO Registration VALUES(103, 'Sumit', 'Mittal', 28)";
      stmt.executeUpdate(sql);
      System.out.println("Inserted records into the table...");
    } catch (SQLException e) {
      e.printStackTrace();
    }
  }
}
```

OUTPUT:

```
C:\>javac JDBCExample.java
C:\>java JDBCExample
Inserting records into the table...
Inserted records into the table...
C:\>
```

Signature of the faculty

EXERCEISE:

WEEK: 11(C) **Date:**

AIM: Write a program to connect to a database using JDBC and delete values from it

Program:

WEEK: 12 **Date:**

AIM: Write a program that works as a simple calculator. Use a Grid Layout to arrange Buttons for digits and for the + - * % operations. Add a text field to display the result.

Program:

```
import javax.swing.*; import javax.swing.event.*; import java.awt.*;
import java.awt.event.*;
class A extends JFrame implements ActionListener
{
public JButton b1, b2, b3, b4, b5, b6, b7, b8, b9, b10, b11, b12, b13, b14, b15, b16;
public JTextField tf1;
public JPanel p;
 public String v = "";
 public String v1 = "0";
public String op = "";
public A( )
{
setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
setSize(400, 400);
p = new JPanel(new FlowLayout());
tf1 = new JTextField(10);
p.add(tf1);
add(p);
setLayout(new GridLayout(0, 3));
b1 = new JButton("1"); b1.addActionListener(this); add(b1);
b2 = new JButton("2"); b2.addActionListener(this); add(b2);
b3 = new JButton("3"); b3.addActionListener(this); add(b3);
b4 = new JButton("4"); b4.addActionListener(this); add(b4);
b5 = new JButton("5"); b5.addActionListener(this); add(b5);
b6 = new JButton("6"); b6.addActionListener(this); add(b6);
b7 = new JButton("7"); b7.addActionListener(this); add(b7);
b8 = new JButton("8"); b8.addActionListener(this); add(b8);
b9 = new JButton("9"); b9.addActionListener(this); add(b9);
b10 = new JButton("0"); b10.addActionListener(this); add(b10);
```

```
b11 = new JButton("+"); b11.addActionListener(this); add(b11);
b12 = new JButton("-"); b12.addActionListener(this); add(b12);
b13 = new JButton("*"); b13.addActionListener(this); add(b13);
b14 = new JButton("/"); b14.addActionListener(this); add(b14);
b16 = new JButton("%"); b16.addActionListener(this); add(b16);
b15 = new JButton("="); b15.addActionListener(this); add(b15);
setVisible(true);
public void actionPerformed(ActionEvent ae)
{
String b = ae.getActionCommand(); switch (b)
{
case "1":
{
v = v + "1";
tf1.setText(v);
}
break; case "2": {
v = v + "2";
tf1.setText(v);
}
break; case "3": {
v = v + "3";
tf1.setText(v);
}
break; case "4": {
v = v + "4";
tf1.setText(v);
}
break; case "5": {
v = v + "5";
tf1.setText(v);
}
break; case "6": {
v = v + "6";
```

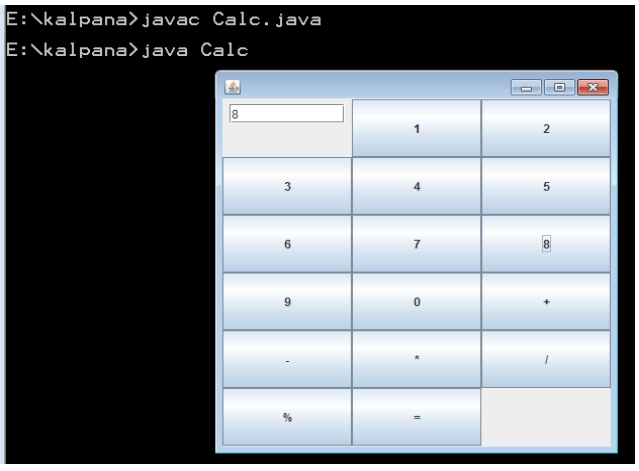
```
tf1.setText(v);
}
break;
case "7": {
v = v + "7";
tf1.setText(v);
}
break; case "8": {
v = v + "8";
tf1.setText(v);
}
break; case "9": {
v = v + "9";
tf1.setText(v);
}
break; case "0": {
v = v + "0";
tf1.setText(v);
}
break; case "+": {
op = "+";
v1 = tf1.getText(); v = "";
}
break; case "-": {
op = "-";
v1 = tf1.getText(); v = "";
}
break; case "*": {
op = "*";
v1 = tf1.getText(); v = "";
}
break; case "/": {
op = "/";
v1 = tf1.getText(); v = "";
```

```
}
break;
case "%": {
op = "%";
v1 = tf1.getText(); v = "";
}
break; case "=": {
switch (op) {
case "+": {
v = tf1.getText();
if (v.equals("")) { v = "0";
}
long i = Long.parseLong(v1) + Long.parseLong(v);
tf1.setText(String.valueOf(i));
v="";
}
break; case "-": {
v = tf1.getText();
if (v.equals("")) {
v = "0";
}
long i = Long.parseLong(v1) - Long.parseLong(v);
tf1.setText(String.valueOf(i));
v="";
}
break;
case "*": {
v = tf1.getText(); if (v.equals("")) {
v = "0";
}
long i = Long.parseLong(v1) * Long.parseLong(v);
tf1.setText(String.valueOf(i));
v="";
}
```

```
break;
case "/": {
try {
v = tf1.getText();
if (v.equals("")) {
v = "0";
}
long i = Long.parseLong(v1) / Long.parseLong(v);
tf1.setText(String.valueOf(i));
v="";
}
 catch (Exception ex)
{ JOptionPane.showMessageDialog(this, ex.getMessage());
}
}
break;
case "%": {
try {
v = tf1.getText();
if (v.equals("")) {
v = "0";
}
long i = Long.parseLong(v1) % Long.parseLong(v);
tf1.setText(String.valueOf(i));
v="";
}
catch (Exception ex) {
JOptionPane.showMessageDialog(this, ex.getMessage());
}
}
break;
}
}
break;
```

```
}
}
}
public class Calc
{
public static void main(String[] args)
{
A a = new A();
}
}
```

OUTPUT:



Signature of the faculty