JAVA PROGRAMMING

Laboratory Manual and Record

B.TECH
(II YEAR – II SEM)
(2020-21)

DEPARTMENT OF INFORMATION TECHNOLOGY

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 INFORMATION TECHNOLOGY

VISION

- To achieve high quality in technical education that provides the skills and attitude to adapt to the global needs of the Information Technology sector, through academic and research excellence..

MISSION

- To equip the students with the cognizance for problem solving and to improve the teaching learning pedagogy by using innovative techniques.
- To strengthen the knowledge base of the faculty and students with motivation towards possession of effective academic skills and relevant research experience.
- To promote the necessary moral and ethical values among the engineers, for the betterment of the society.

PROGRAMME EDUCATIONAL OBJECTIVES (PEOs)

PEO1: PROFESSIONALISM & CITIZENSHIP

To create and sustain a community of learning in which students acquire knowledge and learn to apply it professionally with due consideration for ethical, ecological and economic issues.

PEO2: TECHNICAL ACCOMPLISHMENTS

To provide knowledge-based services to satisfy the needs of society and the industry by providing hands on experience in various technologies in core field.

PEO3: INVENTION, INNOVATION AND CREATIVITY

To make the students to design, experiment, analyze, interpret in the core field with the help of other multi-disciplinary concepts wherever applicable.

PEO4: PROFESSIONAL DEVELOPMENT

To educate the students to disseminate research findings with good soft skills and become a successful entrepreneur.

PEO5: HUMAN RESOURCE DEVELOPMENT

To graduate the students in building national capabilities in technology, education and research

PROGRAM SPECIFIC OUTCOMES (PSOs)

After the completion of the course, B. Tech Information Technology, the graduates will have the following Program Specific Outcomes:

1. **Fundamentals and critical knowledge of the Computer System:-** Able to Understand the working principles of the computer System and its components , Apply the knowledge to build, asses, and analyze the software and hardware aspects of it .

2. **The comprehensive and Applicative knowledge of Software Development:** Comprehensive skills of Programming Languages, Software process models, methodologies, and able to plan, develop, test, analyze, and manage the software and hardware intensive systems in heterogeneous platforms individually or working in teams.

3. **Applications of Computing Domain & Research:** Able to use the professional, managerial, interdisciplinary skill set, and domain specific tools in development processes, identify the research gaps, and provide innovative solutions to them.

PROGRAM OUTCOMES (POs)

Engineering Graduates should possess the following:

1. **Engineering knowledge**: Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.
2. **Problem analysis**: Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.
3. **Design / development of solutions**: Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.
4. **Conduct investigations of complex problems**: Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.
5. **Modern tool usage**: Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations.
6. **The engineer and society**: Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.
7. **Environment and sustainability**: Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.
8. **Ethics**: Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.
9. **Individual and team work**: Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.
10. **Communication**: Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.
11. **Project management and finance**: Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multi disciplinary environments.
12. **Life- long learning**: Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

MALLA REDDY COLLEGE OF ENGINEERING & TECHNOLOGY

Maisammaguda, Dhulapally Post, Via Hakimpet, Secunderabad – 500100

DEPARTMENT OF INFORMATION TECHNOLOGY

GENERAL LABORATORY INSTRUCTIONS

1. Students are advised to come to the laboratory at least 5 minutes before (to the starting time), those who come after 5 minutes will not be allowed into the lab.
2. Plan your task properly much before to the commencement, come prepared to the lab with the synopsis / program / experiment details.
3. Student should enter into the laboratory with:

a. Laboratory observation notes with all the details (Problem statement, Aim, Algorithm, Procedure, Program, Expected Output, etc.,) filled in for the lab session.

b. Laboratory Record updated up to the last session experiments and other utensils (if any) needed in the lab.

c. Proper Dress code and Identity card.

4. Sign in the laboratory login register, write the TIME-IN, and occupy the computer system allotted to you by the faculty.
5. Execute your task in the laboratory, and record the results / output in the lab observation note book, and get certified by the concerned faculty.
6. All the students should be polite and cooperative with the laboratory staff, must maintain the discipline and decency in the laboratory.
7. Computer labs are established with sophisticated and high end branded systems, which should be utilized properly.
8. Students / Faculty must keep their mobile phones in SWITCHED OFF mode during the lab sessions. Misuse of the equipment, misbehaviors with the staff and systems etc., will attract severe punishment.
9. Students must take the permission of the faculty in case of any urgency to go out; if anybody found loitering outside the lab / class without permission during working hours will be treated seriously and punished appropriately.
10. Students should LOG OFF/ SHUT DOWN the computer system before he/she leaves the lab after completing the task (experiment) in all aspects. He/she must ensure the system / seat is kept properly.

HEAD OF THE DEPARTMENT **PRINCIPAL**

INDEX

WEEK-1: **DATE:**

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

```
/*Non Recursive Solution*/
import java.util.Scanner;
class Fib {
	public static void main(String args[ ]) {
		Scanner input=new Scanner(System.in);
		int i,a=0,b=1,c=0,t;
		System.out.println("Enter value of t:");
		t=input.nextInt();
		System.out.print(a);
		System.out.print(" "+b);
		for(i=0;i<t-2;i++) {
			c=a+b;
			a=b;
			b=c;
			System.out.print(" "+c);
		}
		System.out.println();
		System.out.print(t+"th value of the series is: "+c);
	}
}

/* Recursive Solution*/
import java.io.*;
import java.lang.*;
class Demo {
	int fib(int n) {
		if(n==1)
			return (1);
		else if(n==2)
			return (1);
		else
			return (fib(n-1)+fib(n-2));
	}
}
class RecFibDemo {
	public static void main(String args[])throws IOException {
		InputStreamReader obj=new InputStreamReader(System.in);
		BufferedReader br=new BufferedReader(obj);
		System.out.println("enter last number");
		int n=Integer.parseInt(br.readLine());
```

```
                    Demo ob=new Demo();
                    System.out.println("fibonacci series is as follows");
                    int res=0;
                    for(int i=1;i<=n;i++) {
                            res=ob.fib(i);
                    System.out.println(" "+res);
                    }
                    System.out.println();
                    System.out.println(n+"th value of the series is "+res);
            }
}
```

Record Notes:

Record Notes:

Record Notes:

Record Notes:

Record Notes:

Record Notes:

B) Write a java program to multiply two given matrices.

```
import java.util.Scanner;

class Matrixmul
{
 public static void main(String args[])
 {
  int  m, n, p, q, sum = 0, i, j, k;

  Scanner in = new Scanner(System.in);
  System.out.println("Enter the number of rows and columns of first matrix");
  m = in.nextInt();
  n = in.nextInt();

  int first[][] = new int[m][n];

  System.out.println("Enter elements of first matrix");

  for (i = 0; i < m; i++)
   for (j = 0; j < n; j++)
    first[i][j] = in.nextInt();

  System.out.println("Enter the number of rows and columns of second matrix");
  p = in.nextInt();
  q = in.nextInt();

  if (n != p)
   System.out.println("The matrices can't be multiplied with each other.");
  else
  {
   int second[][] = new int[p][q];
   int multiply[][] = new int[m][q];

   System.out.println("Enter elements of second matrix");

   for (i = 0; i < p; i++)
    for (j = 0; j < q; j++)
     second[i][j] = in.nextInt();

   for (i = 0; i < m; i++) {
    for (j = 0; j < q; j++) {
     for (k = 0; k < p; k++)
      sum = sum + first[i][k]*second[k][j];

     multiply[i][j] = sum;
     sum = 0;
```

```
        }
      }
       System.out.println("Product of the matrices:");
       for (i = 0; i < m; i++) {
        for (j = 0; j < q; j++)
          System.out.print(multiply[i][j]+"\t");

       System.out.print("\n");
      }
    }
  }
}
```

Record Notes:

Record Notes:

Record Notes:

Record Notes:

Record Notes:

Record Notes:

Week 2:

A) Write a java program for Method overloading and Constructor overloading.

```
// Method overloading in Java.

public class Sum {

    // Overloaded sum(). This sum takes two int parameters
    public int sum(int x, int y)
    {
        return (x + y);
    }

    // Overloaded sum(). This sum takes three int parameters
    public int sum(int x, int y, int z)
    {
        return (x + y + z);
    }

    // Overloaded sum(). This sum takes two double parameters
    public double sum(double x, double y)
    {
        return (x + y);
    }

    // Driver code
    public static void main(String args[])
    {
        Sum s = new Sum();
        System.out.println(s.sum(10, 20));
        System.out.println(s.sum(10, 20, 30));
        System.out.println(s.sum(10.5, 20.5));
    }
}

//Constructor overloading in Java.

class StudentData
{
   private int stuID;
   private String stuName;
   private int stuAge;
   StudentData()
   {
      //Default constructor
      stuID = 100;
      stuName = "New Student";
      stuAge = 18;
```

```
  }
 StudentData(int num1, String str, int num2)
 {
      //Parameterized constructor
      stuID = num1;
      stuName = str;
      stuAge = num2;
  }
  //Getter and setter methods
  public int getStuID() {
      return stuID;
  }
  public void setStuID(int stuID) {
      this.stuID = stuID;
  }
  public String getStuName() {
      return stuName;
  }
  public void setStuName(String stuName) {
      this.stuName = stuName;
  }
  public int getStuAge() {
      return stuAge;
  }
  public void setStuAge(int stuAge) {
      this.stuAge = stuAge;
  }

  public static void main(String args[])
  {
      //This object creation would call the default constructor
      StudentData myobj = new StudentData();
      System.out.println("Student Name is: "+myobj.getStuName());
      System.out.println("Student Age is: "+myobj.getStuAge());
      System.out.println("Student ID is: "+myobj.getStuID());

      /*This object creation would call the parameterized constructor StudentData(int, String, int)*/

      StudentData myobj2 = new StudentData(555, "Chaitanya", 25);
      System.out.println("Student Name is: "+myobj2.getStuName());
      System.out.println("Student Age is: "+myobj2.getStuAge());
      System.out.println("Student ID is: "+myobj2.getStuID());
  }
}
```

Record Notes:

Record Notes:

Record Notes:

Record Notes:

Record Notes:

Record Notes:

B) Write a java program to display the employee details using Scanner class.

```
import java.util.Scanner;

   class Employee
   {
      int Id;
      String Name;
      int Age;
      long Salary;

      void GetData()          // Defining GetData()
      {

         Scanner sc = new Scanner(System.in);

         System.out.print("\n\tEnter Employee Id : ");
         Id = Integer.parseInt(sc.nextLine());

         System.out.print("\n\tEnter Employee Name : ");
         Name = sc.nextLine();

         System.out.print("\n\tEnter Employee Age : ");
         Age = Integer.parseInt(sc.nextLine());

         System.out.print("\n\tEnter Employee Salary : ");
         Salary = Integer.parseInt(sc.nextLine());

      }

      void PutData()          // Defining PutData()
      {
         System.out.print("\n\t" + Id + "\t" +Name + "\t" +Age + "\t" +Salary);
      }

      public static void main(String args[])
      {

         Employee[] Emp = new Employee[3];
         int i;

         for(i=0;i<3;i++)
            Emp[i] =  new Employee();   // Allocating memory to each object

         for(i=0;i<3;i++)
         {
            System.out.print("\nEnter details of "+ (i+1) +" Employee\n");
            Emp[i].GetData();
         }

         System.out.print("\nDetails of Employees\n");
         for(i=0;i<3;i++)
```

```
Emp[i].PutData();

    }
}
```

Record Notes:

Record Notes:

Record Notes:

Record Notes:

Record Notes:

Record Notes:

C) Write a java program that checks whether a given string is palindrome or not.

```
import java.util.Scanner;

class ChkPalindrome
{
  public static void main(String args[])
  {
    String str, rev = "";
    Scanner sc = new Scanner(System.in);

    System.out.println("Enter a string:");
    str = sc.nextLine();

    int length = str.length();

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

    if (str.equals(rev))
      System.out.println(str+" is a palindrome");
    else
      System.out.println(str+" is not a palindrome");

  }
}
```

Record Notes:

Record Notes:

Record Notes:

Record Notes:

Record Notes:

Record Notes:

Record Notes:

Week 3:

A) Write a java program to represent Abstract class with example.

```
// Abstract class that contains abstract method.

abstract class Shape
{
abstract void numberOfSides();
}
// Classes that illustrates the abstract method.
class Trapezoid
{
void    numberOfSides()
{
System.out.println("The no. of side's in trapezoidal are6");
}
}

class   Triangle
{
void    numberOfSides()
{
System.out.println("The no. of side's in triangle are:3 ");
}
}
class   Hexagon
{
void numberOfSides()
{       System.out.println("The no. of side's in hexagon are:6  ");
}
}
// Class that create objects and call the method.
class ShapeDemo
{
public static void main(String args[])
{
Trapezoid obj1 = new Trapezoid();
Triangle obj2 = new Triangle();
Hexagon obj3 = new Hexagon();
obj1.numberOfSides();
obj2.numberOfSides();
obj3.numberOfSides(); }
}
```

Record Notes:

Record Notes:

Record Notes:

Record Notes:

Record Notes:

Record Notes:

Record Notes:

B) Write a java program to implement Interface using extends keyword.

```
interface Inf1{
  public void method1();
}
interface Inf2 extends Inf1 {
  public void method2();
}
public class Demo implements Inf2{
  /* Even though this class is only implementing the interface Inf2, it has to implement all
the methods of Inf1 as well because the interface Inf2 extends Inf1 */
   public void method1(){
    System.out.println("method1");
   }
   public void method2(){
    System.out.println("method2");
   }
   public static void main(String args[]){
    Inf2 obj = new Demo();
    obj.method2();
   }
}
```

Record Notes:

Record Notes:

Record Notes:

Record Notes:

Record Notes:

Record Notes:

Record Notes:

Week 4:

Write a java program to implement Interface using extends keyword.

```
interface Inf1{
  public void method1();
}
interface Inf2 extends Inf1 {
  public void method2();
}
public class Demo implements Inf2{
  /* Even though this class is only implementing the interface Inf2, it has to implement all
the methods of Inf1 as well because the interface Inf2 extends Inf1 */
   public void method1(){
    System.out.println("method1");
   }
   public void method2(){
    System.out.println("method2");
   }
   public static void main(String args[]){
    Inf2 obj = new Demo();
    obj.method2();
   }
}
```

Record Notes:

Record Notes:

Record Notes:

Record Notes:

Record Notes:

Record Notes:

Record Notes:

Week 5:

A) Write a java program to create inner classes.

```
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();
   }
}
```

Record Notes:

Record Notes:

Record Notes:

Record Notes:

Record Notes:

Record Notes:

Record Notes:

B) Write a java program for creating multiple catch blocks.

```
class MultipleExceptionHandling{
   public static void main(String args[]){
      try{
        System.out.println("Begin: try block");
         String name="abc";
         int nameLength = name.length();
         int res= 10/0;

         System.out.println("End: try block");
      }
      catch(ArithmeticException e){
         System.out.println("Division is not allowed with zero
denominator");
      }
      catch(NullPointerException e){
         System.out.println("Name should not be Null");
      }
      catch(Exception e){
         System.out.println("General Exception");
      }
      System.out.println("End: try-catch block");
   }
}
```

Record Notes:

Record Notes:

Record Notes:

Record Notes:

Record Notes:

Record Notes:

Record Notes:

Week 6:

A) Write a java program for producer and consumer problem using Threads.

```
// Java program to implement solution of producer  consumer problem.

import java.util.LinkedList;

public class Threadexample {
	public static void main(String[] args)
		throws InterruptedException
	{
		// Object of a class that has both produce()  and consume() methods
		final PC pc = new PC();

		// Create producer thread
		Thread t1 = new Thread(new Runnable() {
			@Override
			public void run()
			{
				try {
					pc.produce();
				}
				catch (InterruptedException e) {
					e.printStackTrace();
				}
			}
		});

		// Create consumer thread

		Thread t2 = new Thread(new Runnable() {
			@Override
			public void run()
			{
				try {
					pc.consume();
				}
				catch (InterruptedException e) {
					e.printStackTrace();
				}
			}
		});

		// Start both threads

		t1.start();
		t2.start();
```

```
// t1 finishes before t2

        t1.join();
        t2.join();
}

// This class has a list, producer (adds items to list and consumer (removes items).
public static class PC {

        // Create a list shared by producer and consumer
        // Size of list is 2.

        LinkedList<Integer> list = new LinkedList<>();
        int capacity = 2;

        // Function called by producer thread
        public void produce() throws InterruptedException
        {
                int value = 0;
                while (true) {
                        synchronized (this)
                        {
                                // producer thread waits while list is full
                                while (list.size() == capacity)
                                        wait();

                                System.out.println("Producer produced-"+ value);

                                // to insert the jobs in the list
                                list.add(value++);

                                // notifies the consumer thread that
                                // now it can start consuming
                                notify();

                                // makes the working of program easier
                                // to understand
                                Thread.sleep(1000);
                        }
                }
        }

        // Function called by consumer thread
        public void consume() throws InterruptedException
        {
                while (true) {
                        synchronized (this)
```

```
{
                                        // consumer thread waits while list
                                        // is empty

                                        while (list.size() == 0)
                                                wait();

                                        // to retrive the ifrst job in the list
                                        int val = list.removeFirst();

                                        System.out.println("Consumer consumed-" + val);

                                        // Wake up producer thread
                                        notify();

                                        // and sleep
                                        Thread.sleep(1000);
                                }
                        }
                }
        }
}
```

Record Notes:

Record Notes:

Record Notes:

Record Notes:

Record Notes:

Record Notes:

Record Notes:

B) Write a Java program that implements a multi-thread application that has three threads.

/* The first thread displays "Good Morning" for every one second, the second thread displays "Hello" for every two seconds and third thread displays "Welcome" for every three seconds */

```
class GoodMorning extends Thread {
    synchronized public void run() {
        try {
            int i=0;
            while (i<5) {
                sleep(1000);
                System.out.println("Good morning ");
                i++;
            }
        } catch (Exception e) {
        }
    }
}

class Hello extends Thread {
    synchronized public void run() {
        try {
            int i=0;
            while (i<5) {
                sleep(2000);
                System.out.println("hello");
                i++;
            }
        } catch (Exception e) {
        }
    }
}

class Welcome extends Thread {
    synchronized public void run() {
        try {
            int i=0;
            while (i<5) {
                sleep(3000);
                System.out.println("welcome");
                i++;
            }
        } catch (Exception e) {
        }
    }
}

class MultithreadDemo {
    public static void main(String args[]) {
        GoodMorning t1 = new GoodMorning();
        Hello t2 = new Hello();
```

```
        Welcome t3 = new Welcome();

        t1.start();
        t2.start();
        t3.start();

    }
}
```

Record Notes:

Record Notes:

Record Notes:

Record Notes:

Record Notes:

Record Notes:

Week 7:

A) Write a java program to display File class properties.

// Program to check if a file or directory physically exist or not.

/* Accept a file or directory name from command line arguments. Then the program will check if that file or directory physically exist or not and it displays the property of that file or directory.*/

```
*import java.io.File;

// Displaying file property

class fileProperty

{

    public static void main(String[] args) {

                    //accept file name or directory name through command line args
                    String fname =args[0];

                    //pass the filename or directory name to File object
                    File f = new File(fname);

                    //apply File class methods on File object

                    System.out.println("File name :"+f.getName());
                    System.out.println("Path: "+f.getPath());
                    System.out.println("Absolute path:" +f.getAbsolutePath());
                    System.out.println("Parent:"+f.getParent());
                    System.out.println("Exists :"+f.exists());

            if(f.exists())
            {

                    System.out.println("Is writeable:"+f.canWrite());
                    System.out.println("Is readable"+f.canRead());
                    System.out.println("Is a directory:"+f.isDirectory());
                    System.out.println("File Size in bytes "+f.length());
            }

    }

}
```

Record Notes:

Record Notes:

Record Notes:

Record Notes:

Record Notes:

Record Notes:

Record Notes:

B) Write a java program to represent ArrayList class.

```
// Java program to demonstrate working of ArrayList in Java

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

class arrayli
{
                public static void main(String[] args) throws IOException
                {
                  // size of ArrayList

                  int n = 5;

                  //declaring ArrayList with initial size n

                  ArrayList<Integer> arrli = new ArrayList<Integer>(n);

                  // Appending the new element at the end of the list

                  for (int i=1; i<=n; i++)
                  arrli.add(i);

                  // Printing elements

                  System.out.println(arrli);

                  // Remove element at index 3

                  arrli.remove(3);

                  // Displaying ArrayList after deletion

                  System.out.println(arrli);

                  // Printing elements one by one

                  for (int i=0; i<arrli.size(); i++)
                  System.out.print(arrli.get(i)+" ");
                }
}
```

Record Notes:

Record Notes:

Record Notes:

Record Notes:

Record Notes:

Record Notes:

Record Notes:

Week 8:

Write a Java program loads phone no, name from a text file using hash table.

```
/*Create a text file hashtab.txt  with name followed by a tab and number
akhil     123
srujan   345
edrf      567
Then create a java file Phonebook.java */

import java.io.*;
import java.util.*;
public class Phonebook
{
 public static void main(String args[])
 {
 try
 {
  FileInputStream fis=new FileInputStream("hashtab.txt");
  Scanner sc=new Scanner(fis).useDelimiter("\t");
  Hashtable<String,String> ht=new Hashtable<String,String> ();
  String[] strarray;
  String a,str;
  while(sc.hasNext())
{
 a=sc.nextLine();
 strarray=a.split("\t");
 ht.put(strarray[0],strarray[1]);
 System.out.println("hash table values are"+strarray[0]+":"+strarray[1]);
}
Scanner s=new Scanner(System.in);
System.out.println("enter the name as given in the phone book");
str=s.next();
if(ht.containsKey(str))
{
System.out.println("phone no is"+ht.get(str));
}
else
{
System.out.println("name is not matched");
}
 }
 catch(Exception e)
 {
System.out.println(e);
 }
}
}
```

Record Notes:

Record Notes:

Record Notes:

Record Notes:

Record Notes:

Record Notes:

Record Notes:

Week 9:

A) Write an applet program that displays a simple message

Applet1.java:

```
// Import the packages to access the classes and methods in awt and applet classes.

import java.awt.*;
 import java.applet.*;

public class Applet1 extends Applet
{

// Paint method to display the message. public void paint(Graphics g)

{
g.drawString("HELLO WORLD",20,20);
}

}
```

Applet1.html:

```
/* <applet code="Applet1" width=200 height=300> </applet>*/
```

Record Notes:

Record Notes:

Record Notes:

Record Notes:

Record Notes:

Record Notes:

Record Notes:

B) Write a Java program to compute factorial value using Applet.

```
import java.applet.*;
import java.awt.event.*;
import java.awt.*;

/*<applet code="Factorial" width=500 height=500></applet> */

public class FactorialApplet extends Applet implements ActionListener
{
  Label l1,l2;
  TextField t1,t2;
  Button b1,b2;

    public void init()
  {
     l1=new Label("Enter a value: ");
     l2=new Label("Result:");
     t1=new TextField(10);
     t2=new TextField(10);
     b1=new Button("Calculate");
     b2=new Button("Clear");
     add(l1);
     add(t1);
     add(b1);
     add(b2);
     add(l2);
     add(t2);
     b1.addActionListener(this);
     b2.addActionListener(this);
  }

  public void actionPerformed(ActionEvent ae)
  {
     int n=Integer.parseInt(t1.getText());
     int fact=1;
     if(ae.getSource()==b1)
     {
        if(n==0||n==1)
        {
           fact=1;
        t2.setText(String.valueOf(fact));
        }

    else

     {
```

```
        for(int i=1;i<=n;i++)
            fact=fact*i;
 }
    t2.setText(String.valueOf(fact));
     }
     else if(ae.getSource()==b2)
     {
        t1.setText("");
        t2.setText("");
     }

  }
}
```

Record Notes:

Record Notes:

Record Notes:

Record Notes:

Record Notes:

Record Notes:

C) Write a program for passing parameters using Applet.

```
import java.awt.*;
import java.applet.*;

public class MyApplet extends Applet
{
                String n;
                String a;
                public void init()
                {
                    n = getParameter("name");
                    a = getParameter("age");
                }

                public  void paint(Graphics g)
                {
                    g.drawString("Name is: " + n, 20, 20);
                    g.drawString("Age is: " + a, 20, 40);
                }
}



/*
                <applet code="MyApplet" height="300" width="500">
                    <param name="name" value="Ramesh" />
                    <param name="age" value="25" />
                </applet>
*/
```

Record Notes:

Record Notes:

Record Notes:

Record Notes:

Record Notes:

Record Notes:

Record Notes:

Week 10:

Write a java program for handling Mouse events and Key events

```
//Program to implement mouse events
import java.awt.*;
import java.awt.event.*;
import java.applet.*;

public class Mouseevents extends Applet implements MouseListener,MouseMotionListener
{
        /*<applet code="Mouseevents" width=500 height=300></applet>*/
        String msg="";
        int mousex=0,mousey=0;
        public void init()
        {
                addMouseListener(this);
                addMouseMotionListener(this);
        }

        public void mouseClicked(MouseEvent me)
        {
                mousex=0;
                mousey=10;
                msg="mouse clicked";
                repaint();
        }

        public void mouseEntered(MouseEvent me)
        {
                mousex=0;
                mousey=10;
                msg="mouse entered";
                repaint();
        }

        public void mouseExited(MouseEvent me)
        {
                mousex=0;
                mousey=10;
                msg="mouse exited";
                repaint();
        }

       public void mousePressed(MouseEvent me)
        {
                mousex=me.getX();
```

```
                mousey=me.getY();
                msg="down";
                repaint();
        }

        public void mouseReleased(MouseEvent me)
        {
                mousex=me.getX();
                mousey=me.getY();
                msg="up";
                repaint();
        }


        public void mouseDragged(MouseEvent me)
        {
                mousex=me.getX();
                mousey=me.getY();
                msg="*";
                showStatus("Dragging mouse at "+mousex+" , "+mousey);
                repaint();
        }

        public void mouseMoved(MouseEvent me)
        {
                showStatus("mouse moving at "+me.getX()+" , "+me.getY());
        }
        public void paint(Graphics g)
        {
                g.drawString(msg,mousex,mousey);
        }
}
```

```
//Program to implement key events

import java.awt.*;
import java.awt.event.*;
import java.applet.*;

/*
<applet code="Key" width=300 height=400>
</applet>
*/

public class Key extends Applet implements KeyListener
{
                        int X=20,Y=30;
                        String msg="KeyEvents--->";

                        public void init()
                        {
                              addKeyListener(this);
                              requestFocus();
                              setBackground(Color.green);
                              setForeground(Color.blue);
                        }

                        public void keyPressed(KeyEvent k)
                        {
                              showStatus("KeyDown");
                              int key=k.getKeyCode();

                              switch(key)
                              {
                                    case KeyEvent.VK_UP:
                                          showStatus("Move to Up");
                                          break;
                                    case KeyEvent.VK_DOWN:
                                          showStatus("Move to Down");
                                          break;
                                    case KeyEvent.VK_LEFT:
                                          showStatus("Move to Left");
                                          break;
                                    case KeyEvent.VK_RIGHT:
                                          showStatus("Move to Right");
                                          break;
                              }
                              repaint();
                        }
```

```
    public void keyReleased(KeyEvent k)
    {
        showStatus("Key Up");

    }


    public void keyTyped(KeyEvent k)
    {
        msg+=k.getKeyChar();
        repaint();
    }


    public void paint(Graphics g)
    {
        g.drawString(msg,X,Y);
    }
}
```

Record Notes:

Record Notes:

Record Notes:

Record Notes:

Record Notes:

Record Notes:

Week 11:

A) Write a java program that connects to a database using JDBC program:

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

public class PostgreSQLJDBC
{
    public static void main(String args[])
    {
    Connection c = null;

    try {

    Class.forName("org.postgresql.Driver");
    c = DriverManager.getConnection("jdbc:postgresql://localhost:5432/testdb",
            "postgres", "123");

    } catch (Exception e) {

    e.printStackTrace();
    System.err.println(e.getClass().getName()+": "+e.getMessage());
     System.exit(0);

    }

    System.out.println("Opened database successfully");
    }
}
```

Record Notes:

Record Notes:

Record Notes:

Record Notes:

Record Notes:

Record Notes:

Record Notes:

B) Write a java program to connect to database using JDBC & insert values into table

```
import java.sql.*;

public class insert1
{
   public static void main(String args[])
   {
      String id = "id1";
      String pwd = "pwd1";
      String fullname = "MRCET";
      String email = "MRCET@gmail.com";

      try
      {
         Class.forName("oracle.jdbc.driver.OracleDriver");
         Connection con = DriverManager.getConnection("
          jdbc:oracle:thin:@localhost:1521:orcl", "login1", "pwd1");
         Statement stmt = con.createStatement();

         // Inserting data in database
         String q1 = "insert into userid values('" +id+ "', '" +pwd+
                        "', '" +fullname+ "', '" +email+ "')";
         int x = stmt.executeUpdate(q1);
         if (x > 0)
            System.out.println("Successfully Inserted");
         else
            System.out.println("Insert Failed");

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

Record Notes:

Record Notes:

Record Notes:

Record Notes:

Record Notes:

Record Notes:

Record Notes:

C) Write a java program to connect to a database using JDBC and delete values from table.

```
import java.sql.*;
public class delete
{
    public static void main(String args[])
   {
        String id = "id2";
        String pwd = "pwd2";
        try
        {
                Class.forName("oracle.jdbc.driver.OracleDriver");
                Connection con = DriverManager.getConnection("
                jdbc:oracle:thin:@localhost:1521:orcl", "login1", "pwd1");
                Statement stmt = con.createStatement();

                // Deleting from database
                String q1 = "DELETE from userid WHERE id = '" +
                        id + "' AND pwd = '" + pwd + "'";

                int x = stmt.executeUpdate(q1);

                if (x > 0)
                System.out.println("One User Successfully Deleted");
                 else
                System.out.println("ERROR OCCURED :(");

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

Record Notes:

Record Notes:

Record Notes:

Record Notes:

Record Notes:

Record Notes:

Record Notes:

12.Write a java 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.JOptionPane; import java.awt.*;
import java.awt.event.*;

// Class that initialize the applet and create calculator.
public class Calculator extends JApplet
{
public void init()
{
CalculatorPanel calc=new CalculatorPanel(); getContentPane().add(calc);
}
}
// Class that creates the calculator panel .
class CalculatorPanel extends JPanel implements ActionListener
{
// Creation of JButton.
JButton n1,n2,n3,n4,n5,n6,n7,n8,n9,n0,plus,minus,mul,div,dot,equal;
static JTextField result=new JTextField("0",45); static String lastCommand=null;
// Create the JObjectPane.
JOptionPane p=new JOptionPane(); double preRes=0,secVal=0,res;
private static void assign(String no)
{
if((result.getText()).equals("0"))
result.setText(no); else
        if(lastCommand=="=")
{
result.setText(no); lastCommand=null; }
else
result.setText(result.getText()+no);
}

 // Creation of control panel of calculator and adding buttons using
 GridLayout. public CalculatorPanel()
{
setLayout(new GridLayout());
 result.setEditable(false);
 result.setSize(300,200);
 add(result);
```

```
JPanel panel=new JPanel();
panel.setLayout(new GridLayout(5,5));
n7=new JButton("7");
panel.add(n7);
n7.addActionListener(this);
n8=new JButton("8");
panel.add(n8);
n8.addActionListener(this);
n9=new JButton("9");
panel.add(n9);
n9.addActionListener(this);
div=new JButton("/");
panel.add(div);
div.addActionListener(this);
n4=new JButton("4");
panel.add(n4);
n4.addActionListener(this);
n5=new JButton("5");
panel.add(n5);
n5.addActionListener(this);
n6=new JButton("6");
panel.add(n6);
n6.addActionListener(this);
mul=new JButton("*");
panel.add(mul);
mul.addActionListener(this);
n1=new JButton("1");
panel.add(n1);
n1.addActionListener(this);
n2=new JButton("2");
panel.add(n2);
n2.addActionListener(this);
n3=new JButton("3");
panel.add(n3);
n3.addActionListener(this);
minus=new JButton("-");
panel.add(minus);
minus.addActionListener(this);
dot=new JButton(".");
panel.add(dot);
dot.addActionListener(this);
n0=new JButton("0");
 panel.add(n0); n0.addActionListener(this);
```

```
panel.add(equal);
equal.addActionListener(this); plus=new JButton("+"); panel.add(plus);
plus.addActionListener(this); add(panel);

    }
    // Implementing method in ActionListener.
    public void actionPerformed(ActionEvent
    ae)
    {
    if(ae.getSource()==n1)
              assign("1");
    else if(ae.getSource()==n2)
              assign("2");
     else if(ae.getSource()==n3)
               assign("3");
    else if(ae.getSource()==n4)
              assign("4");
     else if(ae.getSource()==n5)
              assign("5");
    else if(ae.getSource()==n6)
              assign("6");
     else if(ae.getSource()==n7)
              assign("7");
    else if(ae.getSource()==n8)
               assign("8");
     else if(ae.getSource()==n9)
              assign("9");
    else if(ae.getSource()==n0)
              assign("0");
    else     if(ae.getSource()==dot)
    {
    if(((result.getText()).indexOf("."))==-1)
    result.setText(result.getText()+"."); } else if(ae.getSource()==minus)
    {
    preRes=Double.parseDouble(result.getText()); lastCommand="-";
    result.setText("0");
    }
    else     if(ae.getSource()==div)
    {
    preRes=Double.parseDouble(result.getText());
    lastCommand="/";
    result.setText("0");
     }
```

```
else if(ae.getSource()==equal)
{
 secVal=Double.parseDouble(result.getText());
 if(lastCommand.equals("/"))
        res=preRes/secVal;
else if(lastCommand.equals("*"))
        res=preRes*secVal;
else if(lastCommand.equals("-
        ")) res=preRes-secVal;
else if(lastCommand.equals("+"))
        res=preRes+secVal;
result.setText(" "+res); lastCommand="=";
}
else    if(ae.getSource()==mul)
{
preRes=Double.parseDouble(result.getText());
lastCommand="*";
result.setText("0");
}
else    if(ae.getSource()==plus)
{
preRes=Double.parseDouble(result.getText());
lastCommand="+";
result.setText("0");
}
}
}
```

Calculator.html:

```
<applet code="Calculator" width=200 height=300> </applet>
```

Record Notes:

Record Notes

Record Notes:

Record Notes:

Record Notes:

Record Notes: