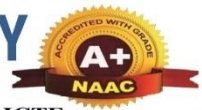




SCIENT INSTITUTE OF TECHNOLOGY

(UGC AUTONOMOUS)

Accredited by NAAC with 'A+' Grade, Affiliated to JNTUH & Approved by AICTE
Ibrahimpattam, Rangareddy, Telangana- 501506
www.scient.ac.in , scient_insteng@yahoo.co.in



JAVA LAB MANUAL

1. Use eclipse or Netbean platform and acquaint with the various menus, create a test project, add a test class and run it see how you can use auto suggestions, auto fill. Try code formatter and code refactoring like renaming variables, methods and classes. Try debug step by step with a small program of about 10 to 15 lines which contains at least one if else condition and a for loop.
2. Write a Java program to demonstrate the OOP principles. [i.e., Encapsulation, Inheritance, Polymorphism and Abstraction]
3. Write a Java program to handle checked and unchecked exceptions. Also, demonstrate the usage of custom exceptions in real time scenario.
4. Write a Java program on Random Access File class to perform different read and write operations.
5. Write a Java program to demonstrate the working of different collection classes. [Use package structure to store multiple classes]
6. Write a program to synchronize the threads acting on the same object. [Consider the example of any reservations like railway, bus, movie ticket booking, etc.]
7. Write a program to perform CRUD operations on the student table in a database using JDBC.
8. Write a Java program that works as a simple calculator. Use a grid layout to arrange buttons for the digits and for the +, -, *, % operations. Add a text field to display the result. Handle any possible exceptions like divided by zero.
9. Write a Java program that handles all mouse events and shows the event name at the center of the window when a mouse event is fired. [Use Adapter classes]

EXPERIMENT-1

1. Use eclipse or Netbean platform and acquaint with the various menus, create a test project, add a test class and run it see how you can use auto suggestions, auto fill. Try code formatter and code refactoring like renaming variables, methods and classes. Try debug step by step with a small program of about 10 to 15 lines which contains at least one if else condition and a for loop.

```
/* Sample java program to check given number is prime or not */
```

```
//Importing packages
```

```
import java.lang.System;
```

```
import java.util.Scanner;
```

```
// Creating Class
```

```
class Sample_Program {
```

```
    // main method
```

```
    public static void main(String args[]) {
```

```
        int i,count=0,n;
```

```
        // creating scanner object
```

```
        Scanner sc=new Scanner(System.in);
```

```
        // get input number from user
```

```
        System.out.print("Enter Any Number : ");
```

```
        n=sc.nextInt();
```

```
        // logic to check prime or not
```

```
        for(i=1;i<=n;i++) {
```

```
            if(n%i==0) {
```

```
                count++;
```

```
            }
```

```
        }
```

```
        if(count==2)
```

```
            System.out.println(n+" is prime");
```

```
        else
```

```
            System.out.println(n+" is not prime");
```

```
}  
}
```

OUTPUT:

```
C:\StudyGlance\Java_Lab_Programs\W1>javac Sample_Program.java  
  
C:\StudyGlance\Java_Lab_Programs\W1>java Sample_Program  
Enter Any Number : 23  
23 is prime  
  
C:\StudyGlance\Java_Lab_Programs\W1>java Sample_Program  
Enter Any Number : 45  
45 is not prime
```

EXPERIMENT-2

2. Write a Java program to demonstrate the OOP principles. [i.e., Encapsulation, Inheritance, Polymorphism and Abstraction]

/* Encapsulation:

The fields of the class are private and accessed through getter and setter methods.*/

```
class Person {  
  
    // private fields  
  
    private String name;  
  
    private int age;  
  
    // constructor  
  
    public Person(String name, int age) {  
  
        this.name = name;  
  
        this.age = age;  
  
    }  
}
```

```

// getter and setter methods

public String getName() {

    return name;

}

public void setName(String name) {

    this.name = name;

}

public int getAge() {

    return age;

}

public void setAge(int age) {

    this.age = age;

}

/* Abstraction:

    The displayInfo() method provides a simple interface to interact with the object.*/

public void displayInfo() {

    System.out.println("Name: " + name);

    System.out.println("Age: " + age);

}

}

/* Inheritance:

    Employee is a subclass of Person, inheriting its properties and methods.*/

class Employee extends Person {

    // private field

```

```

private double salary;

// constructor

public Employee(String name, int age, double salary) {

    super(name, age);

    this.salary = salary;

}

// getter and setter methods

public double getSalary() {

    return salary;

}

public void setSalary(double salary) {

    this.salary = salary;

}

/* Polymorphism:

    Overriding the displayInfo() method to provide a specific implementation for Employee.*/

@Override

public void displayInfo() {

    super.displayInfo();

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

}

}

public class OopPrinciplesDemo {

    public static void main(String[] args) {

```

```
// Demonstrating encapsulation and abstraction

Person person = new Person("Madhu", 30);

System.out.println("Person Info:");

person.displayInfo();

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


// Demonstrating inheritance and polymorphism

Employee employee = new Employee("Naveen", 26, 50000);

System.out.println("Employee Info:");

employee.displayInfo();

}

}
```

Output:

```
D:\Java>javac OopPrinciplesDemo.java

D:\Java>java OopPrinciplesDemo
Person Info:
Name: Madhu
Age: 30
=====
Employee Info:
Name: Naveen
Age: 26
Salary: 50000.0

D:\Java>
```

EXPERIMENT-3

3. Write a Java program to handle checked and unchecked exceptions. Also, demonstrate the usage of custom exceptions in real time scenario.

```
import java.io.File;

import java.io.FileReader;

import java.io.FileNotFoundException;

// Custom Exception

class InvalidAgeException extends Exception {

    public InvalidAgeException(String message) {

        super(message);

    }

}

public class ExceptionsDemo {

    // Method to demonstrate custom exception

    public static void register(String name, int age) throws InvalidAgeException {

        if (age < 18) {

            throw new InvalidAgeException("User must be at least 18 years old.");

        } else {

            System.out.println("Registration successful for user: " + name);

        }

    }

    public static void main(String[] args) {

        //Handling Checked Exception
```

```
try {  
    File file = new File("myfile.txt");  
  
    // This line can throw FileNotFoundException  
  
    FileReader fr = new FileReader(file);  
} catch (FileNotFoundException e) {  
    System.out.println("File not found: " + e.getMessage());  
}  
  
//Handling Unchecked Exception  
  
try {  
    int[] arr = {1, 2, 3};  
  
    // Accessing an out-of-bound index  
  
    System.out.println(arr[6]);  
} catch (ArrayIndexOutOfBoundsException e) {  
    System.out.println("Array index out of bounds: " + e.getMessage());  
}  
  
// Finally block to perform cleanup operations  
  
finally {  
    System.out.println("Cleanup operations can be performed here.");  
}  
  
// Demonstrate custom exception handling  
  
System.out.println("Demonstrating Custom Exception:");  
  
try {  
    // Invalid age for registration  
  
    register("Madhu", 17);  
} catch (InvalidAgeException e) {
```



```
        System.out.println("Custom Exception Caught: " + e.getMessage());
    }
}
}
```

Output:

```
D:\Java>javac ExceptionsDemo.java

D:\Java>java ExceptionsDemo
File not found: myfile.txt (The system cannot find the file specified)
Array index out of bounds: 6
Cleanup operations can be performed here.
Demonstrating Custom Exception:
Custom Exception Caught: User must be at least 18 years old.

D:\Java>
```

EXPERIMENT-4

4. Write a Java program on Random Access File class to perform different read and write operations.

```
import java.io.*;

public class RandomAccessFileExample {

    public static void main(String[] args) {

        try {

            // Create a RandomAccessFile object with read-write mode

            RandomAccessFile file = new RandomAccessFile("data.txt", "rw");

            // Write data to the file

            String data1 = "Hello";

            String data2 = "World";

            file.writeUTF(data1);
```

```
file.writeUTF(data2);

// Move the file pointer to the beginning of the file
file.seek(0);

// Read data from the file
String readData1 = file.readUTF();
String readData2 = file.readUTF();

System.out.println("Data read from file:");

System.out.println(readData1);
System.out.println(readData2);

// Move the file pointer to the ending of the file
file.seek(file.length());

// Append new data to the file
String newData = "Java!";

file.writeUTF(newData);

// Move the file pointer to the beginning of the file
file.seek(0);

// Read data from the file again after appending
readData1 = file.readUTF();
readData2 = file.readUTF();

String readData3 = file.readUTF();

System.out.println("Data read from file after appending:");

System.out.println(readData1);
System.out.println(readData2);
System.out.println(readData3);
```

```

        // Close the file

        file.close();

    } catch (IOException e) {

        System.out.println("An error occurred: " + e.getMessage());

        e.printStackTrace();

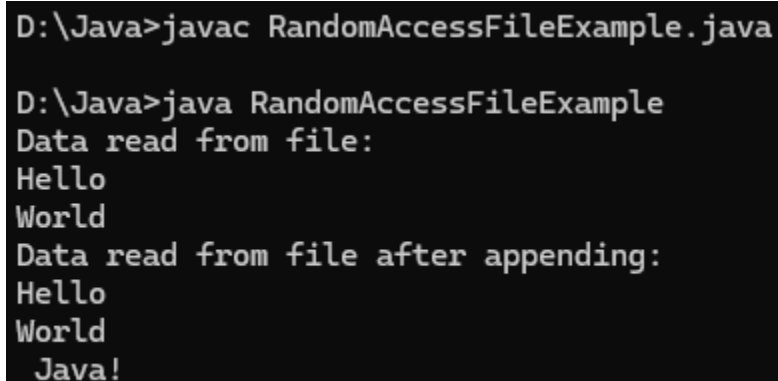
    }

}

}

```

Output:



```

D:\Java>javac RandomAccessFileExample.java

D:\Java>java RandomAccessFileExample
Data read from file:
Hello
World
Data read from file after appending:
Hello
World
Java!

```

EXPERIMENT-5

5. Write a Java program to demonstrate the working of different collection classes. [Use package structure to store multiple classes]

```

package collections;

import java.util.ArrayList;

public class ListExample {

    public static void main(String[] args) {

```

```

ArrayList<String> list = new ArrayList<>();

list.add("Apple");

list.add("Banana");

list.add("Orange");

// to display

System.out.println("List Example:");

for (String fruit : list) {

    System.out.println(fruit);

}

}

}

```

SetExample.java

```

package collections;

import java.util.HashSet;

public class SetExample {

    public static void main(String[] args) {

        HashSet<String> set = new HashSet<>();

        set.add("Apple");

        set.add("Banana");

        set.add("Orange");

        set.add("Apple"); // This won't be added since sets don't allow duplicates

        // To display
    }

}

```

```
        System.out.println("Set Example:");

        for (String fruit : set) {

            System.out.println(fruit);

        }

    }

}
```

MapExample.java

```
package collections;

import java.util.HashMap;

import java.util.Map;

public class MapExample {

    public static void main(String[] args) {

        HashMap<Integer, String> map = new HashMap<>();

        map.put(1, "Apple");

        map.put(2, "Banana");

        map.put(3, "Orange");

        // To display

        System.out.println("Map Example:");

        for (Map.Entry<Integer, String> entry : map.entrySet()) {

            System.out.println(entry.getKey() + ": " + entry.getValue());

        }

    }

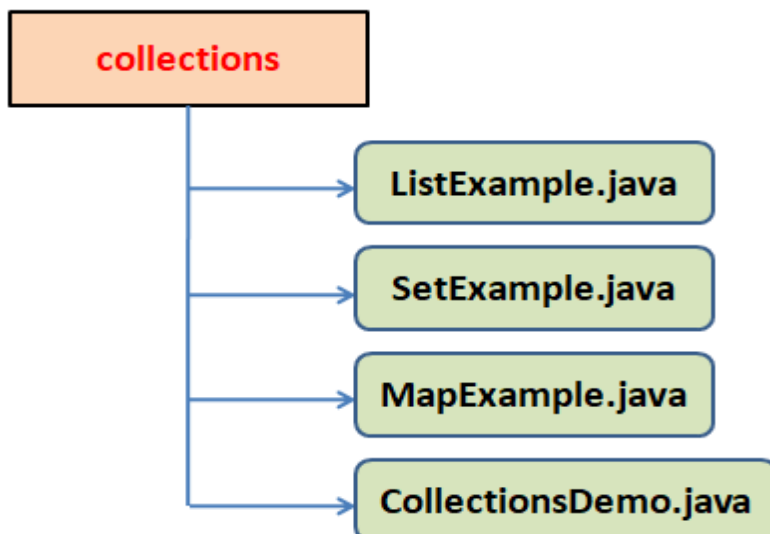
}
```

```
}
```

CollectionsDemo.java

```
package collections;  
  
public class CollectionsDemo {  
  
    public static void main(String[] args) {  
  
        ListExample.main(args);  
  
        SetExample.main(args);  
  
        MapExample.main(args);  
  
    }  
}
```

Output:



```
D:\Java>javac -d . ListExample.java
D:\Java>javac -d . SetExample.java
D:\Java>javac -d . MapExample.java
D:\Java>javac -d . CollectionsDemo.java
D:\Java>java collections.CollectionsDemo
List Example:
Apple
Banana
Orange
Set Example:
Apple
Orange
Banana
Map Example:
1: Apple
2: Banana
3: Orange
```

EXPERIMENT-6

6. Write a program to synchronize the threads acting on the same object. [Consider the example of any reservations like railway, bus, movie ticket booking, etc.]

```
public class Synchronization implements Runnable
{
    int tickets = 3;
    static int i = 1, j = 2, k = 3;
    synchronized void bookticket (String name, int wantedtickets)
    {
        if (wantedtickets <= tickets)
        {
            System.out.println (wantedtickets + " booked to " + name);
            tickets = tickets - wantedtickets;
        }
    }
}
```

```

        else
        {
            System.out.println ("No tickets to book");
        }
    }
    public void run ()
    {
        String name = Thread.currentThread ().getName ();
        if (name.equals ("t1"))
        {
            bookticket (name, i);
        }
        else if (name.equals ("t2"))
        {
            bookticket (name, j);
        }
        else
        {
            bookticket (name, k);
        }
    }
}
public static void main (String[]args)
{
    Synchronization s = new Synchronization ();
    Thread t1 = new Thread (s);
    Thread t2 = new Thread (s);
    Thread t3 = new Thread (s);
    t1.setName ("t1");
    t2.setName ("t2");
    t3.setName ("t3");
    t1.start ();
    t2.start ();
    t3.start ();
}
}

```

Output:

```

1 booked to t1
2 booked to t2
No tickets to book

```

EXPERIMENT-7

7.Write a program to perform CRUD operations on the student table in a database using JDBC.

```
import java.sql.*;

import java.util.Scanner;

public class InsertData {

    public static void main(String[] args) {

        try {

            // to create connection with database

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

            Connection con = DriverManager.getConnection("jdbc:mysql://localhost/mydb", "root", "");

            Statement s = con.createStatement();

            // To read insert data into student table

            Scanner sc = new Scanner(System.in);

            System.out.println("Inserting Data into student table : ");

            System.out.println("_____");

            System.out.print("Enter student id : ");

            int sid = sc.nextInt();

            System.out.print("Enter student name : ");

            String sname = sc.next();

            System.out.print("Enter student address : ");

            String saddr = sc.next();

            // to execute insert query

            s.execute("insert into student values("+sid+", '"+sname+"', '"+saddr+"'");

            System.out.println("Data inserted successfully into student table");
```

```

        s.close();

        con.close();

    } catch (SQLException err) {

        System.out.println("ERROR: " + err);

    } catch (Exception err) {

        System.out.println("ERROR: " + err);

    }

}

}

```

UpdateData.java

```

import java.sql.*;

import java.util.Scanner;

public class UpdateData {

    public static void main(String[] args) {

        try {

            // to create connection with database

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

            Connection con = DriverManager.getConnection("jdbc:mysql://localhost/mydb", "root", "");

            Statement s = con.createStatement();

            // To read insert data into student table

            Scanner sc = new Scanner(System.in);

            System.out.println("Update Data in student table : ");

```

```

        System.out.println("_____");

        System.out.print("Enter student id : ");

        int sid = sc.nextInt();

        System.out.print("Enter student name : ");

        String sname = sc.next();

        System.out.print("Enter student address : ");

        String saddr = sc.next();

        // to execute update query

        s.execute("update student set s_name='"+sname+"',s_address = '"+saddr+"' where s_id = "+sid);

        System.out.println("Data updated successfully");

        s.close();

        con.close();

    } catch (SQLException err) {

        System.out.println("ERROR: " + err);

    } catch (Exception err) {

        System.out.println("ERROR: " + err);

    }

}

}

```

DeleteData.java

```

import java.sql.*;

import java.util.Scanner;

public class DeleteData {

```

```
public static void main(String[] args) {  
    try {  
        // to create connection with database  
        Class.forName("com.mysql.jdbc.Driver");  
        Connection con = DriverManager.getConnection("jdbc:mysql://localhost/mydb", "root", "");  
        Statement s = con.createStatement();  
  
        // To read insert data into student table  
        Scanner sc = new Scanner(System.in);  
        System.out.println("Delete Data from student table : ");  
        System.out.println("_____");  
        System.out.print("Enter student id : ");  
        int sid = sc.nextInt();  
        // to execute delete query  
        s.execute("delete from student where s_id = "+sid);  
        System.out.println("Data deleted successfully");  
        s.close();  
        con.close();  
    } catch (SQLException err) {  
        System.out.println("ERROR: " + err);  
    } catch (Exception err) {  
        System.out.println("ERROR: " + err);  
    }  
}  
}
```

DisplayData.java

```
import java.sql.*;

import java.util.Scanner;

public class DisplayData {

    public static void main(String[] args) {

        try {

            // to create connection with database

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

            Connection con = DriverManager.getConnection("jdbc:mysql://localhost/mydb", "root", "");

            Statement s = con.createStatement();


            // To display the data from the student table

            ResultSet rs = s.executeQuery("select * from student");

            if (rs != null) {

                System.out.println("SID \t STU_NAME \t ADDRESS");

                System.out.println("_____");

                while (rs.next())

                {

                    System.out.println(rs.getString(1) + " \t " + rs.getString(2) + " \t " + rs.getString(3));

                    System.out.println("_____");

                }

                s.close();

                con.close();

            }

        }

    }

}
```

```
    }  
    } catch (SQLException err) {  
        System.out.println("ERROR: " + err);  
    } catch (Exception err) {  
        System.out.println("ERROR: " + err);  
    }  
  
    }  
}
```

Output:

```
D:\>javac InsertData.java  
  
D:\>java InsertData  
Inserting Data into student table :  
-----  
Enter student id : 502  
Enter student name : K.Kiran  
Enter student address : Nizamabad  
Data inserted successfully into student table
```

```
D:\>javac UpdateData.java  
  
D:\>java UpdateData  
Update Data in student table :  
-----  
Enter student id : 789  
Enter student name : K.Ashok  
Enter student address : Singareni  
Data updated successfully
```

```
D:\>javac DeleteData.java

D:\>java DeleteData
Delete Data from student table :

Enter student id : 542
Data deleted successfully
```

```
D:\>javac DisplayData.java

D:\>java DisplayData
```

SID	STU_NAME	ADDRESS
501	Naveen	Hyderabad
521	T.Madhu	Vizag
531	Duraga	Kolcutta
561	ShivaKum	Chennai
789	K.Ashok	Singareni
502	K.Kiran	Nizamabad

EXPERIMENT-8

8. Write a Java program that works as a simple calculator. Use a grid layout to arrange buttons for the digits and for the +, -, *, % operations. Add a text field to display the result. Handle any possible exceptions like divided by zero.

```
import java.awt.*;

import java.awt.event.*;

import java.applet.*;

/*

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

* */
```

```
public class Calculator extends Applet implements ActionListener
{
    String msg=" ";
    int v1,v2,result;
    TextField t1;
    Button b[]=new Button[10];
    Button add,sub,mul,div,clear,mod,EQ;
    char OP;
    public void init()
    {
        Color k=new Color(10,89,90);
        setBackground(k);
        t1=new TextField(50);
        GridLayout gl=new GridLayout(6,3);
        setLayout(gl);
        for(int i=0;i<10;i++)
        {
            b[i]=new Button(""+i);
        }
        add=new Button("+");
        sub=new Button("-");
        mul=new Button("*");
        div=new Button("/");
        mod=new Button("%");
        clear=new Button("Clear");
```



```
EQ=new Button("=");  
t1.addActionListener(this)  
;add(t1);  
for(int i=0;i<10;i++)  
{  
    add(b[i]);  
}  
add(add);  
add(sub);  
add(mul);  
add(div);  
add(mod);  
add(clear);  
add(EQ);  
for(int i=0;i<10;i++)  
{  
    b[i].addActionListener(this);  
}  
add.addActionListener(this);  
sub.addActionListener(this);  
mul.addActionListener(this);  
div.addActionListener(this);  
mod.addActionListener(this);  
clear.addActionListener(this);  
EQ.addActionListener(this);
```

```
}

public void actionPerformed(ActionEvent ae)

{

String str=ae.getActionCommand();

char ch=str.charAt(0);

if ( Character.isDigit(ch))

t1.setText(t1.getText()+str);

else

if(str.equals("+"))

{

v1=Integer.parseInt(t1.getText());

OP='+';

t1.setText("");

}

else if(str.equals("-"))

{

v1=Integer.parseInt(t1.getText()); OP='-';

t1.setText("");

}

else if(str.equals("*"))

{

v1=Integer.parseInt(t1.getText());

OP='*';

t1.setText("");

}

}
```

```
else if(str.equals("/"))
{
v1=Integer.parseInt(t1.getText());

OP='/';

t1.setText("");
}

else if(str.equals("%")){

v1=Integer.parseInt(t1.getText());

OP='%';

t1.setText("");
}

if(str.equals("=")){

v2=Integer.parseInt(t1.getText());

if(OP=='+')

result=v1+v2;

else if(OP=='-')

result=v1-v2;

else if(OP=='*')

result=v1*v2;

else if(OP=='/')

result=v1/v2;

else if(OP=='%')

result=v1%v2;

t1.setText(""+result);

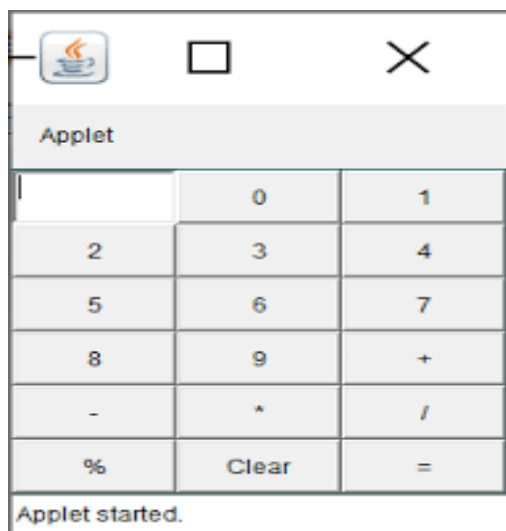
}
```

```

if(str.equals("Clear"))
{
t1.setText("");
}
}
}

```

Output:



EXPERIMENT-9

9. Write a Java program that handles all mouse events and shows the event name at the center of the window when a mouse event is fired. [Use Adapter classes]

```

import javax.swing.*;

import java.awt.*;

```

```

import javax.swing.event.*;

import java.awt.event.*;

class MouseEventPerformer extends JFrame implements MouseListener
{
    JLabel l1;

    public MouseEventPerformer()
    {
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

        setSize(300,300);

        setLayout(new FlowLayout(FlowLayout.CENTER));

        l1 = new JLabel();

        Font f = new Font("Verdana", Font.BOLD, 20);

        l1.setFont(f);

        l1.setForeground(Color.BLUE);

        add(l1);

        addMouseListener(this);

        setVisible(true);
    }

    public void mouseExited(MouseEvent m)
    {
        l1.setText("Mouse Exited");
    }

    public void mouseEntered(MouseEvent m)
    {
        l1.setText("Mouse Entered");
    }
}

```

```
}  
  
public void mouseReleased(MouseEvent m)  
{  
    l1.setText("Mouse Released");  
}  
  
public void mousePressed(MouseEvent m)  
{  
    l1.setText("Mouse Pressed");  
}  
  
public void mouseClicked(MouseEvent m)  
{  
    l1.setText("Mouse Clicked");  
}  
  
public static void main(String[] args) {  
    MouseEventPerformer mep = new MouseEventPerformer();  
}  
}
```

OUTPUT:

