MUSLIM ASSOCIATION
COLLEGE OF ENGINEERING
Venjaramoodu, Thiruvananthapuram 695607, Kerala, India Tel: +91 472 2870786 (O)
Head Office: Muslim Association, Nandavanam, Thiruvananthapuram,
695033, Kerala, India Tel: +91 47 12337629, 2322957

LAB MANUAL

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

SEMESTER 3

(2019 SCHEME)

CSL 203: OBJECT ORIENTED PROGRAMMING LAB (IN JAVA)

OBJECT ORIENTED PROGRAMMING

In Object Oriented Programming, the data is a critical element in the program development and it does not flow freely around the system. Object oriented program allows decomposing a problem into a number of entities called object and then builds data and functions around these entities. Object, Class, Inheritance, Polymorphism, Dynamic binding, Overloading, Data abstraction, Encapsulation, Modularity are the basic concept of OOP.

GENERAL CONCEPTS OF OOPS: -

OBJECT

Objects are the basic run time entities in an Object Oriented Programming. They represent a person, a place or any item that the program handles. The Object Oriented approach views a problem in terms of object involved rather than procedure for doing it. Object is an identifiable entity with some characteristics and behavior. The characteristic of an object are represented by its data and its behavior is represented by its data function associated. Therefore in OOP Programming object represents an entity that can store data and has its interface through function.

CLASS

A class is a group of objects that share a common properties and relationship. 'Object' is an instance of class. A class is a way to bind the data and its associated function together. When defining a class, we are creating a new abstract data type that can be treated like any other built-in-data type. For e.g. Bird is a class but parrot is an object.

DATA ABSTRACTION

Abstraction is the concept of simplifying a real world concept into its essential element. Abstraction refers to act representing essential features without including the background details or explanations. It creates a new data type used encapsulation items that are suited to an

application to be programmed is known as data abstraction. Data types created by the data abstraction process are known as abstract data types. For e.g. Switch board.

ENCAPSULATION

The wrapping up of data and function that operate on the data into a single unit called class is known as encapsulation. The data cannot be accessed directly. If you want to read a data, an item in an object (an instance of the class), you can call a member function in the object. It will read the item and return the value to you. The data is hidden so it is safe from accident alteration. Encapsulation is just a way to implement data abstraction class is the concept of data abstraction. They are known as abstract data type. Data types because; these can be used to create objects of its own type.

INHERITANCE

Inheritance is the capacity of one class of things to inherit capacities or properties from another class. One major reason behind this is the capability to express the inheritance relationship which makes it aware the closeness with the real world. Another reason is idea of reusability. One reason is the transitive nature. The principle behind this sort of diversion is that each subclass shares a common characteristic with the class from which it is derived. A subclass defines only those features that are unique to it.

POLYMORPHISM

Polymorphism is the ability for a message data to be processed more than one form. Polymorphism is the concept of which supports the capability of an object of a class to behave differentially in response to a message or action. Polymorphism is a property by which the same message can be sent to objects of several different classes and each object can respond in a different way depending on its class. Property by which the same message can be sent to objects of several different way depending on its class property by which the same message can

be sent to objects of several different classes and each object can respond in a different way depending on its class property.

DYNAMIC BINDING

Binding refers to the tie up of a procedure call to the address code to be executed in response to the code. Dynamic binding is done at the time of execution.

OVERLOADING

Overloading allows a function or operator to be given more than one definition. A function name having several definitions that are differentiable by the number or type of their arrangements. This not only implements polymorphism but also reduces number of comparison in a program and thereby making the program run faster.

MODULARITY

Act of representing or partitioning a program into industrial components. It is the property of a system that has been decomposed into a set of cohesive and loosed coupled modules. For eg. A complete music system comprises of speakers, cassette players, and real player. Similarly, we can divide a complex program into various modules where each module is a complete unit in itself.

JAVA FEATURES

1. Simple
2. Object-Oriented
3. Platform independent
4. Secured
5. Robust
6. Architecture neutral
7. Portable
8. Dynamic
9. Interpreted
10. High Performance
11. Multithreaded

12. Distributed

SIMPLE

According to Sun, Java language is simple because: syntax is based on C++ (so easier for programmers to learn it after C++). Removed many confusing and/or rarely-used features e.g., explicit pointers, operator overloading etc. No need to remove unreferenced objects because there is Automatic Garbage Collection in Java.

OBJECT-ORIENTED

Object-oriented means we organize our software as a combination of different types of objects that incorporates both data and behavior. Object-oriented programming(OOPs) is a methodology that simplify software development and maintenance by providing some rules.

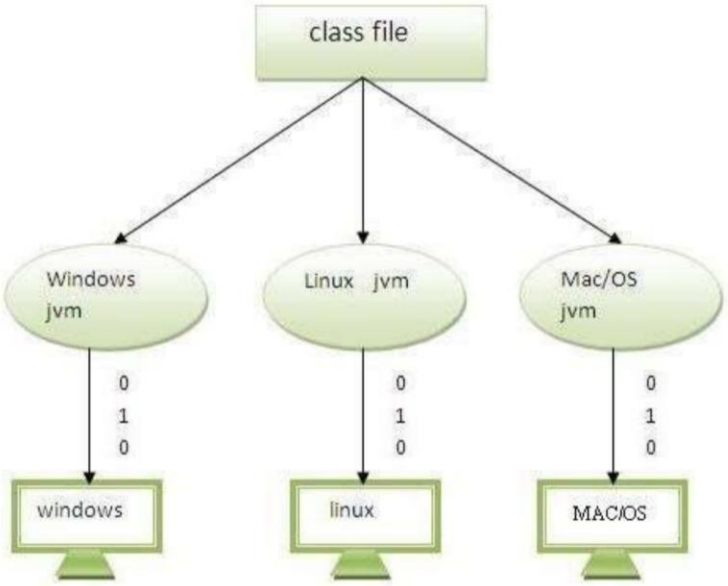
Basic concepts of OOPs are:

1. Object
2. Class
3. Inheritance
4. Polymorphism
5. Abstraction
6. Encapsulation

PLATFORM INDEPENDENT

A platform is the hardware or software environment in which a program runs. There are two types of platforms software-based and hardware-based. Java provides software-based platform. The Java platform differs from most other platforms in the sense that it's a software-based platform that runs on top of other hardware-based platforms. It has two components:

1. Runtime Environment
2. API(Application Programming Interface)

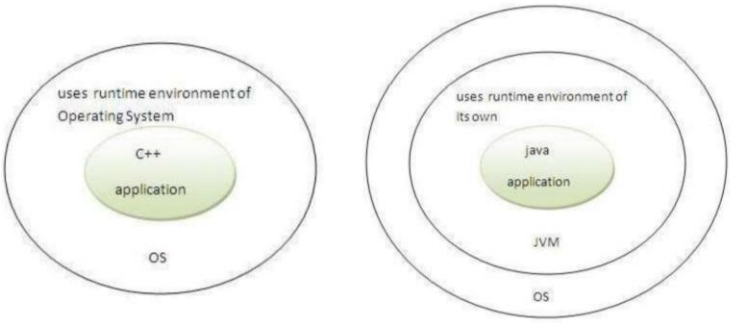


Java code can be run on multiple platforms eg. Windows,Linux etc.Java code is compiled by the compiler and converted into bytecode. This bytecode is a platform independent code because it can be run on multiple platforms i.e. Write Once and Run Anywhere(WORA).

SECURED

Java is secured because:

- No explicit pointer
- Programs run inside virtual machine sandbox.



Class loader- adds security by separating the package for the classes of the local file system from those that are imported from network sources.

Bytecode Verifier- checks the code fragments for illegal code that can violate access right to objects.

Security Manager- determines what resources a class can access such as reading and writing to the local disk.

These securities are provided by Java language. Some security can also be provided by application developer through SSL, JAAS, cryptography etc.

ROBUST

Robust simply means strong. Java uses strong memory management. There are lack of pointers that avoids security problem. There is automatic garbage collection in Java. There is exception handling and type checking mechanism in Java. All these points makes Java robust.

ARCHITECTURE-NEUTRAL

There is no implementation dependent features e.g. size of primitive types is set.

PORTABLE

We may carry the Java bytecode to any platform.

HIGH-PERFORMANCE

Java is faster than traditional interpretation since byte code is "close" to native code still somewhat slower than a compiled language (e.g., C++)

DISTRIBUTED

We can create distributed applications in Java. RMI and EJB are used for creating distributed applications. We may access files by calling the methods from any machine on the internet.

MULTI-THREADED

A thread is like a separate program, executing concurrently. We can write Java programs that deal with many tasks at once by defining multiple threads. The main advantage of multithreading is that it shares the same memory. Threads are important for multi-media, Web applications etc.

CSL 203: OBJECT ORIENTED PROGRAMMING LAB (IN JAVA)

LIST OF EXPERIMENTS

EXP NO:	EXP NAME	EXPERIMENTS
	A. BASIC PROGRAMS	
1	PALINDROME	Write a Java program that checks whether a given string is a palindrome or not. Ex: MALAYALAM is palindrome
2	FREQUENCY OF A GIVEN CHARACTER IN A STRING	Write a Java Program to find the frequency of a given character in a string.
3	MULTIPLY TWO GIVEN MATRICES	Write a Java program to multiply two given matrices.
	B. OBJECT ORIENTED PROGRAMMING CONCEPTS	
4	EMPLOYEE DETAILS	Write a Java program which creates a class named 'Employee' having the following members: Name, Age, Phone number, Address, Salary. It also has a method named 'printSalary( )' which prints the salary of the Employee. Two classes 'Officer' and 'Manager' inherits the 'Employee' class. The 'Officer' and 'Manager' classes have data members 'specialization' and 'department' respectively. Now, assign name, age, phone number,

		address and salary to an officer and a manager by making an object of both of these classes and print the same. (Exercise to understand inheritance).
5	FIND NUMBER OF SIDES IN A GIVEN GEOMETRICAL STRUCTURE	Write a java program to create an abstract class named Shape that contains an empty method named numberOfSides( ). Provide three classes named Rectangle, Triangle and Hexagon such that each one of the classes extends the class Shape. Each one of the classes contains only the method numberOfSides( ) that shows the number of sides in the given geometrical structures. (Exercise to understand polymorphism).
	C. HANDLING DIFFERENT TYPES OF FILES AS WELL AS INPUT AND OUTPUT MANAGEMENT METHODS	
6	FILE HANDLING	Write a Java program that read from a file and write to file by handling all file related exceptions.
7	STRING TOKENIZER	Write a Java program that reads a line of integers, and then displays each integer, and the sum of all the integers.

	D. EXCEPTION HANDLING AND MULTI-THREADING APPLICATIONS	
8	SHOW THE USAGE OF TRY CATCH THROWS AND FINALLY	Write a Java program that shows the usage of try, catch, throws and finally.
9	THREAD SYNCHRONIZATION.	Write a Java program that shows thread synchronization.
	E. GRAPHICS PROGRAMMING	
10	CALCULATOR	Write a Java program that works as a simple calculator. Arrange Buttons for digits and the + - * % operations properly. Add a text field to display the result. Handle any possible exceptions like divide by zero. Use Java Swing.
11	TRAFFIC LIGHT PROGRAM	Write a Java program that simulates a traffic light. The program lets the user select one of three lights: red, yellow, or green. When a radio button is selected, the light is turned on, and only one light can be on at a time. No light is on when the program starts.

	F. STANDARD SEARCHING AND SORTING ALGORITHMS USING DATA STRUCTURES AND ALGORITHMS	
12	LINKED LIST IMPLEMENTATION	Write a Java program for the following: 1) Create a doubly linked list of elements. 2) Delete a given element from the above list. 3) Display the contents of the list after deletion.
13	QUICK SORT USING JAVA	Write a Java program that implements Quick sort algorithm for sorting a list of names in ascending order.

EXPERIMENT NO: 1

PALINDROME

AIM:

To implement a Java program that checks whether a given string is a palindrome or not.

ALGORITHM

Step 1: Start
Step 2: Get the String to check for palindrome.
Step 3: Hold the String in temporary variable.
Step 4: Reverse the string
Step 5: Compare the temporary string with reversed string
Step 6: If both strings are same, print "is a palindrome "
Else print "is not a palindrome"
Step 7: Stop

PROGRAM

```
import java.util.*;
class palindrome
{
  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");
  }}
```

OUTPUT

Enter a string:

MALAYALAM

MALAYALAM is a palindrome

Enter a string:

JAVA

JAVA is not a palindrome

EXPERIMENT NO: 2

FREQUENCY OF A GIVEN CHARACTER IN A STRING

AIM:

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

ALGORITHM:

Step 1: Start
Step 2: Input the string from the user and store in variable str.
Step 3: Input the character whose frequency needs to be computed. Store it in variable ch
Step 4: Initialize count to 0. Initialize the i variable to 0
Step 5:Repeat the steps 5 to 7 until the start index 'i' is greater than str.length().
Step 6: If str.charAt(i) is the same as character in ch, then count=count+1.
Step 7: increment i by 1
Step 8: Print Frequency of the character in string is value in count.
Step 9: Stop

PROGRAM

```
import java.util.*;
class frequency
 {
   public static void main(String args[])
    {
       Scanner sc = new Scanner(System.in);
       System.out.print("Enter the String:");
       String str = sc.nextLine();
       System.out.print("Enter the character:");
       char ch = sc.nextLine().charAt(0);
       int count = 0;
```

```
        for(int i=0;i<str.length();i++)
          {
           if(str.charAt(i) == ch)
            {
             count++;
            }
          }
    System.out.println("Count of occurrence of "+ ch +"="+count);
    }
}
```

OUTPUT

Enter the String: Computer Science
Enter the character: e
Count of occurrence of e=3

EXPERIMENT NO: 3

MULTIPLY TWO GIVEN MATRICES

AIM:

To implement a Java program to multiply two given matrices.

ALGORITHM:

Step 1: Start
Step 2:Enter the number of rows m1 and column n1 in first matrix
Step 3: Enter the number of rows m2 and column n2 in second matrix
Step 4: If n1 is not equal to m2 then print Matrix Multiplication is not possible. Go to step 14
Step 5: Input: matrices A and B
Step 6:Let C be a new matrix of the appropriate size
Step 7: For i from 1 to n: do the steps from 7 to 11
Step 8: For j from 1 to p do the steps from 8 to 11
Step 9:Let $C_{ij} = 0$
Step 10: For k from 1 to m do the steps from10 to 11
Step 11:Set $C_{ij} \leftarrow C_{ij} + A_{ik} \times B_{kj}$
Step 13:Print the product matrix C_{ij}
Step 14: Stop

PROGRAM

```
import java.util.*;
class multiplication
{
   public static void main(String args[])
   {
      Scanner sc = new Scanner(System.in);
      System.out.print("Enter the order - m1:");
      int m1 = sc.nextInt();
      System.out.print("Enter the order - n1:");
      int n1 = sc.nextInt();
      System.out.print("Enter the order - m2:");
      int m2 = sc.nextInt();
```

```
System.out.print("Enter the order - n2:");
int n2 = sc.nextInt();
if(n1 != m2){
System.out.println("Matrix Multiplication not Possible");
return;
}
int A[][] = new int[m1][n1];
int B[][] = new int[m2][n2];
int C[][] = new int[m1][n2];
System.out.println("Read Matrix A");
for(int i=0;i<m1;i++){
for(int j=0;j<n1;j++){
System.out.print("A["+i+"]["+j+"]=");
A[i][j] = sc.nextInt();
}}
System.out.println("Read Matrix B");
for(int i=0;i<m2;i++){
for(int j=0;j<n1;j++){
System.out.print("B["+i+"]["+j+"]=");
B[i][j] = sc.nextInt();
}}
for(int i=0;i<m1;i++){
for(int j=0;j<n2;j++){
C[i][j]=0;
for(int k=0;k<n1;k++){
C[i][j] += A[i][k] * B[k][j];
} } }
System.out.println("Matrix A");
for(int i=0;i<m1;i++){
for(int j=0;j<n1;j++){
System.out.print(A[i][j]+"\t");
}
System.out.println();        }
System.out.println("Matrix B");
for(int i=0;i<m2;i++){
```

```
for(int j=0;j<n2;j++){
System.out.print(B[i][j]+"\t");
}
System.out.println();
}
System.out.println("Matrix C");
for(int i=0;i<m1;i++){
for(int j=0;j<n2;j++){
System.out.print(C[i][j]+"\t");
}System.out.println();       } }}
```

OUTPUT

```
Enter the order - m1:2
Enter the order - n1:2
Enter the order - m2:2
Enter the order - n2:2
Read Matrix A
A[0][0]=1
A[0][1]=2
A[1][0]=3
A[1][1]=4
Read Matrix B
B[0][0]=1
B[0][1]=1
B[1][0]=2
B[1][1]=3
Matrix A
1       2
3       4
Matrix B
1       1
2       3
Matrix C
5       7
11      15
```

EXPERIMENT NO: 4

EMPLOYEE DETAILS

AIM:

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

ALGORITHM:

Step 1: Start

Step 2: Create a class Employee with given variables and method-print-salary() to print the salary.

Step 3: Create two subclasses of class Employee named Officer and Manager. Include the variable 'specialization' in Officer class and ‘department’ in Manager class.

Step 4 : Create a class with the main function. Inside the main function do the following steps.

Step 5: Create an object of Officer class and assign appropriate values to the variables.

Step 6: Create an object of Manager class and assign appropriate values to the variables.

Step 7: Print the values of variables using objects.

Step 8: Invoke the function print-salary() using objects to print the salary.

Step 9: Stop

PROGRAM

```
import java.util.*;
class Employee{
private String name;
private int age;
private long phoneNumber;
private String address;
private double salary;
public void setName(String name){
this.name = name;
}
public void setAge(int age){
this.age=age;
}
public void setPhoneNumber(long phoneNumber){
this.phoneNumber = phoneNumber;
}
public void setAddress(String address){
this.address = address;
}
public void setSalary(double salary){
this.salary = salary;
}
public double printSalary(){
return salary;
}
public String getName(){
return name;
}
public int getAge(){
```

```
return age;
}
public String getAddress(){
return address;
}
public long getPhoneNumber(){
return phoneNumber;
}
}
class Officer extends Employee{
private String specialization;
private String department;
public void setSpecialization(String specialization){
this.specialization = specialization;
}
public void setDepartment(String department){
this.department = department;
}
public String getDepartment(){
return department;
}
public String getSpecialization(){
return specialization;
}
}
class Manager extends Employee{
private String specialization;
private String department;
public void setSpecialization(String specialization){
this.specialization = specialization;
}
public void setDepartment(String department){
this.department = department;
}
public String getDepartment(){
```

```
return department;
}
public String getSpecialization(){
return specialization;
}
}
class inheritance{
public static void main(String args[]){
Scanner sc = new Scanner(System.in);
Officer o = new Officer();
System.out.println("Enter the officer's Detail");
System.out.print("Name:");
o.setName(sc.nextLine());
System.out.print("Address:");
o.setAddress(sc.nextLine());
System.out.print("Specialization:");
o.setSpecialization(sc.nextLine());
System.out.print("Department:");
o.setDepartment(sc.nextLine());
System.out.print("Age:");
o.setAge(sc.nextInt());
System.out.print("Number:");
o.setPhoneNumber(sc.nextLong());
System.out.print("Salary:");
o.setSalary(sc.nextDouble());
sc.nextLine(); // to skip new Line
System.out.println("The officer Detail");
System.out.println("Name:"+o.getName());
System.out.println("Age:"+o.getAge());
System.out.println("Number:"+o.getPhoneNumber());
System.out.println("Address:"+o.getPhoneNumber());
System.out.println("Salary:"+o.printSalary());
System.out.println("Specialization:"+o.getSpecialization());
System.out.println("Department:"+o.getDepartment());
Manager m = new Manager();
```

```
System.out.println("Enter the manager's Detail");
System.out.print("Name:");
m.setName(sc.nextLine());
System.out.print("Address:");
m.setAddress(sc.nextLine());
System.out.print("Specialization:");
m.setSpecialization(sc.nextLine());
System.out.print("Department:");
m.setDepartment(sc.nextLine());
System.out.print("Age:");
m.setAge(sc.nextInt());
System.out.print("Number:");
m.setPhoneNumber(sc.nextLong());
System.out.print("Salary:");
m.setSalary(sc.nextDouble());
sc.nextLine(); // to skip new Line
System.out.println("The manager's Detail");
System.out.println("Name:"+m.getName());
System.out.println("Age:"+m.getAge());
System.out.println("Number:"+m.getPhoneNumber());
System.out.println("Address:"+m.getPhoneNumber());
System.out.println("Salary:"+m.printSalary());
System.out.println("Specialization:"+m.getSpecialization());
System.out.println("Department:"+m.getDepartment());
}}
```

OUTPUT

```
Enter the officer's Detail
Name:ARUN S R
Address:AAAAAA
Specialization:Officer
Department:CSE
Age:23
Number:9823345678
Salary:23000
```

The officer Detail
Name:ARUN S R
Age:23
Number:9823345678
Address:9823345678
Salary:23000.0
Specialization:Officer
Department:CSE
Enter the manager's Detail
Name:ARYAN
Address:rrrrr123
Specialization:Manager
Department:CSE
Age:30
Number:9456578213
Salary:400000
The manager's Detail
Name:ARYAN
Age:30
Number: 9456578213
Address:50000
Salary:40000.0
Specialization:Manager
Department:CSE

EXPERIMENT NO:5

FIND NUMBER OF SIDES IN A GIVEN GEOMETRICAL STRUCTURE

AIM:

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

ALGORITHM:

Step 1: Start

Step 2:Create class Shape with method numberOfSides( )without definition.

Step 3:Create three subclasses of Shape namely Rectangle, Triangle and Hexagon .

Step 4:Define the method numberOfSides( )in all these subclasses Rectangle, Triangle and Hexagon that displays the sides as 4,3,6 respectively.

Step 5: Invoke numberOfSides( ) by creating reference variables of class Shape which refers to each of these subclasses.

Step 6: Stop

PROGRAM:

```
abstract class Shape
  {
    abstract void numberOfSides();
  }

class Rectangle extends Shape
  {
      void numberOfSides()
            {
          System.out.println("Number of Sides of Rectangle is: 4");
            }
 }
class Triangle extends Shape
   {
       void numberOfSides()
     {
  System.out.println("Number of Sides of Triangle Shape is: 3");
  }
}

class Hexagon extends Shape
 {
    void numberOfSides()
       {
  System.out.println("Number of Sides of Hexagon Shape is: 6");
  }
}
public class inheritance
  {
  public static void main(String args[])
     {
          Rectangle R = new Rectangle();
          R.numberOfSides();
```

```
        Triangle T = new Triangle();
        T.numberOfSides();

        Hexagon H = new Hexagon();
        H.numberOfSides();
  }
}
```

OUTPUT:

Number of Sides of Rectangle is: 4

Number of Sides of Triangle Shape is: 3

Number of Sides of Hexagon Shape is: 6

EXPERIMENT NO: 6
FILE HANDLING

AIM:

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

ALGORITHM:

STEP 1: Start
STEP 2: Declare the variables.
STEP 3: Get the file name to read.
STEP 4: Using FileInputStream check whether the file exist.
STEP 5: If the file exists then check for the end of file condition.
STEP 6: Read the contents of the file.
STEP 7: Print the contents of the file.
STEP 8: Stop.

PROGRAM:

```
import java.io.*;
class Filecopy
{ public static void main(String a[]) throws IOException
{
FileInputStream f1=null;
FileOutputStream f2=null;
try
{
f1= new FileInputStream("test1.txt");
f2= new FileOutputStream("cp.txt");
int c;
do
{
c=f1.read();
if(c!=-1)
{
f2.write((char)c);
System.out.print((char)c);
```

```
}
}while(c!=-1);
}
catch(FileNotFoundException e)
        {
System.out.println("File not found");
return;    }
f1.close();
f2.close();
}
}
```

OUTPUT:

We already created a file test1.txt with content "Hello world "

after execution of program a new file is generated with name new cp.txt with contents

readed rom test1.txt.

"Hello world"

EXPERIMENT NO: 7
STRING TOKENIZER

AIM:

To implement a Java program that reads a line of integers, and then displays each integer, and the sum of all the integers.

ALGORITHM:

Step 1: Start

Step 2: Import all classes of package java.util.

Step 3: Create object of stringTokenizer class using constructor with delimiter argument as space for separating tokens.

Step 4:Enter numbers separated by space.

Step 5: Using appropriate methods, separate tokens (integers)

Step 6: Display the tokens and find the sum of all integers.
Step 7: Stop

PROGRAM:

```
import java.util.*;
 class StringTokenizerDemo {
    public static void main(String args[]) {
        int n;
        int sum = 0;

         Scanner sc = new Scanner(System.in);
        System.out.println("Enter integers with one space gap:");
        String s = sc.nextLine();

         StringTokenizer st = new StringTokenizer(s, " ");
        while (st.hasMoreTokens()) {
            String temp = st.nextToken();
            n = Integer.parseInt(temp);
```

```
            System.out.println(n);
            sum = sum + n;
        }
        System.out.println("sum of the integers is: " + sum);
        sc.close();
    }
}
```

OUTPUT:

```
Enter integers with one space gap:
5 10 15 20
5
10
15
20
sum of the integers is: 50
```

EXPERIMENT NO: 8
SHOW THE USAGE OF TRY CATCH THROWS AND FINALLY

AIM:
To implement a Java program that shows the usage of try, catch, throws and finally.

ALGORITHM:
Step 1: Start
Step 2: The exception occurs when an integer is divided by zero.
Step 3: the code will be written within try block.
Step 4: if such an exception occurs then will be catch in catch block by displaying the statement “Exception Occurred”.
Step 5: Stop

PROGRAM:
```
import java.util.Scanner;
class exception_handling{
        public static void main(String args[]){
                Scanner sc = new Scanner(System.in);
                try{
                      System.out.println("Program to perform Division");
                        System.out.print("Enter Number-1:");
                        int a = sc.nextInt();
                        System.out.print("Enter Number-2:");
                        int b = sc.nextInt();
                        int c = a/b;
                        System.out.println("Result="+c);
                }
                catch(ArithmeticException e)
                  {
                        System.out.println(“Exception Occurrred”);
                }
                finally{
```

```
                System.out.println("End of Operation");
            }
        }
}
```

OUTPUT:

```
Program to perform Division
Enter Number-1:120
Enter Number-2:0
Exception Occurrred
End of Operation
```

EXPERIMENT NO: 9
THREAD SYNCHRONIZATION

AIM:

To implement a Java program that shows thread synchronization.

ALGORITHM:

Step 1: Start.

Step 2: Create a class named Table.

Step 3: Create a synchronized method named printTable() to print the multiplication table.

Step 4: Create a thread named MyThread1that extends Thread class.

Step 5: In MyThread1 class, define the run() to print the print the multiplication table of 5.

Step 6: Create a thread named MyThread2that extends Thread class.

Step 7: In MyThread2 class, define the run() to print the print the multiplication table of 100.

Step 8 : Create a class with main function and do the following steps:

Step 9: Create an object of Table class and pass the object to the Thread classes.

Step 10: Invoke the start() method.

Step 11: Stop.

PROGRAM:

```
class Table{
  synchronized void printTable(int n){//method not synchronized
   for(int i=1;i<=5;i++){
     System.out.println(n*i);
      } }}
class MyThread1 extends Thread{
Table t;
MyThread1(Table t){
this.t=t;
}
public void run(){
t.printTable(5);
} }
class MyThread2 extends Thread{
Table t;
MyThread2(Table t){
this.t=t;
}
public void run(){
t.printTable(100);
}  }
class Synchronization{
public static void main(String args[]){
Table obj = new Table();//only one object
MyThread1 t1=new MyThread1(obj);
MyThread2 t2=new MyThread2(obj);
t1.start();
t2.start();
} }
```

OUTPUT:

5
10
15
20
25
100
200
300
400
500

EXPERIMENT NO: 10
CALCULATOR

AIM:

To implement a Java program that works as a simple calculator. Arrange Buttons for digits and the + - * % operations properly. Add a text field to display the result. Handle any possible exceptions like divide by zero. Use Java Swing.

ALGORITHM:

Step 1: Start.

Step 2: Create a class named Calcnew that implements ActionListener.

Step 3: Declare JFrame, JTextField and JButton classes and declare variables a, b, result, operator.

Step 4: Initialize result and operator to 0.

Step 5: Create 17 buttons and one text box for getting the result.

Step 6: Align the components to the correct positions by using the setBounds() method.

Step 7: Add the components in the Frame using the add() method.

Step 8: Set the Frame using setLayout() to be null, setVisible() to true, setSize() and setDefaultCloseOperation().

Step 9: Register the sources using the method addActionListener().

Step 10: Perform all the actions using the actionPerformed() method.

If source is button b1 then concat “1” in textfield and so on.

If source is badd,bsub,bmul,or bdiv,get the value in variable "a" and also set the value of operator accordingly.

Step 11: If getSource() is beq button, then get the value in variable "b".

Step 12: Base on the value of operator, perform addition, subtraction, multiplication or division and store it in variable result.

Step 13: Create a class with main function and invoke the constructor Calcnew().

Step 14: Stop.

PROGRAM

```
import javax.swing.*;
import java.awt.event.*;
 class Calcnew implements ActionListener
{
        JFrame f;
        JTextField t;
JButton
b1,b2,b3,b4,b5,b6,b7,b8,b9,b0,bdiv,bmul,bsub,badd,bdec,beq,bclr;

        static double a=0,b=0,result=0;
        static int operator=0;
        Calcnew()
        {
                f=new JFrame("Calculator");
                t=new JTextField();
                b1=new JButton("1");
                b2=new JButton("2");
                b3=new JButton("3");
                b4=new JButton("4");
```

```
b5=new JButton("5");
b6=new JButton("6");
b7=new JButton("7");
b8=new JButton("8");
b9=new JButton("9");
b0=new JButton("0");
bdiv=new JButton("/");
bmul=new JButton("*");
bsub=new JButton("-");
badd=new JButton("+");
bdec=new JButton(".");
beq=new JButton("=");
bclr=new JButton("Clear");
t.setBounds(30,40,280,30);
b7.setBounds(40,100,50,40);
b8.setBounds(110,100,50,40);
b9.setBounds(180,100,50,40);
bdiv.setBounds(250,100,50,40);
b4.setBounds(40,170,50,40);
b5.setBounds(110,170,50,40);
b6.setBounds(180,170,50,40);
bmul.setBounds(250,170,50,40);
b1.setBounds(40,240,50,40);
b2.setBounds(110,240,50,40);
b3.setBounds(180,240,50,40);
bsub.setBounds(250,240,50,40);
bdec.setBounds(40,310,50,40);
b0.setBounds(110,310,50,40);
beq.setBounds(180,310,50,40);
badd.setBounds(250,310,50,40);
bclr.setBounds(100,380,100,40);

f.add(t);
f.add(b7);
f.add(b8);
```

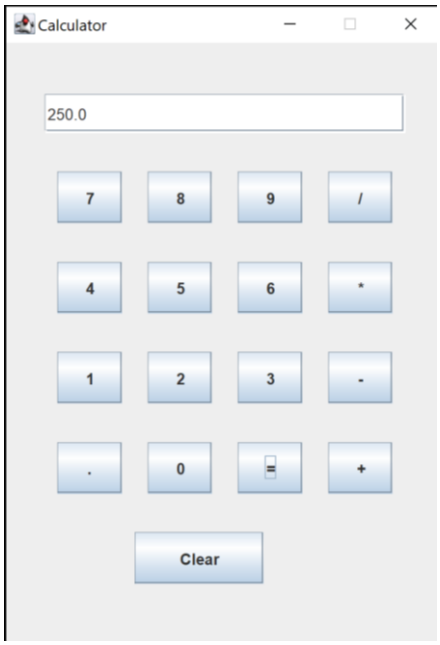
```
        f.add(b9);
        f.add(bdiv);
        f.add(b4);
        f.add(b5);
        f.add(b6);
        f.add(bmul);
        f.add(b1);
        f.add(b2);
        f.add(b3);
        f.add(bsub);
        f.add(bdec);
        f.add(b0);
        f.add(beq);
        f.add(badd);
        f.add(bclr);

        f.setLayout(null);
        f.setVisible(true);
        f.setSize(350,500);

    f.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        f.setResizable(false);

        b1.addActionListener(this);
        b2.addActionListener(this);
        b3.addActionListener(this);
        b4.addActionListener(this);
        b5.addActionListener(this);
        b6.addActionListener(this);
        b7.addActionListener(this);
        b8.addActionListener(this);
        b9.addActionListener(this);
        b0.addActionListener(this);
        badd.addActionListener(this);
        bdiv.addActionListener(this);
```

```
        bmul.addActionListener(this);
        bsub.addActionListener(this);
        bdec.addActionListener(this);
        beq.addActionListener(this);
        bclr.addActionListener(this);
}
public void actionPerformed(ActionEvent e)
{
        if(e.getSource()==b1)
                t.setText(t.getText().concat("1"));

        if(e.getSource()==b2)
                t.setText(t.getText().concat("2"));

        if(e.getSource()==b3)
                t.setText(t.getText().concat("3"));

        if(e.getSource()==b4)
                t.setText(t.getText().concat("4"));

        if(e.getSource()==b5)
                t.setText(t.getText().concat("5"));

        if(e.getSource()==b6)
                t.setText(t.getText().concat("6"));

        if(e.getSource()==b7)
                t.setText(t.getText().concat("7"));

        if(e.getSource()==b8)
                t.setText(t.getText().concat("8"));

        if(e.getSource()==b9)
                t.setText(t.getText().concat("9"));
```

```
if(e.getSource()==b0)
        t.setText(t.getText().concat("0"));

if(e.getSource()==bdec)
        t.setText(t.getText().concat("."));

if(e.getSource()==badd)
{
        a=Double.parseDouble(t.getText());
        operator=1;
        t.setText("");
}
if(e.getSource()==bsub)
{
        a=Double.parseDouble(t.getText());
        operator=2;
        t.setText("");
}
if(e.getSource()==bmul)
{
        a=Double.parseDouble(t.getText());
        operator=3;
        t.setText("");
}
if(e.getSource()==bdiv)
{
        a=Double.parseDouble(t.getText());
        operator=4;
        t.setText("");
}
if(e.getSource()==beq)
{
        b=Double.parseDouble(t.getText());

        switch(operator)
```

```
                {
                        case 1: result=a+b;
                                break;

                        case 2: result=a-b;
                                break;

                        case 3: result=a*b;
                                break;

                        case 4: result=a/b;
                                break;

                        default: result=0;
                }

                t.setText(""+result);
        }
        if(e.getSource()==bclr)
                t.setText("");
    }

    public static void main(String args[])
    {
        new Calcnew();
    }
}
```

OUTPUT:



EXPERIMENT NO: 11
TRAFFIC LIGHT PROGRAM

AIM:

To implement a Java program that simulates a traffic light. The program lets the user select one of three lights: red, yellow, or green. When a radio button is selected, the light is turned on, and only one light can be on at a time. No light is on when the program starts.

ALGORITHM:

Step 1: Start.

Step 2: Create a class named TrafficLight that extends JPanel that implements ActionListener.

Step 3: Declare three JRadioButtons for red(r1), green(r2) and orange(r3) color.

Step 4: Align the components to the correct positions by using the setBounds() method.

Step 5: Add the components using the add() method.

Step 6: Register the sources using the method addActionListener().

Step 7: Perform all the actions using the actionPerformed() method.

If source is r1 then set the background color to red using Color.red. Silmilarly for r2 and r3.

Step 8: Invoke the various graphics method in paintComponent() method.

Step 9: Create a class with main function and create an object of TrafficLight class and assign appropriate values to the variables.

Step 10: Add the component in Frame.

Step 11: Stop.

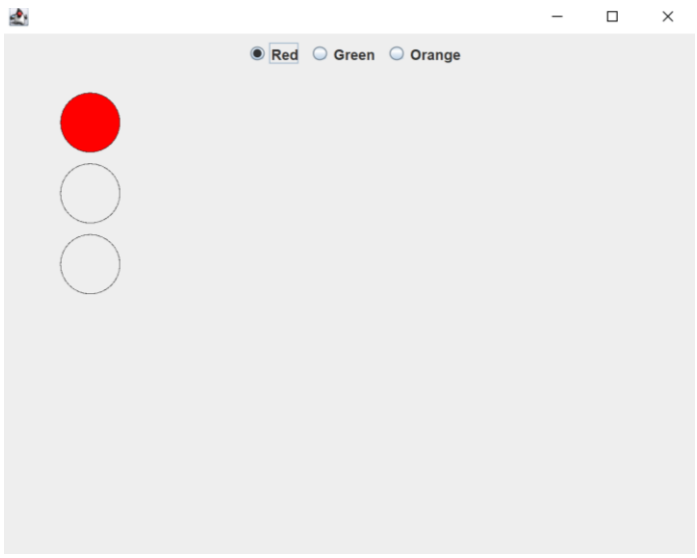
PROGRAM

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
class TrafficLight extends JPanel implements ActionListener{
private JRadioButton r1;
private JRadioButton r2;
private JRadioButton r3;
private Color red_c;
private Color green_c;
private Color orange_c;
public TrafficLight(){
setBounds(0,0,600,480);
r1 = new JRadioButton("Red");
r2 = new JRadioButton("Green");
r3 = new JRadioButton("Orange");
ButtonGroup group = new ButtonGroup();
r1.setSelected(true);
group.add(r1);
group.add(r2);
group.add(r3);
add(r1);
add(r2);
add(r3);
red_c = Color.red;
green_c = getBackground ();
orange_c = getBackground();
r1.addActionListener(this);
r2.addActionListener(this);
r3.addActionListener(this);
}
public void actionPerformed(ActionEvent e){
```

```
if(r1.isSelected() == true){
red_c = Color.red;
green_c = getBackground ();
orange_c = getBackground();
}
else if(r2.isSelected() == true){
red_c = getBackground ();
green_c = Color.green;
orange_c = getBackground();
}
else if(r3.isSelected() == true){
red_c = getBackground ();
green_c = getBackground();
orange_c = Color.orange;
}
repaint();
}
public void paintComponent(Graphics g){
super.paintComponent(g);
g.drawOval(50,50,50,50);
g.drawOval(50,110,50,50);
g.drawOval(50,170,50,50);
g.setColor(red_c);
g.fillOval(50,50,50,50);
g.setColor(orange_c);
g.fillOval(50,110,50,50);
g.setColor(green_c);
g.fillOval(50,170,50,50);
}
}
class Test1{
public static void main(String args[]){
JFrame f1 = new JFrame();
f1.setVisible(true);
f1.setSize(600,480);
```

```
f1.setLayout(null);
TrafficLight t = new TrafficLight();
f1.add(t);
}
}
```

OUTPUT:



EXPERIMENT NO: 12
LINKED LIST IMPLEMENTATION

AIM:

To implement a Java program for the following:

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

ALGORITHM:

Step 1: Start

Step 2: Create a class Node which has two attributes: data and link. link is a pointer to the next node.

Step 3: insert() will add a new node to the list: Create a new node. It first checks, whether the head is equal to null which means the list is empty. If the list is not empty, the new node will be added to end of the list such that tail's next will point to the newly added node.

Step 4: delete() will delete a given element from the given list.

Step 5: display() will display the nodes present in the list.

Step 6: Stop

PROGRAM:

```
import java.util.Scanner;
class LinkedList{
        private Node head;
        class Node{
                private int data;
                private Node link;
                public Node(int data){
                        this.data = data;
                        this.link = null;
                }       }
        public void insert(int data){
                Node temp = new Node(data);
                if(head == null){
                        head = temp;}else{
```

```
                Node ptr = head;
                while(ptr.link != null){
                        ptr = ptr.link;
                }
                ptr.link = temp;
        }
    }
    public void delete(){
        int x = head.data;
        head = head.link;
        System.out.println("Element "+x +" got deleted");
    }
    public void display(){
        if(head == null)
            System.out.println("List is Empty");
        else{
            Node ptr = head;
            while(ptr != null){
                System.out.print(ptr.data +"\t");
                ptr = ptr.link;
            }
            System.out.println();
        }
    }

}
class Main{
    public static void main(String [] args){
        LinkedList list = new LinkedList();
        Scanner sc = new Scanner(System.in);
        String choice = "";
        while(!choice.equals("4")){
            System.out.print("1. Insert at End \n2. Delete From
Front \n3. Display \n4.Exit\n");
            System.out.println("Enter the choice:");
```

```
                choice = sc.nextLine();
                switch(choice){
            case "1": System.out.print("Enter the number to insert:");
                                int data = sc.nextInt();
                                sc.nextLine();
                                list.insert(data);
                                System.out.println("Data
inserted Successfully");
                                break;
                        case "2": list.delete();
                                break;
                        case "3": list.display();
                                break;
                        case "4": break;
                        default: System.out.println("Invalid Choice");
                    }
            }

      }}
```

OUTPUT:

```
1. Insert at End
2. Delete From Front
3. Display
4.Exit
Enter the choice:
1
Enter the number to insert:10
Data inserted Successfully
1. Insert at End
2. Delete From Front
3. Display
4.Exit
Enter the choice:
```

```
1
Enter the number to insert:20
Data inserted Successfully
1. Insert at End
2. Delete From Front
3. Display
4.Exit
Enter the choice:
1
Enter the number to insert:30
Data inserted Successfully
1. Insert at End
2. Delete From Front
3. Display
4.Exit
Enter the choice:
3
10     20     30
1. Insert at End
2. Delete From Front
3. Display
4.Exit
Enter the choice:
2
Element 10 got deleted
1. Insert at End
2. Delete From Front
3. Display
4.Exit
Enter the choice:
3
20     30
1. Insert at End
2. Delete From Front
3. Display
```

```
4.Exit
Enter the choice:
3 2
Element 20 got deleted
1. Insert at End
2. Delete From Front
3. Display
4.Exit
Enter the choice:
2
Element 30 got deleted
1. Insert at End
2. Delete From Front
3. Display
4.Exit
Enter the choice:
3
List is Empty
1. Insert at End
2. Delete From Front
3. Display
4.Exit
Enter the choice:
```

EXPERIMENT NO: 13
QUICK SORT USING JAVA

AIM:

To implement a Java program that implements Quick sort algorithm for sorting a list of names in ascending order.

ALGORITHM:

Step 1: Start.

Step 2: Create a class named variables with main function.

Step 3: Create a class named QuickSortOnStrings and using the object of this class to assign values to the variables.

Step 3: Declare variable length and array for storing the names.

Step 4: Inialize array with the names to be sorted.

Ste 5: Declare a method named quicksort() and perform the following:

Step 1 − Make any element as pivot

Step 2 − Partition the array on the basis of pivot

Step 3 − Apply quick sort on left partition recursively

Step 4 − Apply quick sort on right partition recursively.

Step 6: Stop

PROGRAM

```
public class Main {
String names[];
int length;
public static void main(String[] args) {
Main  obj = new Main();
```

```
String stringsList[] = {"Raja", "Gouthu", "Rani", "Gouthami",
"Honey", "Heyaansh", "Hello"};
obj.sort(stringsList);
for (String i : stringsList) {
System.out.print(i);
System.out.println();
}
}
void sort(String array[]) {
if (array == null || array.length == 0) {
return;
}
this.names = array;
this.length = array.length;
quickSort(0, length - 1);
}
void quickSort(int lowerIndex, int higherIndex) {
int i = lowerIndex;
int j = higherIndex;
String pivot = this.names[lowerIndex + (higherIndex - lowerIndex) /
2];
while (i <= j) {
while (this.names[i].compareToIgnoreCase(pivot) < 0) {
i++;
}
while (this.names[j].compareToIgnoreCase(pivot) > 0) {
j--;
}

if (i <= j) {
exchangeNames(i, j);
i++;
j--;
}
}
```

```
if (lowerIndex < j) {
quickSort(lowerIndex, j);
}
if (i < higherIndex) {
quickSort(i, higherIndex);
}
}
void exchangeNames(int i, int j) {
String temp = this.names[i];
this.names[i] = this.names[j];
this.names[j] = temp;
}
}
```

OUTPUT:

```
Gouthami
Gouthu
Hello
Heyaansh
Honey
Raja
Rani
```