GH Raisoni College of Engineering and Management, Pune

(An Empowered Autonomous Institute Affiliated to Savitribai Phule Pune University)

Computer Engineering Department

Value Added Certification Course in IoS and Android Development

Laboratory Manual

Subject: Core Java and UI/UX using XML

(With effect from Academic Year 2024-25)

Evaluation Scheme

Class: SY B.Tech	**Sem :** III
Exam Scheme : Internal Practical : 25 Marks External Practical : 25 Marks	

Sr.No	Name of the Experiment
1	Write some simple programs in Java such as: i) To find factorial of number. ii) To display first 50 prime numbers. iii) To find sum and average of N numbers
2	Write a program in Java to implement a Calculator with simple arithmetic operations
3	Write a program in Java with class Rectangle with the data fields width, length, area and colour.The length, width and area are of double type and colour is of string type. The methods are get_length (), get_width (), get_colour () and find_area (). Create two objects of Rectangle and compare their area and colour. If the area and colour both are the same for the objects then display " Matching Rectangles", otherwise display " Non-matching Rectangle"
4	Write Programs in Java to sort i) List of integer's ii) List of names. The objective of this assignment is to learn Arrays and Strings in Java
5	Write a Program in Java to add two matrices. The objective of this assignment is to learn Arrays in Java
6	Write a program in Java to create a player class. Inherit the classes Cricket_player, Football_player and Hockey_player from player class. The objective of this assignment is to learn the concepts of inheritance in Java
7	Write a Java program which imports user defined package and uses members of the classes contained in the package
8	Write a java program which use try and catch for exception handling.
9	Write a program to demonstrate status of key on an Applet window such as KeyPressed, KeyReleased, KeyUp, KeyDown.
10	Write a program to create a frame using AWT. Implement mouseClicked, mouseEntered() and mouseExited() events. Frame should become visible when the mouse enters it.

Subject Teacher **HOD**

Experiment No: 1		
Aim: Write some simple programs in Java such as: i) To find factorial of number. ii) To display first 50 prime numbers. iii) To find sum and average of N numbers		

Objective: To understand the basic program concept in Java Programming

Outcome: Students will able to understand the about Java run time environment, various IDE used for Java programming and programming structure of Java Programming.

Pre-requisite: Computer with JDK installed and any IDE for Java Programming

Theory:

To write any program in java, first we should understand the logic behind the problem statement. In this experiment we are writing java program for three different problem statements.

1. To find factorial of number:

Factorial of a whole number 'n' is defined as the product of that number with every whole number less than or equal to 'n' till 1. For example, the factorial of 4 is 4 × 3 × 2 × 1, which is equal to 24. It is represented using the symbol '!' So, 24 is the value of 4!

2. To display first 50 prime numbers.

Prime numbers are the numbers that have only two factors, that are, 1 and the number itself. Consider an example of number 5, which has only two factors 1 and 5. This means it is a prime number. Let us take another example of the number 6, which has more than two factors, i.e., 1, 2, 3, and 6. This means 6 is not a prime number. Now, if we take the example of the number 1, we know that it has only one factor. So, it cannot be a prime number as a prime number should have exactly two factors. This means 1 is neither a prime nor a composite number, it is a unique number.

3. To find sum and average of N numbers

The average of a set of numbers is simply the sum of the numbers divided by the total number of values in the set. For example, suppose we want the average of 24, 55, 17, 87 and 100. Simply find the sum of the numbers: 24 + 55 + 17 + 87 + 100 = 283 and divide by 5 toget 56.6.

Java Program to find factorial of number

```java
import java.util.*;
class Fact
{
    public static void main(String[] args)
    {
        int fact=1;
        Scanner sc= new Scanner (System.in);
        System.out.print ("Enter the number: ");
        int n=sc.nextInt();
        for(int i=1;i<=n;i++)
        {
            fact=fact*i;
        }
        System.out.print ("Factorial: "+fact);
    }
}
```

Output:

```
Enter the number: 5

Factorial: 120
```

Java Program to display first 50 prime numbers

```java
public class Prime {

   public static void main(String[] args) {

    int num = 50, count;

    for (int i = 1; i <= num; i++) {
     count = 0;
     for (int j = 2; j <= i / 2; j++) {
      if (i % j == 0) {
       count++;
       break;
      }
     }

     if (count == 0) {
      System.out.println (i);
     }
    }
   }
  }
```

Output:

```
1
2
3
5
7
11
13
17
19
23
29
31
37
41
43
47
```

Java Program to find sum and average of N numbers

```java
public class SumAvg {
    public static void main(String[] args) {
        int arr[] = {5,6,48,6,78,20,65};
        int len = arr.length;
        double avg =0;

        double sum = 0;

        for (int i=0;i<len;i++){
            sum += arr[i];
        }
        avg = sum/len;
        System.out.println ("Sum of "+len+" number in "+sum);
        System.out.println ("Average is "+avg);
    }
}
```

Output:

```
Sum of 7 number in 228.0
Average is 32.57142857142857
```

Application: GUI Based Simple Calculator

Conclusion:

Thus we have written Java program for

i) To find factorial of number.
ii) To display first 50 prime numbers.
iii) To find sum and average of N numbers

After writing and simulating above program we have understood about Java environment, IDE used to write java programs and also some programming structure of java programming.

Conclusion:

Thus we have written Java program for

iv) To find factorial of number.
v) To display first 50 prime numbers.
vi) To find sum and average of N numbers

After writing and simulating above program we have understood about Java environment, IDE used to write java programs and also some programming structure of java programming.

Questions:

1. How factorial of a number is going to calculate
2. What is the logic behind to find prime number in Java Programming

Experiment No: 2		
Aim: Write a program in Java to implement a Calculator with simple arithmetic operations		

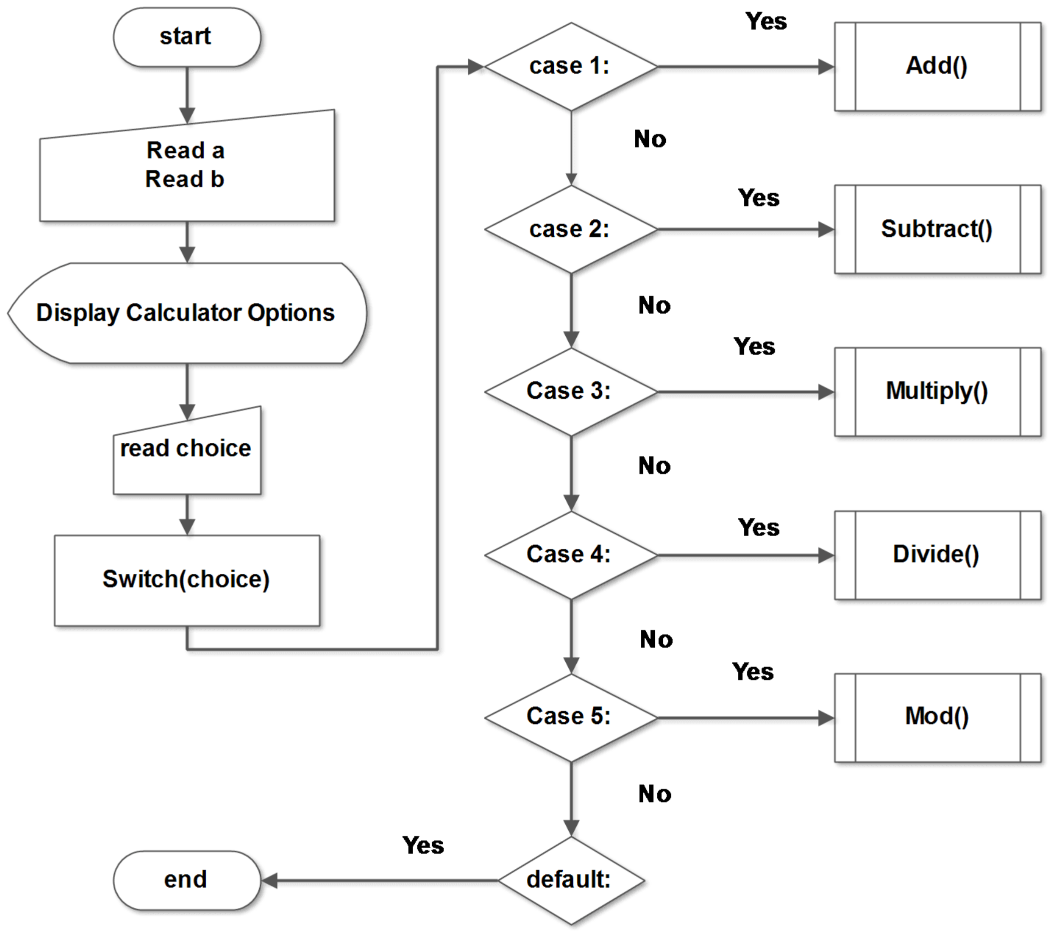
Objective: To learn Constants, Variables, and Data Types, Operators and Expressions, Decision making statements in Java.

Outcome: Student will able to understand and write java program that consists of Constants, Variables, and Data Types, Operators and Expressions, Decision making statements

Pre-requisite: Computer with JDK installed and any IDE for Java Programming

Theory:

Implementing calculator in Java needs to understand about constants, variables, and data types, operator, control statements and these concepts play a major role. If we know Java tokens and control statements it is very easy to implement the simple calculator using switch statements. Flowchart for the simple calculator using switch case as shown in below.



Java Program:

```java
import java.util. Scanner;
class SimpleCalculator {
  public static void main(String[] args) {

    char operator;
    Double number1, number2, result;

    // create an object of Scanner class
    Scanner input = new Scanner(System.in);

    // ask users to enter operator
    System.out.println ("Choose an operator: +, -, *, or /");
    operator = input.next().charAt(0);

    // ask users to enter numbers
    System.out.println ("Enter first number");
    number1 = input.nextDouble ();

    System.out.println ("Enter second number");
    number2 = input.nextDouble ();

    switch (operator) {

      // performs addition between numbers
      case '+':
        result = number1 + number2;
        System.out.println (number1 + " + " + number2 + " = " + result);
        break;

      // performs subtraction between numbers
      case '-':
        result = number1 - number2;
        System.out.println (number1 + " - " + number2 + " = " + result);
        break;

      // performs multiplication between numbers
      case '*':
        result = number1 * number2;
        System.out.println (number1 + " * " + number2 + " = " + result);
        break;

      // performs division between numbers
      case '/':
        result = number1 / number2;
        System.out.println (number1 + " / " + number2 + " = " + result);
        break;

      default:
        System.out.println ("Invalid operator!");
        break;
    }

    input.close();
  }
}
```

Output:

```
Choose an operator: +, -, *, or /
+
Enter first number
20
Enter second number
30
20.0 + 30.0 = 50.0
```

Application: GUI Based Simple calculator

Conclusion

Thus we have written java program for simple calculator using switch statement. For this experiment we got to know about how Constants, Variables, and Data Types, Operators and Expressions, Decision making statements are used in Java programming.

Questions:

1. How factorial of a number is going to calculate
2. What is the logic behind to find prime number in Java Programming

Experiment No: 3		
Aim: Write a program in Java with the class Rectangle with the data field’s width, length, area and colour the length, width and area are of double type and the colour is of string type. The methods are get_length (), get_width (), get_colour () and find_area (). Create two objects of Rectangle and compare their area and colour. If the area and colour both are the same for the objects then display “Matching Rectangles”, otherwise display “Non-matching Rectangle”		

Objective: To learn Class, Objects, Methods, and Data Fields in Java Object Oriented Programming

Outcome: Student will able to understand and write java program that consists of Class, Objects, Methods, and Data Fields

Pre-requisite: Computer with JDK installed and any IDE for Java Programming

Theory:

Java Class:

A class is a blueprint for the object. Before we create an object, we first need to define the class.

We can think of the class as a sketch (prototype) of a house. It contains all the details about the floors, doors, windows, etc. Based on these descriptions we build the house. House is the object.

Since many houses can be made from the same description, we can create many objects from a class. Below are the some properties of class

- Class is a set of object which shares common characteristics/ behavior and common properties/ attributes.
- Class is not a real world entity. It is just a template or blueprint or prototype from which objects are created.
- Class does not occupy memory.
- Class is a group of variables of different data types and group of methods

A class in java can contain:

1. data member
2. method
3. constructor
4. nested class and interface

In general, class declarations can include these components, in order:

Modifiers: A class can be public or has default access (Refer this for details).

Class keyword: class keyword is used to create a class.

Class name: The name should begin with an initial letter (capitalized by convention).

Superclass (if any): The name of the class's parent (superclass), if any, preceded by the keyword extends. A class can only extend (subclass) one parent.

Interfaces (if any): A comma-separated list of interfaces implemented by the class, if any, preceded by the keyword implements. A class can implement more than one interface.

Body: The class body is surrounded by braces, { }.

Syntax of the class with respect to aim of the experiment is

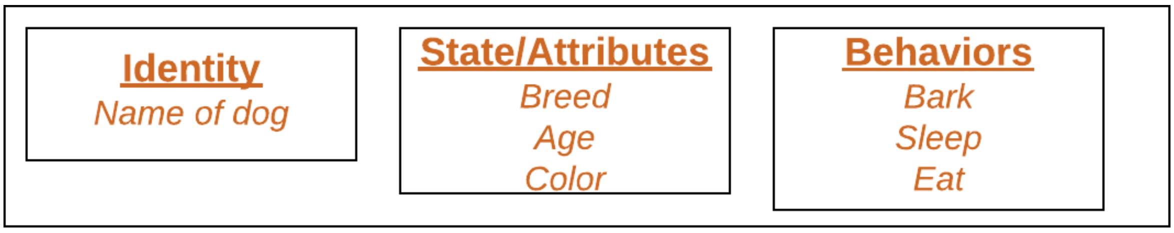
```
Class Rectangle

{

double width, length, area;

String color;

}
```

Java Objects:

It is a basic unit of Object-Oriented Programming and represents real life entities. A typical Java program creates many objects, which as you know, interact by invoking methods. An object consists of:

- **State:** It is represented by attributes of an object. It also reflects the properties of an object.
- **Behavior:** It is represented by methods of an object. It also reflects the response of an object with other objects.
- **Identity:** It gives a unique name to an object and enables one object to interact with other objects.

Example of an object: dog



When an object of a class is created, the class is said to be instantiated. All the instances share the attributes and the behavior of the class. But the values of those attributes, i.e. the state are unique for each object. A single class may have any number of instances.

Java Program:

```
import java.util.*;
class Rectangle
{
    double length;
    double width;
    String color;
    double area;

    void get_length(double x)
    {
        length=x;
    }

    void get_width(double y)
    {
        width=y;
    }

    void get_color(String s)
    {
         String color=s;
    }

    double rect_area()
    {
        area= length*width;
        return area;
    }
}

class Rect
{
    public static void main(String[] args) {
        Scanner sc= new Scanner(System.in);
        Rectangle r1= new Rectangle();
        Rectangle r2= new Rectangle();

        System.out.print("Enter the length of the rectangle1: ");
        double x1=sc.nextDouble();

        System.out.print("Enter the length of the rectangle2: ");
        double x2=sc.nextDouble();
        System.out.print("Enter the width of the rectangle1: ");
        double y1=sc.nextDouble();

        System.out.print("Enter the width of the rectangle2: ");
        double y2=sc.nextDouble();

        System.out.println("Enter the color of the rectangle1: ");
        String s1=sc.next();

        System.out.println("Enter the color of the rectangle2: ");
        String s2=sc.next();

        r1.get_length(x1);
        r2.get_length(x2);
        r1.get_width(y1);
        r2.get_width(y2);
        r1.get_color(s1);
        r2.get_color(s2);

        double rect1=r1.rect_area();
        System.out.println("Area of Rectangle1: "+rect1);
        double rect2=r2.rect_area();
        System.out.println("Area of Rectangle2: "+rect2);

        if(rect1==rect2 && s2.equals(s1))
        {
           System.out.println ("Matching Rectangles");
        }
        else
        {
           System.out.println ("Non-Matching Rectangles");
        }

    }
}
```

Output:

```
Enter the length of the rectangle1: 10

Enter the length of the rectangle2: 10

Enter the width of the rectangle1: 20

Enter the width of the rectangle2: 20
Enter the color of the rectangle1:
red
Enter the color of the rectangle2:
red
Area of Rectangle1: 200.0
Area of Rectangle2: 200.0
Matching Rectangles
```

Application: Java based geometry applications

Conclusion:

Thus we have written java program for rectangle class, from this experiment we got to know the OOPS concepts of java programming

Questions:

1. Define class and Object
2. What is mean by constructor
3. Which key word is used for creating objects for a particular class
4. Whet is data field and method in Java Programming

Experiment No: 4		
Aim: Write Programs in Java to sort i) List of integer's ii) List of names. The objective of this assignment is to learn Arrays and Strings in Java		

Objective: To learn how to implement non primitive data types in Java Programming

Outcome: Student will able to understand and write java program to sort a list of integers and list of names present in the array.

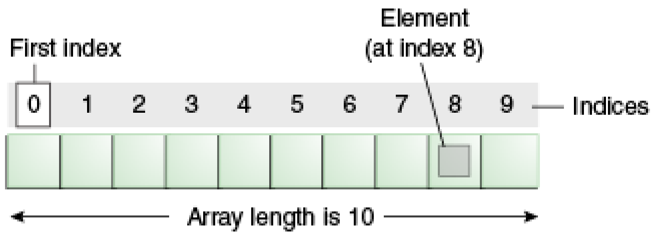
Pre-requisite: Computer with JDK installed and any IDE for Java Programming

Theory:

Java array is an object which contains elements of a similar data type. Additionally, the elements of an array are stored in a contiguous memory location. It is a data structure where we store similar elements. We can store only a fixed set of elements in a Java array.

In Java, array is an object of a dynamically generated class. Java array inherits the Object class, and implements the Serializable as well as Cloneable interfaces. We can store primitive values or objects in an array in Java. Like C/C++, we can also create single dimensional or multidimensional arrays in Java.

Moreover, Java provides the feature of anonymous arrays which is not available in C/C++.



The general form of a one-dimensional array declaration is

```
type var-name[];

OR

type[ ] var-name;
```

To create an array of integers, you could write:

```
int[ ] myNum = {10, 20, 30, 40};
```

In Java, a string is a sequence of characters. For example, "hello" is a string containing a sequence of characters 'h', 'e', 'l', 'l', and 'o'.
We use double quotes to represent a string in Java. For example,

//Create a String

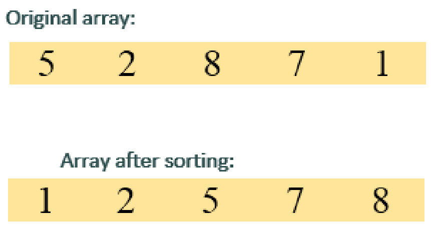
String type = “Java Programming”

Here, we have created a string variable named type. The variable is initialized with the string

Java Programming. Java String class provides a lot of methods to perform operations on strings such as compare(), concat(), equals(), split(), length(), replace(), compareTo(), intern(), substring() etc.

Java Program to sort List of Integers

In this program, we need to sort the given array in ascending order such that elements will be arranged from smallest to largest. This can be achieved through two loops. The outer loop will select an element, and inner loop allows us to compare selected element with rest of the elements.



Elements will be sorted in such a way that the smallest element will appear on extreme left which in this case is 1. The largest element will appear on extreme right which in this case is 8

Program

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

//Initialize array
int [] arr = new int [] {5, 2, 8, 7, 1};int temp = 0;

//Displaying elements of original array
System.out.println("Elements of original array: ");for (int i = 0; i
< arr.length; i++) {
  System.out.print(arr[i] + " ");
 }

 //Sort the array in ascending orderfor (int i = 0; i
 < arr.length; i++) {
  for (int j = i+1; j < arr.length; j++) {if(arr[i] > arr[j])
    {
       temp  =  arr[i]; arr[i]  =  arr[j];
       arr[j] = temp;
       }
    }
 }

 System.out.println();
 //Displaying elements of array after sorting System.out.println("Elements of array
 sorted in ascending order: ");for (int i = 0; i < arr.length; i++) {
  System.out.print(arr[i] + " ");
         }
 }
}
```

Output:

```
Elements of original array:

5 2 8 7 1

Elements of array sorted in ascending order:1 2 5 7 8
```

Java Program to Sort List of Names:

For, sorting names in an Alphabetical order there are multiple ways to sort the array, like using inbuilt Arrays.sort()

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

class NameSort {
    public static void main(String[] args)
    {
        // storing input in variable
        int n = 4;
        // create string array called names
        String names[]
            = { "Rahul", "Ajay", "Gourav", "Riya" };
        // inbuilt sort function
        Arrays.sort(names);
        // print output array
        System.out.println(
            "The names in alphabetical order are: ");
        for (int i = 0; i < n; i++) {
            System.out.println(names[i]);
        }
    }
}
```

Output:

```
The names in alphabetical order are:

Ajay

Gourav

Rahul

Riya
```

Application: Java based sorting applications.

Conclusion

Thus we have written java program for Array and String, from this experiment we got to know the sorting of array and string in java.

Questions

1. What is the difference between linear sorting and Bubble sort?
2. What is the application of sorting in software development?

Experiment No: 5		
Aim: Write a Program in Java to add two matrices.		

Objective: The objective of this assignment is to two dimensional array in Java

Outcome: Student will able to understand matrix addition operations and how to write program using for loop in Java.

Pre-requisite: Computer with JDK installed and any IDE for Java Programming

Theory:

Addition of two matrices can be carried if and only if both the matrices are in the same order. In matrix addition, each term of one matrix is added to the other matrix’s term, at the same location, i.e. the term at first row first column of Matrix 1 will be added to the term at first row first column of Matrix 2 and so on. A pictorial example is given below.

Matrix Addition

$$\begin{bmatrix} a & b \\ c & d \end{bmatrix} + \begin{bmatrix} w & x \\ y & z \end{bmatrix} = \begin{bmatrix} a+w & b+x \\ c+y & d+z \end{bmatrix}$$

<u>Java Program</u>

```
public class AddMatrices {
    public static void main(String[] args) {
        int rows = 2, columns = 3;
        int[ ][ ] firstMatrix = { {2, 3, 4}, {5, 2, 3} };
        int[ ][ ] secondMatrix = { {-4, 5, 3}, {5, 6, 3} };

        // Adding Two matrices
        int[][] sum = new int[rows][columns];
        for(int i = 0; i < rows; i++) {
            for (int j = 0; j < columns; j++) {
                sum[i][j] = firstMatrix[i][j] + secondMatrix[i][j];
            }
        }
        // Displaying the result
        System.out.println("Sum of two matrices is: ");
        for(int[] row : sum) {
            for (int column : row) {
                System.out.print(column + "    ");
            }
            System.out.println();
        }
    }
}
```

Output:

```
Sum of two matrices is:

-2    8    7

10    8    6
```

Application: Java based image processing applications

Conclusion

Thus we have written java program for adding two matrices in java programming, from this experiment we understood how to add two matrices using for loop.

Questions:

1. How two dimensional matrix is defined in Java Programming
2. What are the application of matrix in Java Programming

Experiment No: 6		
Aim: Write a program in Java to create a player class. Inherit the classes Cricket_player,Football_player and Hockey_player from player class.		

Objective: The objective of this assignment is to learn the concepts of inheritance in Java.

Outcome: Student will able to understand inheritance concept by writing Java programming.

Pre-requisite: Computer with JDK installed and any IDE for Java Programming.

Theory:

Inheritance in Java is a mechanism in which one object acquires all the properties and behaviours of a parent object. The idea behind inheritance in Java is that you can create new classes that are built upon existing classes. When you inherit from an existing class, you can reuse methods and fields of the parent class. Moreover, you can add new methods and fields in your current class also. The inheritance provides a great advantage called code reusability.

Terms used in Inheritance

- **Class:** A class is a group of objects which have common properties. It is a template or blueprint from which objects are created.
- **Sub Class/Child Class:** Subclass is a class which inherits the other class. It is also called a derived class, extended class, or child class.
- **Super Class/Parent Class:** Superclass is the class from where a subclass inherits the features. It is also called a base class or a parent class.
- **Reusability:** As the name specifies, reusability is a mechanism which facilitates you to reuse the fields and methods of the existing class when you create a new class. You can use the same fields and methods already defined in the previous class.

The syntax of Java Inheritance

```
class Subclass-name extends Superclass-name
      {
            //methods and fields
       }
```

The extends keyword indicates that you are making a new class that derives from an existing class. The meaning of "extends" is to increase the functionality.

Java Program

```
class Player
{
    public void Display()
    {
        System.out.println ("This is Player class");
    }
}

class Cricket_Player extends Player
{
    public void Display()
    {
        System.out.println ("My favorite Cricket Player is Sachin Tendulkar");
    }
}

class Football_Player extends Player
{
    public void Display ()
    {
        System.out.println ("My favorite Football Player is Ronaldo,
Cristiano");
    }
}

class Hockey_Player extends Player
{
    public void Display()
    {
        System.out.println("My favorite Hockey Player is Dhyan Chand");
    }
}
class Main
{
    public static void main(String[] args) {

        Cricket_Player d1= new Cricket_Player();
        d1.Display();
        Football_Player d2= new Football_Player();
        d2.Display();
        Hockey_Player d3= new Hockey_Player();
        d3.Display();
        }
 }
```

Output:

```
My favorite Cricket Player is Sachin Tendulkar

My favorite Football Player is Ronaldo, Cristiano

My favorite Hockey Player is Dhyan Chand
```

Application: To create web based applications

Conclusion:

Thus we have written java program for inheritance in java programming, from this experiment we understood Parent class, Child class and extending child class from parent class in inheritance.

Application:

1. What is mean by super class, sub class in inheritance
2. What are the different types of inheritance in Java Programming
3. Why multiple inheritance is not going to implement in Java Programming

Experiment No: 7		
Aim: Write a Java program which imports user defined package and uses members of the classes contained in the package.		

Objective: The objective of this assignment is to learn the concepts of user defined packages in Java.

Outcome: Student will able to understand how to create user defined package and utilizing this package in another java program.

Pre-requisite: Computer with JDK installed and any IDE for Java Programming.

Theory:

Java package is a mechanism of grouping similar type of classes, interfaces, and sub-classes collectively based on functionality. When software is written in the Java programming language, it can be composed of hundreds or even thousands of individual classes. It makes sense to keep things organized by placing related classes and interfaces into packages. The classes contained in the packages of other programs can be easily reused.

Using packages while coding offers a lot of advantages like:

- **Re-usability:** The classes contained in the packages of another program can be easily reused
- **Name Conflicts:** Packages help us to uniquely identify a class, for example, we can have company.sales.Employee and company.marketing.Employee classes
- **Controlled Access:** Offers access protection such as protected classes, default classes and private class
- **Data Encapsulation:** They provide a way to hide classes, preventing other programs from accessing classes that are meant for internal use only
- **Maintenance:** With packages, you can organize your project better and easily locate related classes

<u>Creating Packages or user defined packages</u>

User-defined packages are those which are developed by users in order to group related classes, interfaces and sub packages.

- Creating a package in Java is a very easy task. Choose a name for the package and include a package command as the first statement in the Java source file.
- The java source file can contain the classes, interfaces, enumerations, and annotation types that you want to include in the package.
- For example, the following statement creates a package named MyPackage.

package MyPackage;

Java Program

Including a Class in Java Package

To create a class inside a package, you should declare the package name as the first statement of your program. Then include the class as part of the package. But, remember that, a class can have only one package declaration. Here’s a simple program to understand the concept.

```
package MyPackage;
public class Compare {
int num1, num2;
Compare(int n, int m) {
     num1 = n;
     num2 = m;
}

public void getmax () {
    if (num1 > num2 ) {
        System.out.println("Maximum value of two numbers is " + num1);

  }

    else {

        System.out.println("Maximum value of two numbers is " + num2);

    }

}

public static void main(String args[]) {
        Compare current[] = new Compare[3];
        current[1] = new Compare(5, 10);
        current[2] = new Compare(123, 120);
        for(int i=1; i < 3 ; i++)
              {

              current[i].getmax();

              }

       }

}
```

Output:

```
Maximum value of two numbers is 10

Maximum value of two numbers is 123
```

Creating a class inside package while importing another package

Here's a sample program demonstrating the concept.

```
package Edureka;

import MyPackage.Compare;



public class Demo {

    public static void main (String args[]) {
        int n=10, m=10;
        Compare current = new Compare (n, m);
        if (n != m) {
             current.getmax();

        }

        else {

            System.out.println ("Both the values are same");

        }

}

}
```

Output:

```
Both the values are same
```

Application: To create packages for different types of applications

Conclusion: Thus we have written java program for user defined packages in java programming, from thisexperiment we have studied and practically understood how to create user defined packages and utilizing these packages in another java programming.

Questions:

1. Why we need packages in Java Programming
2. What is the difference between system packages and user defined packages

Experiment No: 8		
Aim: Write a java program which use try and catch for exception handling.		

Objective: The objective of this assignment is to learn the concepts of Exception handling in Java.

Outcome: Student will able to understand about compile time error, run time error and how to handle the exceptions

Pre-requisite: Computer with JDK installed and any IDE for Java Programming.

Theory:

It is common to make mistake while developing as well as typing a program. A mistake might lead to an error causing to program to produce unexpected results. Errors are the wrongs that can make a program go wrong

Types of Errors

Error may broadly be classified into two categories

- Compile-time errors
- Run-time errors

Compile time error:

All syntax error will be detected and displayed by the Java compiler and therefore these errors are known as compile-time error. Whenever the compiler display an error, it will not create the .class file.

Most of the compile-time errors are due to typing mistakes. The most common problems are:

- Missing semicolons
- Missing brackets
- Missing double quotes
- Use of undeclared variables
- Incompatible types in assignments
- Bad reference to objects

Run-time error:

Sometimes, a program may compile successfully creating the .class file but may not run properly. Such programs may produce wrong results due to wrong logic or may terminate due to errors such as stack overflow. Most common run-time errors are.

- Dividing an integer by zero

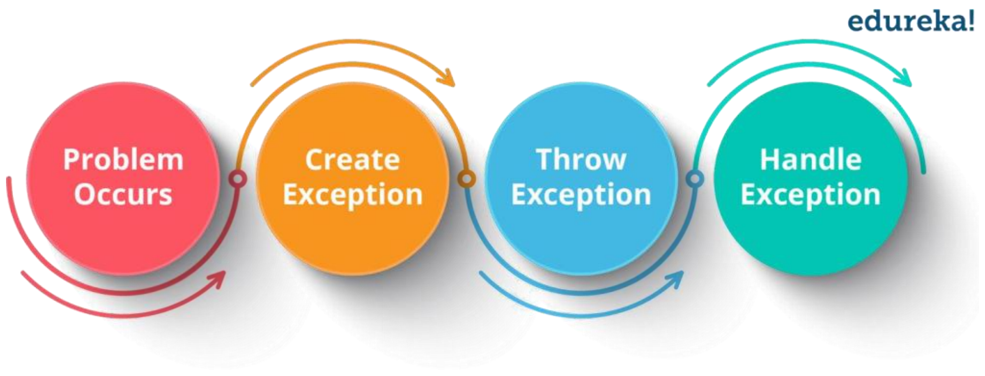
- Accessing element that is out of bounds of an array
- Trying to store a value into an array of incompatible class or type.
- Attempting to use negative size of an array
- Converting invalid string to number

Exceptions

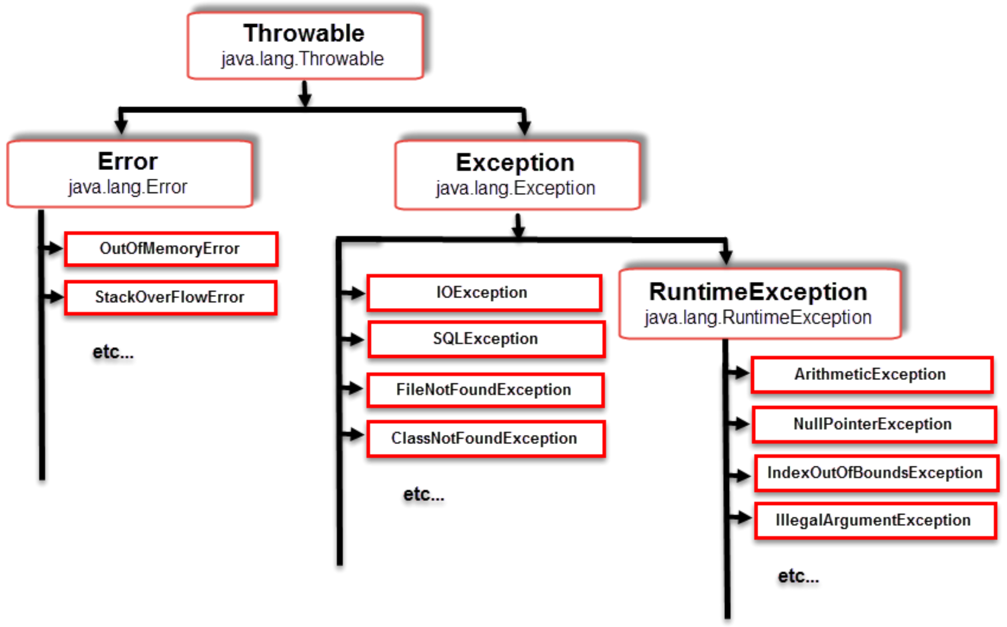
An exception is an unexpected event that occurs during program execution. It affects the flow of the program instructions which can cause the program to terminate abnormally.

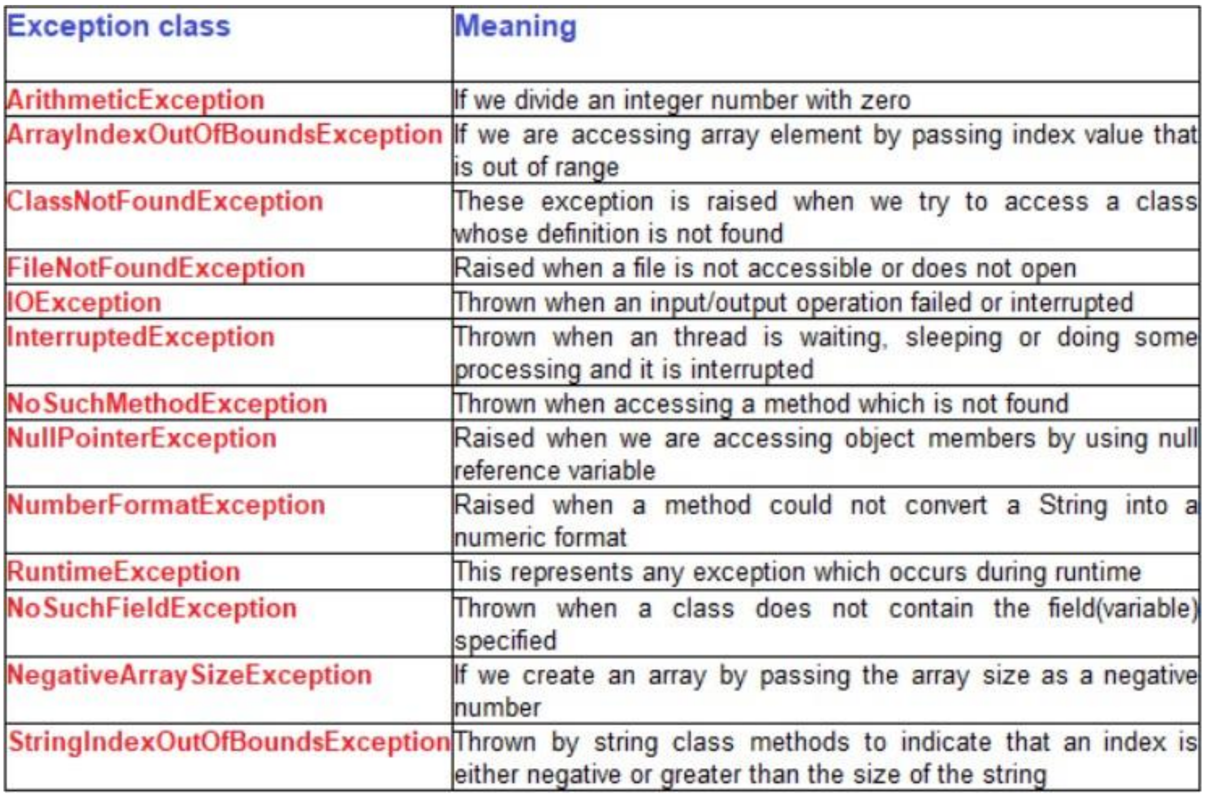
What is Exception Handling?

Exception Handling is a mechanism to handle runtime errors. The following steps are included while handing exceptions



Hierarchy of Java Exception classes





Exception class	Meaning
ArithmeticException	If we divide an integer number with zero
ArrayIndexOutOfBoundsException	If we are accessing array element by passing index value that is out of range
ClassNotFoundException	These exception is raised when we try to access a class whose definition is not found
FileNotFoundException	Raised when a file is not accessible or does not open
IOException	Thrown when an input/output operation failed or interrupted
InterruptedException	Thrown when an thread is waiting, sleeping or doing some processing and it is interrupted
NoSuchMethodException	Thrown when accessing a method which is not found
NullPointerException	Raised when we are accessing object members by using null reference variable
NumberFormatException	Raised when a method could not convert a String into a numeric format
RuntimeException	This represents any exception which occurs during runtime
NoSuchFieldException	Thrown when a class does not contain the field(variable) specified
NegativeArraySizeException	If we create an array by passing the array size as a negative number
StringIndexOutOfBoundsException	Thrown by string class methods to indicate that an index is either negative or greater than the size of the string

Java Programming

```
public class SampleMultipleCatchBlock{
 public static void main(String args[]){
    try{
       int a[]=new int[5];
       a[5]=30/0;
      }
      catch(ArithmeticException e)
        {System.out.println("task1 is completed");}
      catch(ArrayIndexOutOfBoundsException e)
        {System.out.println("task 2 completed");}
      catch(Exception e)
        {System.out.println("task 3 completed");}
      System.out.println("remaining code");
  }
}
```

Output:

```
task1 is completed

remaining code
```

Applications: Network based communication systems

Conclusion

Thus we have written java program for Exception handling in java programming, from this experiment we have studied try and catch block to handle the multiple exception.

Questions:

1. What is the purpose of a try-catch block in programming?
2. How a try-catch block work, and what does is the basic syntax for it in a programming language of your choice?
3. What is an exception in programming, and why are try-catch blocks used to handle them?

Experiment No: 9		
Aim : Write a program to demonstrate status of key on an Applet window such as KeyPressed, KeyReleased, KeyUp, KeyDown.		

AIM: Write a program to demonstrate status of key on an Applet window such Keypressed, KeyReleased, KeyUp, KeyDown.

OBJECTIVES: To study the mechanism of handling keyboard events.

APPRATUS: PC with Eclipse & JDK (version lesser than 8).

THEORY:

An event which indicates that a keystroke occurred in a component. This low-level event is generated by a component object (such as a text field) when a key is pressed, released, or typed. The event is passed to every **KeyListener** or **KeyAdapter** object which registered to receive such events using the component's **addKeyListener** method. (**KeyAdapter** objects implement the **KeyListener** interface.) Each such listener object gets this **KeyEvent** when the event occurs.

Pressing and releasing key on keyboard results in generating following key events:

1. **KEY_PRESSED**
2. **KEY_TYPED** (is only generated if a valid Unicode character could be generated.)
3. **KEY_RELEASED**

The "**Key typed**" events are higher-level and generally do not depend on platform or keyboard layout. They are generated when a Unicode character is entered, and are the preferred way to find out about character input. In the simplest case, a key typed event is produced by a single key press (e.g., 'a'). Often, however, characters are produced by series of key presses (e.g., 'shift' + 'a'), and the mapping from key pressed events to key typed events may be many-to-one or many-to-many. Key releases are not usually necessary to generate a key typed event, but there are some cases where the key typed event is not generated until a key is released (e.g., entering ASCII sequences via the Alt-Numpad method in Windows). No key typed events are generated for keys that don't generate Unicode characters (e.g., action keys, modifier keys, etc.).

The **getKeyChar()** method always returns a valid Unicode character or **CHAR_UNDEFINED**. Character input is reported by **KEY_TYPED** events: **KEY_PRESSED** and **KEY_RELEASED** events are not necessarily associated with character input. Therefore, the result of the **getKeyChar()** method is guaranteed to be meaningful only for **KEY_TYPED** events. For key pressed and key released events, the **getKeyCode** method returns the event's **keyCode**. For key typed events, the **getKeyCode()** method always returns **VK_UNDEFINED**. The **getExtendedKeyCode()** method may also be used with many international keyboard layouts.

"**Key pressed**" and "**key released**" events are lower-level and depend on the platform and keyboard layout. They are generated whenever a key is pressed or released, and are the only way to find out about keys that don't generate character input (e.g., action keys, modifier keys, etc.).

The key being pressed or released is indicated by the **getKeyCode()** and **getExtendedKeyCode()** methods, which return a virtual key code. Virtual key codes are used to report which keyboard key has been pressed, rather than a character generated by the combination of one or more keystrokes (such as "A", which comes from shift and "a"). For example, pressing the Shift key will cause a **KEY_PRESSED** event with a **VK_SHIFT keyCode**, while pressing the 'a' key will result in a **VK_A** keyCode. After the 'a' key is released, a **KEY_RELEASED** event will be fired with **VK_A**. Separately, a **KEY_TYPED** event with a **keyChar** value of 'A' is generated.

Some constants defined by key event:

VK_0, VK_9, VK_Z, VK_A, VK_ENTER, VK_UP, VK_DOWN, VK_LEFT, VK_RIGHT etc.

Methods in KeyEvent Class:

- `int getExtendedKeyCode()` - Returns an extended key code for the event.
- `static int getExtendedKeyCodeForChar(int c)` - Returns an extended key code for a unicode character.
- `char getKeyChar()` - Returns the character associated with the key in this event.
- `int getKeyCode()` - Returns the integer keyCode associated with the key in this event.
- `int getKeyLocation()` - Returns the location of the key that originated this key event.
- `static String getKeyModifiersText(int modifiers)` - Returns a String describing the modifier key(s), such as "Shift", or "Ctrl+Shift".
- `static String getKeyText(int keyCode)` - Returns a String describing the keyCode, such as "HOME", "F1" or "A".
- `boolean isActionKey()` - Returns whether the key in this event is an "action" key.
- `String paramString()` - Returns a parameter string identifying this event.
- `void setKeyChar(char keyChar)` - Set keyChar value to indicate logical character.
- `void setKeyCode(int keyCode)` - Set the keyCode value to indicate a physical key.
- `void setModifiers(int modifiers)` - Deprecated. as of JDK1.1.4

KeyListener Interface

This is listener interface for receiving keyboard events (keystrokes). The class that is interested in processing a keyboard event either implements this interface (and all the methods it contains) or extends the abstract KeyAdapter class (overriding only the methods of interest).

The listener object created from that class is then registered with a component using the component's addKeyListener method. A keyboard event is generated when a key is pressed, released, or typed. The relevant method in the listener object is then invoked, and the KeyEvent is passed to it.

Methods available are:

- **`void keyPressed(KeyEvent e)`** - Invoked when a key has been pressed.
- **`void keyReleased(KeyEvent e)`** - Invoked when a key has been released.
- **`void keyTyped(KeyEvent e)`** - Invoked when a key has been typed.

Steps in Handling Keyboard Event

To handle keyboard events, you use the same general architecture as that of mouse events. The difference is that we will be implementing the **`KeyListener`** interface. When a key is pressed, a **`KEY_PRESSED`** event is generated. This results in a call to the **`keyPressed()`** event handler. When the key is released, a **`KEY_RELEASED`** event is generated and the **`keyReleased()`** handler is executed. If a character is generated by the keystroke, then a **`KEY_TYPED`** event is sent and the **`keyTyped()`** handler is invoked. Each time the user presses a key, at least two and often three events are generated.

If all you care about are actual characters, then you can ignore the information passed by the key press and release events. However, if your program needs to handle special keys, such as the *arrow* or *function* keys, then it must watch for them through the **`keyPressed()`** handler.

ALGORITHM:

1. Import the packages required for KeyEvent handling mechanism.
2. Create a class that extends applet and implements KeyListener interface.
3. Define init() method of applet
4. Register a key Listener in init() method
5. Override keyPressed(), keyReleased(), and keyTyped() methods to for handling of events with the keys.
6. Define paint method to display the event or action of key.

CONCLUSIONS:

1. Explain event handling mechanism?
2. Explain mechanism for handling keyboard event?
3. Explain methods of KeyListener Interface?

Experiment No: 10		
Write a program to create a frame using AWT. Implement mouseClicked(), mouseEntered() and mouseExited() events. Frame should become visible when the mouse enters it.		

AIM: Write a program to create a frame using AWT. Implement mouseClicked(), mouseEntered() and mouseExited() events. Frame should become visible when the mouse enters it.

OBJECTIVES: To study concept of frames and mechanism of handling mouse events**.**

APPRATUS: PC with Eclipse and JDK

THEORY:

MouseEvent is a subclass of InputEvent. This event indicates a mouse action occurred in a component. The mouse events are generated due to following:

- **Mouse Events**
  - a mouse button is pressed
  - a mouse button is released
  - a mouse button is clicked (pressed and released)
  - the mouse cursor enters the unobscured part of component's geometry
  - the mouse cursor exits the unobscured part of component's geometry
- **Mouse Motion Events**
  - the mouse is moved
  - the mouse is dragged

A MouseEvent object is passed to every MouseListener or MouseAdapter object which is registered to receive the “interesting” mouse events using the component's addMouseListener method. (MouseAdapter objects implement the MouseListener interface.)

A MouseEvent object is also passed to every MouseMotionListener or MouseMotionAdapter object which is registered to receive mouse motion events using the component's addMouseMotionListener method. (MouseMotionAdapter objects implement the MouseMotionListener interface.)

To handle mouse events, we must implement MouseListener, MouseMotionListener and MouseWheelListener interfaces. When a mouse event occurs, events are generated and sent to the registered MouseListeners.

The fields for **`java.awt.event.MouseEvent`** class:

- **`static int BUTTON1`** - Indicates mouse button #1; used by **`getButton()`**
- **`static int BUTTON2`** - Indicates mouse button #2; used by **`getButton()`**
- **`static int BUTTON3`** - Indicates mouse button #3; used by **`getButton()`**
- **`static int MOUSE_CLICKED`** - The "mouse clicked" event
- **`static int MOUSE_DRAGGED`** - The "mouse dragged" event
- **`static int MOUSE_ENTERED`** - The "mouse entered" event
- **`static int MOUSE_EXITED`** - The "mouse exited" event

- `static int MOUSE_FIRST` - The first number in range of ids used for mouse events
- `static int MOUSE_LAST` - The last number in range of ids used for mouse events
- `static int MOUSE_MOVED` - The "mouse moved" event
- `static int MOUSE_PRESSED` - The "mouse pressed" event
- `static int MOUSE_RELEASED` - The "mouse released" event
- `static int MOUSE_WHEEL` - The "mouse wheel" event
- `static int NOBUTTON` - Indicates no mouse buttons; used by **`getButton()`**
- `static int VK_WINDOWS` - Constant for the Microsoft Windows "Windows" key.

Mouse Listener Interface

The listener interface for receiving “interesting” mouse events (press, release, click, enter, and exit) on a component. The class that is interested in processing a mouse event either implements this interface (and all the methods it contains) or extends the abstract MouseAdapter class (overriding only the methods of interest).
The methods in this interface are :

- `void mouseClicked(MouseEvent e)` - Invoked when mouse button has been clicked (pressed and released) on a component.
- `void mouseEntered(MouseEvent e)` - Invoked when mouse enters a component.
- `void mouseExited(MouseEvent e)` - Invoked when mouse exits a component.
- `void mousePressed(MouseEvent e)` - Invoked when a mouse button has been pressed on a component.
- `void mouseReleased(MouseEvent e)` - Invoked when a mouse button has been released on a component.

MouseMotionListener Interface

The listener interface for receiving mouse motion events on a component. The class that is interested in processing a mouse motion event either implements this interface (and all the methods it contains) or extends the abstract MouseMotionAdapter class (overriding only the methods of interest). The listener object created from that class is then registered with a component using the component's addMouseMotionListener method. A mouse motion event is generated when the mouse is moved or dragged.
The methods available are :

- `void mouseDragged(MouseEvent e)` - Invoked when a mouse button is pressed on a component and then dragged.
- `void mouseMoved(MouseEvent e)` - Invoked when the mouse cursor has been moved onto a component but no buttons have been pushed.

MouseWheelListener Interface

The listener interface for receiving mouse wheel events on a component. The class that is interested in processing a mouse wheel event implements this interface (and all the methods it contains). The listener object created from that class is then registered with a component using the component's addMouseWheelListener method. A mouse wheel event is generated when the

mouse wheel is rotated. When a mouse wheel event occurs, that object's mouseWheelMovedmethod is invoked.
It has single method:

- **`void mouseWheelMoved(MouseWheelEvent mwe) -`** Invoked when the mousewheel is rotated.

ALGORITHM:

1. Import the packages.
2. Generate a class that extends Frame & implements required mouse listeners.
3. Create a Frame.
4. Set conditions for size, layout, and visibility of frame.
5. Register the mouse listener
6. Implementation the listener methods to handle mouse events.
7. Display appropriate messages for the mouse events.
8. Add logic to close fame.

CONCLUSIONS:

1. Explain frame window in java.
2. State types of mouse events & explain mechanisms to handle them?