D Y PATIL
DEEMED TO BE
UNIVERSITY
RAMRAO ADIK
INSTITUTE OF TECHNOLOGY
NAVI MUMBAI

LAB MANUAL

(FOR STUDENT)

SEM III
B. TECH.

Object Oriented
Programming Lab

D Y PATIL
DEEMED TO BE
UNIVERSITY
RAMRAO ADIK
INSTITUTE OF TECHNOLOGY
NAVI MUMBAI

DEPARTMENT OF
COMPUTER ENGINEERING

Vision of the Institute

To foster and permeate higher and quality education with value added engineering and technology programs by providing all facilities in terms of technology and platforms for all round development with social awareness for youths.

Mission of the Institute

To become a pivotal center of service to Industry, academy and society with the latest technology by providing facilities for **advanced research and development** programs on par with **international standards**.

To produce engineering and technology professionals who are **innovative and inspiring thought leaders**, adept at **solving problems** faced by our nation and world by providing **quality education**.

Goal of the Institute

We at RAIT assure our main stakeholders 100% quality of students for the programmes we deliver. This quality assurance stems from the teaching and learning processes we have at work at our campus. The teachers are handpicked from reputed institutions IIT/NIT/MU, etc. and they inspire the students to be innovative in thinking and practical in approach. We have installed internal procedures to better skill sets of instructors by sending them to training courses, workshops, seminars and conferences. We also have a full-fledged course curriculum and deliveries planned in advance for a structured semester long programme. These tools help us to ensure same quality of teaching independent of any individual instructor. Each classroom is equipped with internet and other digital learning resources.

The effective learning process in the campus comprises a clean and stimulating classroom environment and availability of lecture notes and digital resources prepared by instructor from the comfort of home. In addition, a student is provided with good number of assignments that would trigger his thinking process. The testing process involves an objective test paper that would gauge the understanding of concepts of the students. The Q&A process also ensures that the learning process is effective. The summer internships and project work-based training ensure learning process to include practical and industry relevant aspects. Various technical events, seminars and conferences make the student learning complete.

Vision of the Department

To impart higher and quality education in computer science with value added engineering and technology programs to prepare technically sound, ethically strong engineers with social awareness. To extend the facilities, to meet the fast changing requirements and nurture the youths with international competencies and exemplary level of employability and research under highly competitive environments.

Mission of the Department

- To impart quality and value-based education to be at far with the state of art technology. To mobilize the resources and equip the institution with men and materials of excellence to provide knowledge and develop technologies in the thrust areas of computer science and Engineering.
- To provide the diverse platforms of sports, technical, co-curricular and extracurricular activities for the overall development of student with ethical attitude.

- To prepare the students to sustain the impact of computer education for social needs encompassing industry, educational institutions and public service.
- To collaborate with IITs, reputed universities and industries for the technical and overall upliftment of students for continuing learning and entrepreneurship.

Index

Course Objective, Course Outcome & Experiment Plan

Course Objective:

	Course Objective
1	To understand the object-oriented programming basics and its features.
2	Able to use a programming language to resolve problems.
3	To understand and apply Object Oriented Programming (OOP) principles using Java.
4	To study various java programming concept like multithreading, exception handling, packages etc.
5	To explain components of GUI based programming.
6	To understand the object-oriented programming basics and its features.

Course Outcomes:

CO	CO Description
CO1	Understand basics of OOP and apply fundamental programming constructs
CO2	Understand and illustrate the features of classes and objects
CO3	Elaborate the concept of strings, arrays and vectors
CO4	Develop a program that implements the concept of inheritance, interfaces and packages
CO5	Implement the notion of exception handling and multithreading
CO6	Develop GUI based application

Exp. No.	Week No.	Experiment Name	CO	Weightage
1	W1	The grading system describes how well students have achieved the learning objectives or goals established for a class of course of study. This system helps to categorize the students according to their grades. Design a system that reads marks obtained by a student in a test of 100 marks and assign the grade.	**CO1**	**5**
2	W2	Prime numbers are important because the security of many encryption algorithms are based on the fact that it is very fast to multiply two large numbers and get the result, while it is extremely computer-intensive to do the reverse. Enlist all the prime numbers between 1 and 1000 to create a base for cryptography.	**CO1**	**5**
3	W3	The vendor provide a rubber material that provides the protection to the edges of rectangular object and paper material for protection of front and back of rectangular object. So a vendor needs an application to calculate the amount of rubber and paper material required for covering rectangular object. Write area method for calculating area of rectangle and square.(using class and object)	**CO2**	**4**
4	W4	School students need to study and learn formula for calculating area of different shapes like circle, rectangle, triangle and square. Design an application which will read require parameters for the area	**CO2**	**3**

		calculation of different shapes (use method overloading)		
5	W5	An electrical engineer needs a complex number calculator for performing the operation of addition of alternating current represented using complex number. Create an application that takes 2 objects complex number as parameters and return the object which is addition of 2 numbers passed as parameter of complex number.	**CO2**	**3**
6	W6	2D array is used in many real-life applications where we need to organize data in tabular/matrix format. Hence a matrix manipulator is required with functionality of reading, displaying and flipping data from the matrix. Generate the methods, for the functionality mentioned above for creation the matrix manipulator.	**CO3**	**5**
7	W7	A character sequence is to be read as an input and result need to declare as "yes" or "no" by investigating the fact that traversing the characters sequence backwards and forwards results in same sequence. Write a program for the same using StringBuffer.	**CO3**	**5**
8	W8	Write a program to calculate volume of sphere using multilevel inheritance. The base class method will accept the radius from user. A class will be derived from the above mentioned class that will have a method to find the area of a circle derived from this will have method to calculate and display the volume of the sphere.	**CO4**	**5**

9	W9	Consider a university where students who participate in the National Games or Olympics are given some grace marks. The grace marks provided are fixed and same for every student. Create an application that keeps student's academic marks and Sports grace marks separate and generate total of marks considering academics and sports both. Also invoke methods of base class & interface using reference. (Hint: Make use of Interface).	**CO4**	**5**
10	W10	Through Custom exception user can raise application-specific error code. You are required to calculate a square of even number provided as input by user. However if a user provides an odd number as input, then an exception must be thrown explicitly with message indicating the input number must be even number.	**CO5**	**5**
11	W11	Divide your program into two parts: One to read a number and the other to calculate its square. Provide a simultaneous execution of both parts of a program to maximum utilize the CPU time.	**CO5**	**5**
12	W12	Create a GUI application which is fully equipped with the functionality of solving various computation problems such as Addition, subtraction, Multiplication and Division. The GUI application should runs in a browser and displays the output as result of these mathematical operations.	**CO6**	**10**

Study & Evaluation Scheme

Course Code	Course Name	Teaching Scheme (Hrs)			Credits Assigned			
		Theory	Practical	Tutorial	Theory	Practical	Tutorial	Total
231CEU CL32	Object Oriented Programming Lab	--	02	--	--	01	--	01

Course Code	Course Name	Examination Scheme			Examination Scheme			
		IA	MSE	ESE	Term work	Practical	Tutorial	Total
231CEU CL32	Object Oriented Programming Lab	--	--	--	25	25	--	50

Term Work:

1. Term work assessment must be based on the overall performance of the student with every experiment graded from time to time. The grades should be converted into marks as per the Credit and Grading System manual and should be added and averaged.
2. The final certification and acceptance of term work ensures satisfactory performance of laboratory work and minimum passing marks in term work.

Practical & Oral:

Practical & Oral exam will be based on the entire syllabus of Object Oriented Programming

D Y PATIL
DEEMED TO BE
UNIVERSITY
RAMRAO ADIK
INSTITUTE OF TECHNOLOGY
NAVI MUMBAI

OOP Lab

EXPERIMENT NO. 1

Demonstration of Branching Statement

Experiment No. 1

1. Aim:

The grading system describes how well students have achieved the learning objectives or goals established for a class of course of study. This system helps to categorize the students according to their grades. Design a system that reads marks obtained by a student in a test of 100 marks and assign the grade.

Marks	Grade
0 to 39	Fail
40 to 49	Pass
50 to 59	Second Class
60 to 69	First Class
70 to 100	Distinction

2. Objectives:

- To understand the concept of Object Oriented Programming.
- To understand the switch case statement

3. Outcomes:

To apply fundamental programming constructs to understand object oriented programming concepts

4. Software Required:

Ubuntu, JDK 1.5.0 onwards

5. Theory:

The control statement that allows us to make a decision from the number of choices is called a switch .A switch statement allows you to test the value of an expression and, depending on that value, to jump directly to some location within the switch statement. Only expressions of certain types can be used. The value of the expression can be one of the primitive integer type's int, short, or byte. It can be the primitive char type.

The positions that you can jump to are marked with case labels that take the form: "case constant:" This marks the position the computer jumps to when the expression evaluates to the given constant. As the final case in a switch statement you can, optionally, use the label "default:" which provides a default jump point that is used when the value of the expression is not listed in any case label.

//Syntax of switch statement:

```
switch (expression )
{
case constant1 : statement ;
break ;
case constant2 : statement ;
break;
case constant3 : statement ;
break;
default : statement ;
}
```

Sample Program:

```
class sample
{
 public static void main( String args[])
{
int i = 1 ;
```

```
switch ( i )
{
case 1 : System.out.println( "I am in case 1") ; break;
case 2 : System.out.println ( "I am in case 2") ; break;
case 3 : System.out.println ( "I am in case 3") ; break;
default: System.out.println ( "I am in default case") ;
}
}
```

6. Algorithm:

Step 1: Initialize marks of student (between 0 to 100)

Step 2: Using switch case display grade of the student

Step 3: switch (marks/10)

Step 3.1: Print the class of students according to 10 different cases.

Step 3.2: Case 0, 1, 2, 3: Fail

Step 3.3: Case 4: Pass

Step 3.4: Case 5: Second Class

Step 3.5: Case 6: First Class

Step 3.6: Case 7, 8, 9, 10: Distinction

Step 4: Stop

7. Conclusion:

After performing this experiment we are able to use branching structures, control structure and switch case which helps in making decision by using switch cases.

8. Quiz / Viva Questions:

- Explain switch case statement?
- What is use of default label in switch case?

9. References:

1. Jaime Nino, Frederick A. Hosch, 'An introduction to Programming and Object Oriented Design using Java', Wiley Student Edition.
2. E Balgurusamy, "Programming with JAVA", Tata McGraw Hill

D Y PATIL
DEEMED TO BE
UNIVERSITY
RAMRAO ADIK
INSTITUTE OF TECHNOLOGY
NAVI MUMBAI

OOP Lab

EXPERIMENT NO.

2

Demonstration of Looping Statement

Experiment No. 2

1. Aim:

Prime numbers are important because the security of many encryption algorithms are based on the fact that it is very fast to multiply two large numbers and get the result, while it is extremely computer-intensive to do the reverse. Enlist all the prime numbers between 1 and 1000 to create a base for cryptography.

2. Objectives:

- Understand the basics of Control Statements
- Use the Selection (if-else) statements and Repetition (for loop) statement.

3. Outcomes:

To apply fundamental programming constructs to understand object oriented programming concepts

4. Software Required:

Ubuntu, JDK 1.5.0 onwards

5. Theory:

Control Statements: The control statement are used to control the flow of execution of the program. This execution order depends on the supplied data values and the conditional logic. Java contains the following types of control statements:

1- Selection Statements

2- Repetition Statements

3- Branching Statements

Selection statements:

1. If Statement: This is a control statement to execute a single statement or a block of code, when the given condition is true and if it is false then it skips if block and rest code of program is executed.

   Syntax:

   ```
   if(conditional expression){
     <Statements>;
    ...;
    ...;
   }
   ```

2. If-else Statement: The "if-else" statement is an extension of if statement that provides another option when 'if' statement evaluates to "false" i.e. else block is executed if "if" statement is false.

   Syntax:

   ```
   if(conditional expression){
      <Statements>;
      ...;
      ...;
      }
   else{
        <Statements>;
        ....;
        ....;
   }
   ```

6. Algorithm:

Step 1: Initialize i=1, num=1000, flag=1

Step 2: If i<=num

Step 2.1: Initialize j=2

Step 2.2: If j<i do

Step 2.2.1: If i%j==0

Step 2.2.1.1: Set flag=0

Step 2.2.1.2: break

Step 2.2.2: Increment j

Step 2.3: if flag=1

Step 2.3.1: Display i

Step 2.4: Increment i

Step 3: STOP

7. Conclusion:

After performing this experiment we are able to create a class and control statements.

8. Quiz / Viva Questions:

- What is a control statement?
- What are the types of the Control statements?

9. References:

1. Ralph Bravaco , Shai Simoson , "Java Programing From the Group Up" ,Tata McGraw-Hill
2. Java 2, "The Complete Reference of Java", Schildt publication

D Y PATIL
DEEMED TO BE
UNIVERSITY
RAMRAO ADIK
INSTITUTE OF TECHNOLOGY
NAVI MUMBAI

OOP Lab

EXPERIMENT NO.

3

Demonstration of Class, Objects, Methods

Experiment No. 3

1. Aim:

The vendor provide a rubber material that provides the protection to the edges of rectangular object and paper material for protection of front and back of rectangular object. So a vendor needs an application to calculate the amount of rubber and paper material required for covering rectangular object. Write area method for calculating area of rectangle and square (using class and object).

2. Objectives:

- To understand the object-oriented programming basics and its features.
- Able to use a programming language to resolve problems.
- To understand and apply Object Oriented Programming (OOP) principles using Java.

3. Outcomes:

To illustrate the concept of packages, classes and objects

4. Software Required:

Ubuntu, JDK 1.5.0 onwards

5. Theory:

Objects are the basic runtime entities in an object oriented system. A class may be thought of as a 'data type' and an object as 'variable' of that data type. The data and code of the class are accessed by object of that class. In this program, the object of class circle is used to access the methods: getdata() & area() of this class.

Syntax:

```
class Class_Name
    {
        public static void main(String args[])
          {
                    // Statements;
           }
    }
```

For example:

```
class Room
    {
        float length;
         float breadth;
         void getdata(float a, float b)
         {
             length=a;
             breadth=b;
         }
        public static void main(String args[])
        {
          float area;
          Room r1=new Room();
          R1.getdata(20,10);
          area=r1.length*r1.breadth;
          System.out.println("Area= "+area);
          }
```

```
}
```

Output: Area=200.0

6. Algorithm:

1. Start
2. Declare a class named Rectangle
3. Declare data members length and breadth
4. Define a method getDim() to take input for data members
5. Define a method area() to calculate area of Rectangle
6. Declare a method perimeter() to calculate the perimeter of Rectangle
7. Create object of Rectangle class and invoke methods of Rectangle class inside main() method
8. Stop

7. Conclusion:

After performing this experiment we are able to create a class and object. The classes are interacting with each other by accessing the methods and members of class using different objects.

8. Quiz / Viva Questions:

- What is class and object? How to access class using object?
- How two objects communicate?

9. References:

1. Ralph Bravaco , Shai Simoson , "Java Programing From the Group Up" ,Tata McGraw-Hill
2. Grady Booch, Object Oriented Analysis and Design
3. Jaime Nino, Frederick A. Hosch, 'An introduction to Programming and Object Oriented Design using Java', Wiley Student Edition. Wayne Tomasi, "Electronics Communication Systems", Pearson education, Fifth edition.
4. Java 2, "The Complete Reference of Java", Schildt publication

D Y PATIL
DEEMED TO BE
UNIVERSITY
RAMRAO ADIK
INSTITUTE OF TECHNOLOGY
NAVI MUMBAI

OOP Lab

EXPERIMENT NO. 4

Method Overloading

Experiment No. 4

1. Aim:

School students need to study and learn formula for calculating area of different shapes like circle, rectangle, triangle and square. Design an application which will read require parameters for the area calculation of different shapes (use method overloading).

2. Objectives:

- To understand the object-oriented programming basics and its features.
- Able to use a programming language to resolve problems.
- To understand and apply Object Oriented Programming (OOP) principles using Java.

3. Outcomes:

To illustrate the concept of method overloading

4. Software Required:

Ubuntu, JDK 1.5.0 onwards

5. Theory:

Java allows to create methods that have the same name, but different parameter lists and different definitions. This is called method overloading.

Method Overloading:

- Two or more methods within the same class that have the same name but methods must differ in the type and/or number of their parameters.
- Method overloading is one of the ways that Java implements polymorphism.

- When an overloaded method is invoked, Java uses the type and/or number of arguments to determine which version of the overloaded method to actually call.

Syntax:

```
class ClassName
{
    void Function1(data_type a, data_type b)
     {
        // Statements

     }
    void Function1(data_type a, data_type b, data_type c)
     {
        // Statements
     }
    public static void main(String args[])
     {
         ClassName obj=new ClassName();
          obj.Function1(Value1,Value2);
          obj.Function1(Value1,Value2,Value3);
       }
}
```

For example:

```
class Calculation
{
   void sum(int a,int b)
    {
    System.out.println("Addition of two numbers is:"+(a+b));
    }
    void sum(int a, int b, int c)
    {
       System.out.println("Addition of three numbers is:"+(a+b+c));
    }

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

```
        Calculation c=new Calculation();
        c.sum(10,40,60);
        c.sum(20,20);
        }
}
```

Output:

Addition of two numbers is: 110

Addition of three numbers is: 40

6. Algorithm:

1. Start
2. Declare a class named Shapes
3. Overload a method area(int) to calculate area of Square
4. Overload a method area(int, int) to calculate area of Rectangle
5. Overload a method area(int, int, int) to calculate area of Triangle
6. Create object of Shapes class and invoke methods with different parameters inside main() method to achieve method overloading.
7. Stop

7. Conclusion:

After performing this experiment we are able to create a class and object. The classes are interacting with each other by accessing the methods and members of class using different objects.

8. Quiz / Viva Questions:

- What is class and object? How to access class using object?
- How two objects communicate?

9. References:

1. Ralph Bravaco , Shai Simoson , "Java Programing From the Group Up" ,Tata McGraw-Hill
2. Grady Booch, Object Oriented Analysis and Design

3. Jaime Nino, Frederick A. Hosch, 'An introduction to Programming and Object Oriented Design using Java', Wiley Student Edition. Wayne Tomasi, "Electronics Communication Systems", Pearson education, Fifth edition.
4. Java 2, "The Complete Reference of Java", Schildt publication

D Y PATIL
DEEMED TO BE
UNIVERSITY
RAMRAO ADIK
INSTITUTE OF TECHNOLOGY
NAVI MUMBAI

OOP Lab

EXPERIMENT NO.

5

Constructor Overloading

Experiment No. 5

1. Aim:

An electrical engineer needs a complex number calculator for performing the operation of addition of alternating current represented using complex number. Create an application that takes 2 objects complex number as parameters and return the object which is addition of 2 numbers passed as parameter of complex number.

2. Objectives:

- Solve the real world scenarios using top down approach.
- Understand the concept and properties of constructor

3. Outcomes:

To illustrate the concept of constructor overloading

4. Software Required:

Ubuntu, JDK 1.5.0 onwards

5. Theory:

A constructor is a special public member method whose task is to initialize the object data members when an object is declared. The constructor of a class should have the same name as the class. For example if the name of class is Employee then the constructor is a method with the name Employee( ).

Characteristics of a constructor:

1. They should be declared in the public section.
2. They automatically get invoked when the objects are created.

3. They do not have return types not even void and therefore they cannot return values.

4. Like other methods they can have default arguments.

Types of Constructor:

1. Default Constructor: A constructor without arguments is called a default constructor. When no constructor is created explicitly then Java first implicitly creates a constructor without any parameter and invokes it. The default constructor then initializes the instance variables to default values, i.e. zero for numeric data types, null for string type and false for Boolean.

2. Parameterized Constructor: The constructor with arguments is called a parameterized constructor. In parameterized constructors the instance variable is initialized automatically at run time according to the values passed to parameters during the object creation.

Constructor Overloading:

Constructor overloading means we can create and use more than one constructor in a class. A constructor can be overloaded on the basis of following facts:

1. Number of parameters

2. Data type of parameters

3. Order of parameters

When we create object of the classes, the instance variables are initialized according to the constructor signature.

For example:

```
class Product
    {
        int x, y;
        Product( );      //Default Constructor
        Product(int a, int b )        //Parameterized Constructor
    }
```

6. Algorithm:

Step 1: Start

Step 2: Define class complex with

Step 2.1: data members real and imaginary

Step 2.2: default constructor to initialize real and imaginary to 0

Step 2.3: parameterized constructor to initialize real and imaginary

Step 2.4: void show() method to display complex number

Step 2.5: Complex sum(Complex, Complex) method to perform addition of two complex numbers

Step 3: Define main() method

Step 3.1: Create object A, B and C of Complex class

Step 3.2: Make a call to sum() and show() method

Step 4: Stop

7. Conclusion:

This experiment implements constructors and give us the clear scenario of characteristics and types of constructor along with parameters passing to constructor. We can formulate and solve real world problems.

8. Quiz / Viva Questions:

- What is constructor?
- What is garbage collection?
- Enlist different types of constructors

9. References:

1. Ralph Bravaco , Shai Simoson , "Java Programing From the Group Up" ,Tata McGraw-Hill
2. Grady Booch, Object Oriented Analysis and Design

3. Jaime Nino, Frederick A. Hosch, 'An introduction to Programming and Object Oriented Design using Java', Wiley Student Edition. Wayne Tomasi, "Electronics Communication Systems", Pearson education, Fifth edition.
4. Java 2, "The Complete Reference of Java", Schildt publication

D Y PATIL
DEEMED TO BE
UNIVERSITY
RAMRAO ADIK
INSTITUTE OF TECHNOLOGY
NAVI MUMBAI

OOP Lab

EXPERIMENT NO.

6

Demonstration of 2D- Array

Experiment No. 6

1. Aim:

2D array is used in many real-life applications where we need to organize data in tabular/matrix format. Hence a matrix manipulator is required with functionality of reading, displaying and flipping data from the matrix. Generate the methods, for the functionality mentioned above for creation the matrix manipulator.

2. Objectives:

- To understand the concept of Object Oriented Programming.
- To understand how to use of Array in java

3. Outcomes:

To elaborate the concept of arrays

4. Software Required:

Ubuntu, JDK 1.5.0 onwards

5. Theory:

Java provides a data structure, the array, which stores a fixed-size sequential collection of elements of the same type. An array is used to store a collection of data, but it is often more useful to think of an array as a collection of variables of the same type. Instead of declaring individual variables, such as number0, number1, ..., and number99, you declare one array variable such as numbers and use numbers[0], numbers[1], and ..., numbers[99] to represent individual variables. This tutorial introduces how to declare array variables, create arrays, and process arrays using indexed variables.

Declaring Array Variables:

- To use an array in a program, you must declare a variable to reference the array, and you must specify the type of array the variable can reference. Here is the syntax for declaring an array variable:

  dataType[] arrayRefVar; // preferred way

  OR

  dataType arrayRefVar[]; // works but not preferred way

- Note: The style dataType[] arrayRefVar is preferred. The style dataType arrayRefVar[] comes from the C/C++ language and was adopted in Java to accommodate C/C++ programmers.
- Example: The following code snippets are examples of this syntax:

  double[] myList; // preferred way.

  OR

  double myList[]; // works but not preferred way.

Creating Arrays:

- You can create an array by using the new operator with the following syntax:

  arrayRefVar = new dataType[arraySize];

- The above statement does two things:
  1. It creates an array using new dataType[arraySize];
  2. It assigns the reference of the newly created array to the variable arrayRefVar.
- Declaring an array variable, creating an array, and assigning the reference of the array to the variable can be combined in one statement, as shown below:

  dataType[] arrayRefVar = new dataType[arraySize];

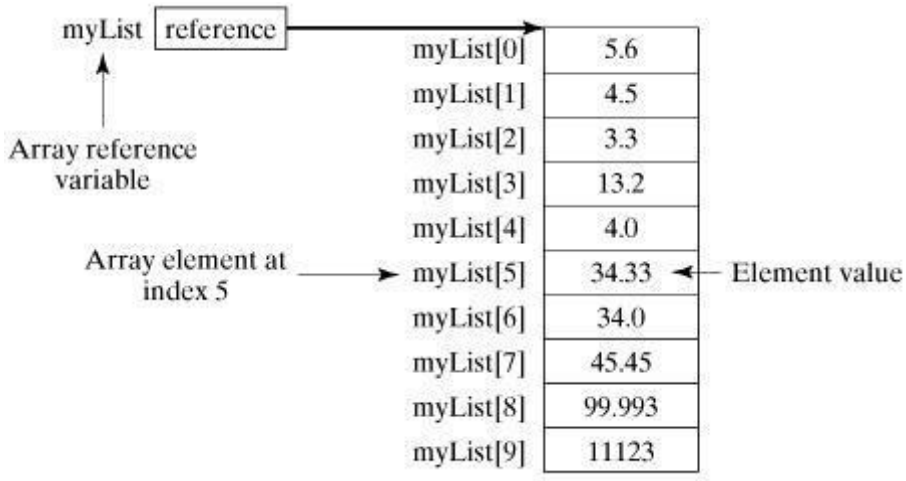
  Alternatively you can create arrays as follows:

  dataType[] arrayRefVar = {value0, value1, ..., valuek};

- The array elements are accessed through the index. Array indices are 0-based; that is, they start from 0 to arrayRefVar.length-1.
- Example: Following statement declares an array variable, myList, creates an array of 10 elements of double type, and assigns its reference to myList:

double[] myList = new double[10];

- Following picture represents array myList. Here myList holds ten double values and the indices are from 0 to 9.



6. Algorithm:

1. Create a class Matrix and declare two 2D array for addition and transpose of matrix and variables r, c, i, j are used.

```
class Matrix
{
  int a[][],b[][],sum[][],t[][];
  int r,c,i,j;
```

2. Create a method read() and enter the elements for two matrices

```
 void read()
   {
      Scanner sc=new Scanner(System.in);
      System.out.println("Enter no. of rows and columns:");
      r=sc.nextInt();
      c=sc.nextInt();
      a =new int[r][c];
```

```
    b =new int[r][c];
    System.out.println("Enter First Matrix:");
    for(i=0;i<r;i++)
     {
       for(j=0;j<c;j++)
        {
         //System.out.print("Enter data:");
          a[i][j]=sc.nextInt();
        }
     }
    System.out.println("Enter Second Matrix:");
    for(i=0;i<r;i++)
     {
       for(j=0;j<c;j++)
        {
          //System.out.print("Enter data:");
          b[i][j]=sc.nextInt();
        }
     }
  } //end of read
```

3. Create a method addition() for the ddition of two matrices

```
 void addition()
  {
    sum=new int[r][c];
    for(i=0;i<r;i++)
    for(j=0;j<c;j++)
```

```
            sum[i][j]=a[i][j]+b[i][j];
    } //end of addition
```

4. Create another method display() to display the sum of matrices

```
  void display()
   {
     for(i=0;i<r;i++)
       {
        for(j=0;j<c;j++)
          System.out.print(sum[i][j]+" ");
        System.out.println();
        }
   } //end of display
```

5. Create a method transpose() to transpose the matrix after the addition of two matrices.

```
  void transpose()
   {
     t=new int[r][c];
     for(i=0;i<c;i++)
        for(j=0;j<r;j++)
           t[i][j]=sum[j][i];
      for(i=0;i<r;i++)
      {
        for(j=0;j<c;j++)
          System.out.print(t[i][j]+" ");
        System.out.println();
        }
```

```
    } //end of transpose
} //end of class Matrix
```

6. Stop

7. Conclusion:

This experiment implements transpose matrix, we also learnt how to create an array data structure which include 2Dimentional array.

8. Quiz / Viva Questions:

- What is array?
- How to access array in java?

9. References:

1. Jaime Nino, Frederick A. Hosch, 'An introduction to Programming and Object Oriented Design using Java', Wiley Student Edition.
2. E Balgurusamy, "Programming with JAVA", Tata McGraw Hill

D Y PATIL
DEEMED TO BE
UNIVERSITY
RAMRAO ADIK
INSTITUTE OF TECHNOLOGY
NAVI MUMBAI

OOP Lab

EXPERIMENT NO.

7

Demonstration of StringBuffer methods

Experiment No. 7

1. Aim:

A character sequence is to be read as an input and result need to declare as "yes" or "no" by investigating the fact that traversing the characters sequence backwards and forwards results in same sequence. Write a program for the same using StringBuffer.

2. Objectives:

- Understand basic concepts in Object Oriented Programming
- Study different operations on strings and string Buffer class

3. Outcomes:

To elaborate the concept of strings

4. Software Required:

Ubuntu, JDK 1.5.0 onwards

5. Theory:

String manipulation is the most common part of many Java programs. Strings are very important in programming languages as they are used to process long textual/symbolic information. The information you provide is in form of ordered sequence of characters and symbols is called as string.

In C, C++ character arrays are used to process long textual information. Using character arrays is time consuming so in java, strings are not considered as the fundamental data type. In Java String class is provided to work with strings along with facility of character array.

In Java, Strings are immutable as once an object of the String class is created and initialized with some value then it cannot be changed. The Java development environment provides two classes that store and manipulate character data: String, for constant strings, and StringBuffer, for mutable strings.

Another way of making a string in java is making an object of StringBuffer class. One major speciality of the object of this class is that it is mutable. The size of this object can be changed again and again during runtime i.e. we can insert, append, and delete etc. a character from the string. An object of StringBuffer has some advantages of extra methods provided in the class. Let us see methods supported in this class.

There are some StringBuffer methods given as below:

append(String): Append the string with original string.

insert(int,char): This method is used to insert the character passed at indexed passed.

replace(int startIndex, int endIndex, String str): This method is used to replace the string from specified startIndex and endIndex.

delete(int startIndex, int endIndex): This method is used to delete the string from specified startIndex and endIndex.

reverse(): Rreverses the Original string.

capacity(): This method is used to return the current capacity of the StringBuffer object.

char charAt(int index): This method is used to return the character at the specified position.

setcharAt(int,index): The character passed is set at the index specified in the brackets.

length(): This method is used to return the length of the string i.e. total number of characters.

public String substring(int beginIndex, int endIndex): is used to return the substring from the specified beginIndex and endIndex.

- **Defining string with String class:**

    1. String str = new String ( ); str = "Hello";

    2. String str = "Hi";

- **Defining string with StringBuffer class:**

  StringBuffer str = new StringBuffer("Hello");

- **Defining array of strings:**

  String array[] ={"str1", "str2", "str3" , "str4"};

6. Algorithm:

Step 1: Start.
Step 2: Define class Palindrome.
Step 3: Declare 2 String variables str and rev.
Step 4: Take input from user for String str
Step 5: Declare StringBuffer variable str1 using String str as a parameter
Step 6: Reverse StringBuffer str1
Step 7: Convert StringBuffer str1 to String and store it in String rev
Step 8: if String str equals to String rev
    Step 8.1: Display String str is a palindrome
    Step 8.2: else display String str is not a palindrome
Step 9: Stop

7. Conclusion:

With this experiment we have learned difference between String and StringBuffer, different methods of String class and StringBuffer class. We have checked whether the given string is palindrome or not and also we are able to solve real time problems.

8. Quiz / Viva Questions:

- What is difference between String and StringBuffer?
- What are the methods used in StringBuffer Class?

9. References:

1. Jaime Nino, Frederick A. Hosch, 'An introduction to Programming and Object Oriented Design using Java', Wiley Student Edition.
2. E Balgurusamy, "Programming with JAVA", Tata McGraw Hill

D Y PATIL
DEEMED TO BE
UNIVERSITY
RAMRAO ADIK
INSTITUTE OF TECHNOLOGY
NAVI MUMBAI

OOP Lab

EXPERIMENT NO.

8

Demonstration of Multilevel Inheritance

Experiment No. 8

1. Aim:

Write a program to calculate volume of sphere using multilevel inheritance. The base class method will accept the radius from user. A class will be derived from the above mentioned class that will have a method to find the area of a circle derived from this will have method to calculate and display the volume of the sphere.

2. Objectives:

- To understand the concept of inheritance.
- To understand how to use method overriding.

3. Outcomes:

To understand and demonstrate the concept of inheritance

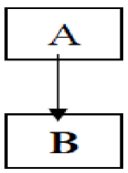
4. Software Required:

Ubuntu, JDK 1.5.0 onwards

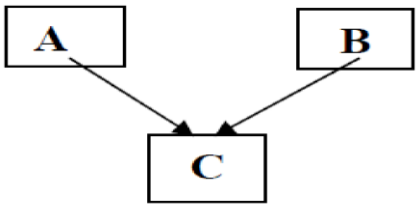
5. Theory:

Types of Inheritance:

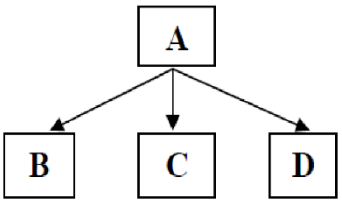
1. Single Inheritance: A derived class with only one base class is called as Single Inheritance.



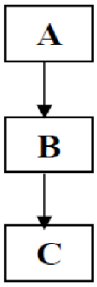
2. Multiple Inheritances: A derived class with several base classes is called Multiple Inheritance.



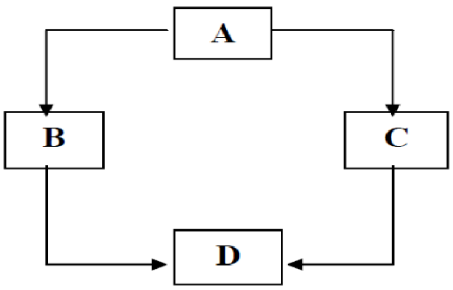
3. Hierarchical Inheritance: More than one class may inherit the features of one class this process is called as Hierarchical Inheritance.



4. Multilevel Inheritance: The mechanism of deriving a class from another derived class is called Multilevel Inheritance.



5. Hybrid Inheritance: There could be situations where we need to apply one or more types of inheritances to design a program this process is called Hybrid Inheritance.



Syntax:

```
class  A

{

}

class  B extends A

{

 }

class  C extends B

{

}
```

Overriding and Hiding Methods

Instance Methods

An instance method in a subclass with the same signature (name, plus the number and the type of its parameters) and return type as an instance method in the superclass overrides the superclass's method. The ability of a subclass to override a method allows a class to inherit from a superclass whose behavior is "close enough" and then to modify behavior as needed. The overriding method has the same name, number and type of parameters, and return type as the method it overrides. An overriding method can also return a subtype of the type returned by the overridden method. This is called a covariant return type.

When overriding a method, you might want to use the @Override annotation that instructs the compiler that you intend to override a method in the superclass. If, for some reason, the compiler detects that the method does not exist in one of the superclasses, it will generate an error.

Class Methods

If a subclass defines a class method with the same signature as a class method in the superclass, the method in the subclass hides the one in the superclass.

The distinction between hiding and overriding has important implications. The version of the overridden method that gets invoked is the one in the subclass. The version of the

hidden method that gets invoked depends on whether it is invoked from the superclass or the subclass. Let's look at an example that contains two classes. The first is Animal, which contains one instance method and one class method:

```
public class Animal {
    public static void testClassMethod() {
        System.out.println("The class" + " method in Animal.");
    }
    public void testInstanceMethod() {
        System.out.println("The instance " + " method in Animal.");
    }
}
```

The second class, a subclass of Animal, is called Cat:

```
public class Cat extends Animal {
    public static void testClassMethod() {
        System.out.println("The class method" + " in Cat.");
    }
    public void testInstanceMethod() {
        System.out.println("The instance method" + " in Cat.");
    }
    public static void main(String[] args) {
        Cat myCat = new Cat();
        Animal myAnimal = myCat;
        Animal.testClassMethod();
        myAnimal.testInstanceMethod();
    }
}
```

The Cat class overrides the instance method in Animal and hides the class method in Animal. The main method in this class creates an instance of Cat and calls testClassMethod() on the class and testInstanceMethod() on the instance.

The output from this program is as follows:

The class method in Animal.

The instance method in Cat.

6. Algorithm:

Step 1: Start

Step 2: Create base class Circle with method to accept radius from user.

Step 3: Derive class Area from the base class with methods calculate() to calculate area and display() to display area.

Step 4: Derive another class Volume from Area which will calculate volume of sphere and have method display () to display the volume.

Step 5: Stop

Here, the method display () of the derived class 'Volume' overrides the method display () of the class 'Area'

7. Conclusion:

From this experiment we have learnt how to inherit the property of base class. Also learnt how to perform hiding and overriding concept.

8. Quiz / Viva Questions:

- What is multilevel inheritance?
- What is method overriding?

9. References:

1. Jaime Nino, Frederick A. Hosch, 'An introduction to Programming and Object Oriented Design using Java', Wiley Student Edition.
2. E Balgurusamy, "Programming with JAVA", Tata McGraw Hill

D Y PATIL
DEEMED TO BE
UNIVERSITY
RAMRAO ADIK
INSTITUTE OF TECHNOLOGY
NAVI MUMBAI

OOP Lab

EXPERIMENT NO.

9

Demonstration of Interface

Experiment No. 9

1. Aim:

Consider a university where students who participate in the National Games or Olympics are given some grace marks. The grace marks provided are fixed and same for every student. Create an application that keeps student's academic marks and Sports grace marks separate and generate total of marks considering academics and sports both. Also invoke methods of base class & interface using reference. (Hint: Make use of Interface).

2. Objectives:

- Study feature of OOP like inheritance.
- Study implementation of multiple inheritance in Java

3. Outcomes:

To understand and demonstrate the concept of interfaces

4. Software Required:

Ubuntu, JDK 1.5.0 onwards

5. Theory:

The mechanism of inheriting the features of more than one base class into a single class is known as multiple inheritances. Java does not support multiple inheritance but the multiple inheritance can be achieved by using the interface.

In Java Multiple Inheritance can be achieved through use of Interfaces by implementing more than one interface in a class. An interface is a group of related methods with empty bodies. Interfaces define only abstract methods and final fields. Therefore it is the

responsibility of the class that implements an interface to define the code for implementation of these methods.

Syntax:

```
interface InterfaceName
{
        return_type method_name1(parameter-list);
        data_type variable_name =value;
}
```

6. Algorithm:

Step 1: Start

Step 2: Create class Student

2.1 Declare roll_no as int type

2.2 Write getnumber() method to get roll number of a student

2.3 Write putnumber() method to print roll number of a student

Step 3: Define class test which extends student

3.1 Declare sem1, sem2 as float type

3.2 Write getmarks() method to get marks of a student

3.3 Write putmarks() method to print marks of a student

Step 4: Define interface sports

4.1 Declare score of float type

4.2 Declare putscore();

Step 5: Define class Result which extends test & implements sports

5.1 Declare total of type float

5.2.1 Write display() method; Calculate total of sem1,sem2 & score

5.2.2 Invoke putnumber() ,putmarks(),putscore() methods

5.2.3 Print total

5.3 Write putscore() method to display score

Step 6: Define class Hybrid

Step 7: Define Main method, Create object of class Result and Invoke getnumber(), getmarks(), display().

Step 8: Stop

7. Conclusion:

With this experiment we have understood that Interfaces are mainly used to provide polymorphic behavior and its function to break up the complex designs and clear the dependencies between objects.

8. Quiz / Viva Questions:

- What is interface?
- How multiple inheritance can be implemented in java?
- What are advantages and disadvantages of interface?

9. References:

1. Ralph Bravaco , Shai Simoson , "Java Programing From the Group Up" ,Tata McGraw-Hill
2. Grady Booch, Object Oriented Analysis and Design
3. Jaime Nino, Frederick A. Hosch, 'An introduction to Programming and Object Oriented Design using Java', Wiley Student Edition. Wayne Tomasi, "Electronics Communication Systems", Pearson education, Fifth edition.
4. Java 2, "The Complete Reference of Java", Schildt publication

D Y PATIL
DEEMED TO BE
UNIVERSITY
RAMRAO ADIK
INSTITUTE OF TECHNOLOGY
NAVI MUMBAI

OOP Lab

EXPERIMENT NO.

10

Create User-Defined Exception

Experiment No. 10

1. Aim:

Through Custom exception user can raise application-specific error code. You are required to calculate a square of even number provided as input by user. However if a user provides an odd number as input, then an exception must be thrown explicitly with message indicating the input number must be even number.

2. Objectives:

- Study feature of OOP like exception handing.
- Study how to create own exception in Java

3. Outcomes:

To interpret and apply the notion of exception handling.

4. Software Required:

Ubuntu, JDK 1.5.0 onwards

5. Theory:

Exceptions are usually used to denote something unusual that does not conform to the standard rules. In programming, exceptions are events that arise due to the occurrence of unexpected behaviour in certain statements, disrupting the normal execution of a program. Exception is a run time error. Exceptions can arise due to a number of situations. For example,

- Trying to access the 11th element of an array when the array contains of only 10 element (ArrayIndexOutOfBoundsException)
- Division by zero (ArithmeticException)

- Accessing a file which is not present (FileNotFoundException)
- Failure of I/O operations (IOException)
- Illegal usage of null. (NullPointerException)

Top class in exception hierarchy is Throwable. This class has two siblings: Error and Exception. All the classes representing exceptional conditions are subclasses of the Exception class.

The exception handling mechanism consists of following elements:

- **try/catch:** All the statements to be tried for exceptions are put in a try block. The catch block is used to catch any exception raised from the try block. If exception occurs in any statement in the try block control immediately passes to the corresponding catch block.
- **finally:** The finally block will execute whether or not an exception is thrown.
- **throw:** It is used to explicitly throw an exception. It is useful when we want to throw a user-defined exception.
- **throws:** If a method is capable of causing an exception that it does not handle, it must specify this behavior using throws keyword so that callers of the method can guard themselves against that exception

6. Algorithm:

1. Start
2. Create class OddException which extends Exception class
3. Declare num as int type
4. Write OddException(int x) constructor to get numbers as input.
5. Write toString() method to display a message when exception is thrown.
6. Create a class MyExceptionDemo with main function and Write static function OddNoException() throws OddException when number is odd.
7. Define main method
8. In try block
9. Invoke OddException(3)
10. In catch block
11. Print catch exception
12. Stop

7. Conclusion:

With this experiment we implemented exception handling mechanism and understood how to create and handle user defined exception.

8. Quiz / Viva Questions:

- Define exception.
- Differentiate between error and exception.
- What is meant by the runtime exception?
- What are different types of exceptions?
- Explain role of try catch.
- What is use of finally block and when it gets executed?
- How to create user defined exception.

9. References:

1. Java 2, "The Complete Reference of Java", Schildt publication.
2. Ivor Horton, 'Beginning JAVA', Wiley India.
3. DietalandDietal, 'Java: How to Program', 8/e,PHI
4. 'JAVA Programming', Black Book, Dreamtech Press.

D Y PATIL
DEEMED TO BE
UNIVERSITY
RAMRAO ADIK
INSTITUTE OF TECHNOLOGY
NAVI MUMBAI

OOP Lab

EXPERIMENT NO.

11

Demonstration of Multithreading

Experiment No. 11

1. Aim:

Divide your program into two parts: One to read a number and the other to calculate its square. Provide a simultaneous execution of both parts of a program to maximum utilize the CPU time.

2. Objectives:

- Study basic constructs of java
- Understand importance of multithreading

3. Outcomes:

To interpret and apply the notion of multithreading.

4. Software Required:

Ubuntu, JDK 1.5.0 onwards

5. Theory:

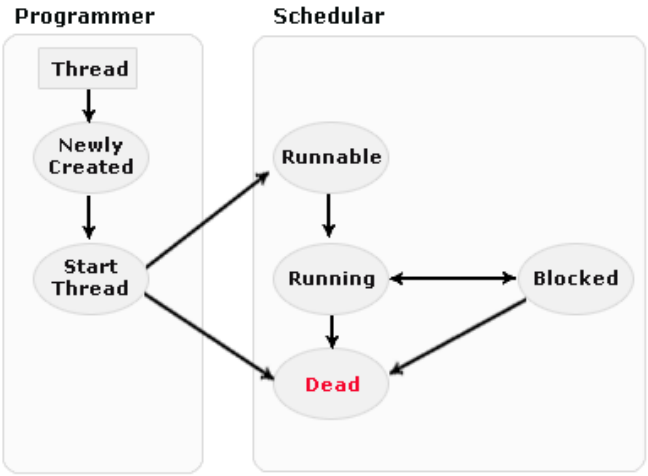
Processes and Threads: In concurrent programming, there are two basic units of execution: processes and threads. In the Java programming language, concurrent programming is mostly concerned with threads.

Processes: A process has a self-contained execution environment. A process generally has a complete, private set of basic run-time resources; in particular, each process has its own memory space.

Threads: Threads are sometimes called lightweight processes. Both processes and threads provide an execution environment, but creating a new thread requires fewer resources than creating a new process. Threads exist within a process — every process has at least

one thread. Threads share the process's resources, including memory and open files. Multithreaded execution is an essential feature of the Java platform. Every application has at least one thread — or several, if you count "system" threads that do things like memory management and signal handling. But from the application programmer's point of view, you start with just one thread, called the main thread.

Different states of a thread are:



New state: After the creations of Thread instance the thread is in this state but before the start() method invocation. At this point, the thread is considered not alive.

Runnable (Ready-to-run) state: A thread start its life from Runnable state. A thread first enters runnable state after the invoking of start() method but a thread can return to this state after either running, waiting, sleeping or coming back from blocked state also. On this state a thread is waiting for a turn on the processor.

Running state: A thread is in running state that means the thread is currently executing. There are several ways to enter in Runnable state but there is only one way to enter in running state: the scheduler select a thread from runnable pool.

Dead state: A thread can be considered dead when its run() method completes. If any thread comes on this state that means it cannot ever run again.

Blocked: A thread can enter in this state because of waiting the resources that are hold by another thread.

As we have seen different states that may be occur with the single thread. A running thread can enter to any non-runnable state, depending on the circumstances. A thread cannot enter directly to the running state from non-runnable state, firstly it goes to

runnable state. Now let's understand the some non-runnable states which may be occur handling the multithreads.

Sleeping: On this state, the thread is still alive but it is not runnable, it might be return to runnable state later, if a particular event occurs. On this state a thread sleeps for a specified amount of time. You can use the method sleep( ) to stop the running state of a thread.

static void sleep(long millisecond) throws InterruptedException

Waiting for Notification: A thread waits for notification from another thread. The thread sends back to runnable state after sending notification from another thread.

final void wait(long timeout) throws InterruptedException

final void wait(long timeout, int nanos) throws InterruptedException

final void wait() throws InterruptedException

Some Important Methods defined in java.lang.Thread are shown in the table:

Method	Return Type	Description
currentThread( )	Thread	Returns an object reference to the thread in which it is invoked.
getName( )	String	Retrieve the name of the thread object or instance.
start( )	Void	Start the thread by calling its run method.
run( )	Void	This method is the entry point to execute thread, like the main method for applications.
sleep( )	Void	Suspends a thread for a specified amount of time (in milliseconds).
isAlive( )	Boolean	This method is used to determine the thread is running or not.
activeCount( )	Int	This method returns the number of active threads in a particular thread group and all its subgroups.
interrupt( )	Void	The method interrupt the threads on which it is invoked.
yield( )	Void	By invoking this method the current thread pause its execution temporarily and allow other threads to execute.
join( )	Void	This method and join(long millisec) Throws InterruptedException. These two methods are invoked on a thread. These are not returned until either the thread has completed or it is timed out respectively.

Threads can be created in Java in the following ways:

- Extending the java.lang.Thread class
- Implementing the java.lang.Runnable Interface.

1. Extending the java.lang.Thread class:

This is a simplest form of creating threads in a program. Consider the following code snippet.

```
public class ThreadExample extends Thread
{
    public void run()
    {
        System.out.println("This is an example on extending thread class");
    }
    -----
    -----
}
```

In this code snippet, the ThreadExample class is created by extending the Thread class. The run( ) method of the Thread class is overridden by the ThreadExample class to provide the code to be executed by a thread. In Java we cannot extend a class from more than one class. Therefore while creating a thread by extending the Thread class we cannot simultaneously extend any other class.

2. Extending the java.lang.Runnable Interface:

Threads can also be created by implementing the Runnable interface of the java.lang package. This package allows to define a class that can implement the Runnable interface and extend any other class. Consider the following code snippet.

```
public class ThreadExample implements Runnable
{
    public void run()
    {
```

```
                System.out.println("This is an example on runnable interface");
        }
        public static void main(String[] args)
        {
                ThreadExample thrd = new ThreadExample( );
                Thread T = new Theard(thrd);
         }
}
```

In above code snippet, the ThreadExample class is declared by implementing Runnable interface and overriding the run( ) method in which it defines the code to be executed by a thread.

6. Algorithm:

1. Start
2. Define class which implements Runnable interface.
   2.1 Create Thread t1.
   2.2 Create run( ) method to print 1 to 10 numbers
       for(int i=1;i<=10;i++)
       print i
3. Define class which implements Runnable interface.
   3.1 Create Thread t2.
   3.2 Create run() method to print it's square
       for(int i=1;i<=10;i++)
       print i*i
4. Define class multithread
5. Define main method

6. Invoke t1.start( ) then t2.start( )

7. Stop

7. Conclusion:

In this experiment we have implemented multithreading which is helpful to improve performance of system.

8. Quiz / Viva Questions:

- What is multithreading?
- What are advantages of multithreading?
- What is synchronization in multithreading?

9. References:

1. Ralph Bravaco , Shai Simoson , "Java Programing From the Group Up" ,Tata McGraw-Hill
2. Grady Booch, Object Oriented Analysis and Design
3. Jaime Nino, Frederick A. Hosch, 'An introduction to Programming and Object Oriented Design using Java', Wiley Student Edition. Wayne Tomasi, "Electronics Communication Systems", Pearson education, Fifth edition.
4. Java 2, "The Complete Reference of Java", Schildt publication.

D Y PATIL
DEEMED TO BE
UNIVERSITY
RAMRAO ADIK
INSTITUTE OF TECHNOLOGY
NAVI MUMBAI

OOP Lab

EXPERIMENT NO.

12

Creating GUI application

Experiment No. 12

1. Aim:

Create a GUI application which is fully equipped with the functionality of solving various computation problems such as Addition, subtraction, Multiplication and Division. The GUI application should runs in a browser and displays the output as result of these mathematical operations.

2. Objectives:

- To explain components of GUI based programming

3. Outcomes:

To develop GUI based application to understand event-based GUI handling principles.

4. Software Required:

Ubuntu, JDK 1.5.0 onwards

5. Theory:

Swing is a part of Java Foundation classes (JFC), the other parts of JFC are java2D and Abstract window toolkit (AWT). AWT, Swing & Java 2D are used for building graphical user interfaces (GUIs) in java. In this tutorial we will mainly discuss about Swing API which is used for building GUIs on the top of AWT and are much more light-weight compared to AWT.

The main characteristics of this architecture are:

- The swing component's data is represented using Model.
- It is visually represented using a view.

- The controller component of the MVC architecture reads input from the user on the view and then these changes are passed to the component data.
- In each Swing component, the view and controller are clubbed together while the model is a separate one. This gives swing a pluggable look and feel feature.

The features of the swing API are summarized below:

- Swing components are platform-independent.
- The API is extensible.
- Swing components are light-weight. The swing components are written in pure Java and also components are rendered using Java code instead of underlying system calls.
- Swing API provides a set of advanced controls like TabbedPane, Tree, Colorpicker, table controls, etc. which are rich in functionality.
- The swing controls are highly customizable. This is because the appearance or look-and-feel of the component is independent of internal representation and hence we can customize it in the way we desire.
- We can simply change the values and thus alter the look-and-feel at runtime

JFrame: A frame is an instance of JFrame. Frame is a window that can have title, border, menu, buttons, text fields and several other components. A Swing application must have a frame to have the components added to it.

JPanel: A panel is an instance of JPanel. A frame can have more than one panels and each panel can have several components. You can also call them parts of Frame. Panels are useful for grouping components and placing them to appropriate locations in a frame.

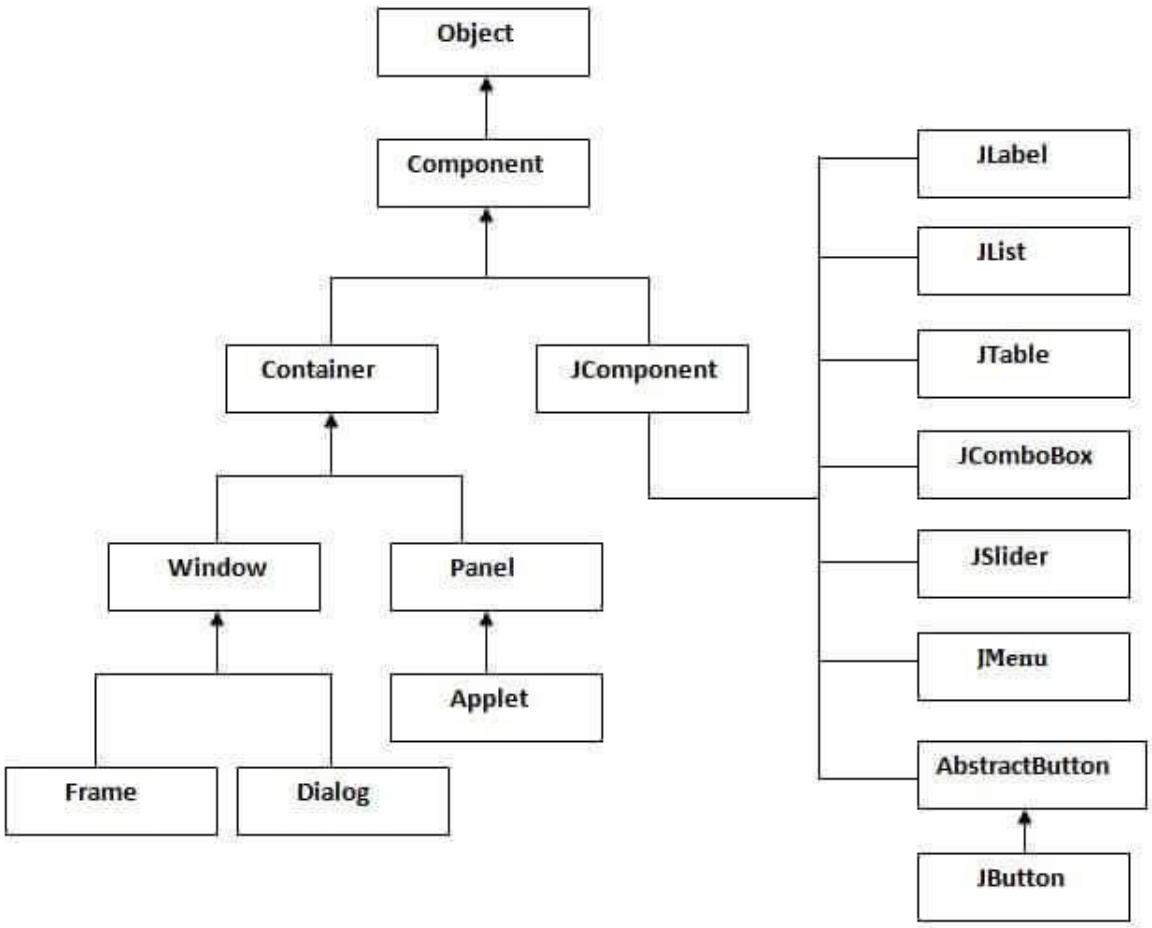
JLabel: A label is an instance of JLabel class. A label is unselectable text and images. If you want to display a string or an image on a frame, you can do so by using labels. In the above example we wanted to display texts "User" & "Password" just before the text fields, we did this by creating and adding labels to the appropriate positions.

JTextField: Used for capturing user inputs, these are the text boxes where user enters the data.

JPasswordField: Similar to text fields but the entered data gets hidden and displayed as dots on GUI.

JButton: A button is an instance of JButton class. In the above example we have a button "Login".

The hierarchy of java swing API is given below:



Commonly used Methods of Component class:

The methods of Component class are widely used in java swing that are given below,

Method	Description
public void add(Component c)	add a component on another component.
public void setSize(int width,int height)	sets size of the component.
public void setLayout(LayoutManager m)	sets the layout manager for the component.
public void setVisible(boolean b)	sets the visibility of the component. It is by default false.

6. Algorithm:

1. Start
2. Import the swing packages and awt packages.
3. Create the class scientificcalculator that implements action listener.
4. Create the container and add controls for digits, scientific calculations and decimal Manipulations.
5. The different layouts can be used to lay the controls.
6. When the user presses the control, the event is generated and handled.
7. The corresponding decimal, numeric and scientific calculations are performed.
8. Stop

7. Conclusion:

In this experiment we have implemented GUI application with swing components. We have understood how to design any interface and how to pass parameters.

8. Quiz / Viva Questions:

- What is Swing?
- How it is different from application program?
- What are the components of Swing?

9. References:

1. Java 2, "The Complete Reference of Java", Schildt publication.
2. Ivor Horton, 'Beginning JAVA', Wiley India.
3. DietalandDietal, 'Java: How to Program', 8/e,PHI
4. 'JAVA Programming', Black Book, Dreamtech Press.

D Y PATIL
DEEMED TO BE
UNIVERSITY
RAMRAO ADIK
INSTITUTE OF TECHNOLOGY
NAVI MUMBAI