ALGORITHM:

Factorial

1. Create class Factorial
2. Declare integer variable 'n'
3. Accept value on *n* using Scanner class
4. Define method computeFactorial & implement logic to find factorial of number
5. Display value of factorial

Prime Numbers

1. Create class PrimeNumbers
2. Declare integer variable 'n'
3. Accept value on *n* using Scanner class
4. Implement logic to find prime numbers upto *n*
5. Display all prime numbers

Sum & Average

1. Create class SumAvg
2. Declare integer variable 'n'
3. Accept value on *n* using Scanner class
4. Define method computeSum & implement logic to find sum of *n* numbers
5. Define method computeAvg & implement logic to find sum of *n* numbers
6. Define main( ) method
7. Display sum & average value

}

ALGORITHM:

1. Define class Calculator
2. Accept two numbers using Scanner class
3. Display operations available
4. Accept choice from user
5. Define methods add, subtract, divide, multiply & factorial to implement respective logic
6. Define main( ) method
7. Use switch case statement to perform desired operation
8. Provide option to continue or stop.

CONCLUSIONS:

1. What are default values of primitive data types?
2. Explain different control statements?
3. Explain enhanced for loop.

Any modification done in callee method will have no effect on the original para
method.

ALGORITHM:

1. Define class Rectangle.
2. Define attributes width, length, area and color.
3. Define parameterized constructor in class rectangle.
4. Define methods to read width, length and color.
5. Define method rectArea to compute area of rectangle.
6. Define method colorCompare to compare colors of rectangle
7. Define main( ) method
8. Initialize variables using parameterized constructor

types.

ALGORITHM:

1. Define a class Adder
2. Define 3 instance variables of type int
3. Define parameterized constructor with two arguments & initialize first two instance variables
4. Define another parameterized constructor with three arguments & initialize all three instance variables
5. Define method add with return type int and pass first two instance variables to this method. Return the addition of two variables passed.

6. Define overloaded method add with return type int and pass all three instance variables to this method. Return the addition of three variables passed.
7. Define another class TestAdder
8. Create first object of class Adder by passing two values
9. Call required method add
10. Create second object of class Adder by passing three values
11. Call required method add
12. Display results of addition properly

CONCLUSIONS:

1. What is method overloading?
2. What is constructor overloading?
3. What are advantages of overloading?
4. Differentiate between overloading and overriding?

ALGORITHM:

1. Design a class for a ArraySort with method arraySort() to sort integer array
2. Design a class for a StringSort with methods stringSort() to sort string array.
3. Declare main class with main method.
4. Create two arrays of type int and String.
5. Input the elements of respective array from user.
6. Create objects of ArraySort and StringSort classes.
7. Call method from these classes to sort respective integer or string arrays.
8. Also sort arrays directly using inbuilt sort method.
9. Display original and sorted arrays using both ways.

CONCLUSIONS:

1. What is need of wrapper class?
2. Compare the process of simple array and array of objects creation.
3. Compare arrays instantiating process in Java.
4. What are the advantages and disadvantages of Array?
5. How do you compare the two arrays in java?

4. **row_size**: Number of Row elements an array can store. For example, row_size = 5, then the array will have five rows.
5. **column_size**: Number of Column elements an array can store. For ex, column_size = 6, then the array will have 6 Columns.

   Ex. double[][] Employees = new double[5][6];

ALGORITHM:

1. Identify and design a methods main class arr_add.
2. Input number of rows and columns for two dimensional matrix.
3. Create three arrays as a, b and c as objects, with two dimensional rows and columns entered above.
4. Input elements for matrix a and b.
5. Use Loop through the arrays a and b, add the corresponding elements e.g $a_{11}+b_{11}=c_{11}$
6. Store result of addition in matrix c.
7. Display matrix c.

ALGORITHM:

1. Create player class with attributes as name, age & ranking.
2. Add display method that displays all above parameters.
3. Inherit class Cricket_Player with attributes as game & role.
4. Add display method that calls parent class display() method & displays attributes.
5. Inherit class Football_Player with attributes as game & place.
6. Add display method that calls parent class display() method & displays attributes.
7. Inherit class Hockey_Player with attributes as game & position.
8. Add display method that calls parent class display() method & displays attributes.
9. Create main method & call all methods.

CONCLUSIONS:

1. State & explain different forms of inheritance in Java.
2. What is the need of inheritance?
3. Which type of inheritance is not allowed in Java Class? Why?

ALGORITHM:

1. Create interface Shape with variable pi & methods area() & perimeter ().
2. Declare constructor.
3. Create class Circle that implements interface & has variable radius.
4. It will implement both methods of interface.
5. Create class Rectangle that implements interface & has variables length & width.
6. Declare constructor & call super class constructor.
7. Implement methods of interface.
8. Create class Ellipse that implements interface & has variables major & minor.
9. Declare constructor & call super class constructor.
10. Implement methods of interface.
11. Create main class that displays area & perimeter for Circle, Rectangle & Ellipse.

CONCLUSIONS:

1. Differentiate between abstract class & interface?
2. State types of declaring interfaces?
3. After compilation of interface state whether .class file is generated or not.

There are 5 keywords which are used in handling exceptions in Java:

Sr.No.	Method
try	The *try* keyword is used to specify a block where we should place exception code. The *try* block must be followed by either *catch* or *finally*. It means, we can't use *try* block alone.
catch	The *catch* block is used to handle exception. It must be preceded by *try* block which means we can't use *catch* block alone. It can be followed by *finally* block later.
finally	The *finally* block is used to execute the important code of the program. It is executed whether an exception is handled or not.
throw	The *throw* keyword is used to throw an exception.
throws	The *throws* keyword is used to declare exceptions. It doesn't throw an exception. It specifies that there may occur an exception in the method. It is always used with method signature.

ALGORITHM:

1. Define a class
2. Create a divide by zero exception
3. Create ArrayIndexOutofBound exception.
4. Use try-catch block to handle exception
5. Print stack trace of both exceptions.
6. Use finally block & display message "Finally Block Executed".

CONCLUSIONS:

1. Define exception.
2. State difference between error & exception.
3. Explain exception handling mechanism in Java.

ALGORITHM:

1. Create an applet AppletDemo
2. Define paint() method
3. Define oval shape with drawOval() & fillOval() methods with appropriate parameters.
4. Define oval shape with drawRect() & fillRect() methods with appropriate parameters
5. Define rectangle shape with drawOval() & fillOval() methods with appropriate parameters
6. Define circle shape with drawOval() & fillOval() methods with appropriate parameters
7. Define line with drawLine() methods with appropriate parameters
8. Define arc shape with drawArc() & fillArc() methods with appropriate parameters.
9. Display text message with drawString() method.

CONCLUSIONS:

1. Differentiate between Applet & Application?
2. Explain awt package & graphics class in Java?
3. Explain Swing in Java.

ALGORITHM:

1. Create a class
2. Create object of FileReader
3. Create object of BufferedReader
4. Create object of FileWriter
5. Create object of BufferedWriter
6. Read data from file1.txt line by line
7. Write date into file2.txt
8. Close objects

CONCLUSIONS:

1. Explain file management in Java?
2. Explain byte & character data handling in Java Files?

```
class ThreadDemo implements Runnable{
    public void run(){
        //Thread Definition
    }
}
```

Step 2: Create thread by defining object that is instantiated from runnable class as targ of thread.

Step 3: Once Thread object is created, start it by calling `start()` method, whi executes call to `run()` method.

```
public class TestThread{
    public static void main(String args[]){
        ThreadDemo obj = new ThreadDemo();
        Thread t1= new Thread(obj);  //Thread Object
        t1.start();
    }
}
```

ALGORITHM:

1. Create a class
2. Implement thread that prints values from 1 to 5
3. Create another class
4. Implement thread that prints values from A to E
5. Create main class
6. Execute both threads

CONCLUSIONS:

1. Explain multithreading in Java?
2. Explain two ways to implement thread?
3. State advantages of multithreading.