M.Sc. [Information Technology]  
313 14  
LAB: OBJECT ORIENTED PROGRAMMING AND JAVA  
I - Semester

ALAGAPPA UNIVERSITY  
[Accredited with 'A+' Grade by NAAC (CGPA: 3.64) in the Third Cycle and Graded as Category-I University by MHRD-UGC]  
KARAIKUDI – 630 003  
DIRECTORATE OF DISTANCE EDUCATION
ALAGAPPA UNIVERSITY

[Accredited with 'A+' Grade by NAAC (CGPA:3.64) in the Third Cycle and Graded as Category-I University by MHRD-UGC]

(A State University Established by the Government of Tamil Nadu)

KARAUKUDI – 630 003

Directorate of Distance Education

M.Sc. [Information Technology]

I - Semester

313 14

LAB: OBJECT ORIENTED PROGRAMMING AND JAVA
LAB: OBJECT ORIENTED PROGRAMMING AND JAVA

Syllabi

BLOCK 1: JAVA FUNDAMENTAL PROBLEMS
1. Simple Java Problems
2. Class and objects
3. Conditional control using java
4. Looping using java

BLOCK 2: OOP CONCEPTS
5. Function overloading programs
6. Operator overloading programs
7. Inheritance programs, Packages
8. Polymorphism programs, Message passing programs

BLOCK 3: THREAD & VIRTUAL FUNCTION
9. Threads
10. Virtual function programs

BLOCK 4: I/O AND EXCEPTION HANDLING
11. Exception handling programs
12. I/O manipulation programs

BLOCK 5: NETWORK PROGRAMMING
13. Applet programs
14. Implementation of simple network programs using java
Java is a third generation programming language which implements the concept of Object-Oriented Programming (OOPs). It inherits many features of the existing languages, C and C++, along with the addition of new features, making it a simple object-oriented language that is also easy to learn. Java can either have single or compound statements. Java has control statements that are broadly classified into three categories, namely conditional statements, iteration statements and jump statements. The main objective of object-oriented programming is to present various real-world objects as program elements. All concepts related to object-oriented programming, such as data abstraction, encapsulation, inheritance and polymorphism, are implemented with the help of classes. Working with actual data requires a mechanism that deals with a collection of data items. In Java, different data types like arrays and vectors are offered to handle such collections.

This lab manual, Object Oriented Programming and Java, contains several programs based on java concepts, such as classes, function overloading, operator overloading, threads, exception handling and applet to provide the concept of programming. In addition, it will help students in coding and debugging their programs. The manual provides all logical, mathematical and conceptual programs that can help to write programs very easily in java language. These exercises shall be taken as the base reference during lab activities for students. There are also many Try Yourself Questions provided to students for implementation in the lab.
LAB REQUIREMENTS:

To write and run a java program, you need to install a software like J2SDK 1.7. SDK stands for system development kit. SDK is also known as JDK (Java Development Kit) which contains JRE (Java Runtime Environment). It provides a platform that enables the program to run on your computer.

Following are the steps given below that explains how to write and execute a java program.

**Step 1:** Write a Java code using text editor (notepad).

1. Write a program to print hello java.

   ```java
   //main class
   public class Sample1
   {
   public static void main(String args[])
   {
   System.out.println("Hello Java");
   }
   }
   ```

**Step 2:** Save the file as Sample1.java. We have named the file as Sample1, the thing is that we should always name the file same as the public class name. In our program, the public class name is Sample1. So, our file name should be Sample1.java.

**Step 3:** Set environment variable.

Follow the steps to set the environment variable:

Right Click on MyComputer → Properties → Advanced System settings → Inside Advanced tab

Click Environment variables → Inside System Variables click New → Give variable name (For example var) → Give variable value. It is path in your system where java compiler is available (For example variable value :C:\Program Files\Java\jdk1.6.0_23\bin ). Inside bin javac is Java compiler.
Click Ok.

Step 4: Go to command prompt by using start → Run → cmd OR start → type cmd in search program and file.

Step 5: Write following command for compilation of program.

javac Sample1.java

Step 6: To run program use, the following command.

java Sample1

Output:

Hello Java

2. Write a program to add two integers and print it on the screen.

```java
public class AddTwoIntegers
{

    public static void main(String[] args)
    {

        int first = 10;
        int second = 20;

        int sum = first + second;

        System.out.println("The sum is: "+ sum);
    }
}
```
3. Write a program to multiply two floating point numbers.

```java
public class MultiplyTwoNumbers {
    public static void main(String[] args) {
        float first = 1.5f;
        float second = 2.0f;
        float product = first * second;
        System.out.println("The product is: " + product);
    }
}
```

Output:

```
The product is: 3.0
```
4. Write a program to swap two numbers using a temporary variable.

```java
public class SwapNumbers {
    public static void main(String[] args) {
        float first = 1.20f, second = 2.45f;
        System.out.println("—Before swap—");
        System.out.println("First number = "+ first);
        System.out.println("Second number = "+ second);

        // Value of first is assigned to temporary
        float temporary = first;

        // Value of second is assigned to first
        first = second;

        // Value of temporary (which contains the initial
        // value of first) is assigned to second
        second = temporary;
        System.out.println("—After swap—");
        System.out.println("First number = "+ first);
        System.out.println("Second number = "+ second);
    }
}
```
5. Write a program to print the largest number among the three numbers.

```java
public class Largest {
    public static void main(String[] args) {
        double n1 = -4.5, n2 = 3.9, n3 = 2.5;
        if (n1 >= n2 && n1 >= n3)
            System.out.println(n1 + " is the largest number.");
        else if (n2 >= n1 && n2 >= n3)
            System.out.println(n2 + " is the largest number.");
        else
            System.out.println(n3 + " is the largest number.");
    }
}
```
Try yourself:
1. Write a Java program to divide two numbers and print on the screen.
2. Write a Java program to print the result of the following operations.
   a. \(-5 + 8 \times 6\)
   b. \((55+9) \% 9\)
3. Write a Java program to print the sum (addition), multiply, subtract, divide and remainder of two numbers
4. Write a Java program to print the area and perimeter of a circle.

6. Write a program to demonstrate the implementation of class, object and constructor.

```java
//main class
public class Puppy
{
    // This constructor has one parameter, name.
    public Puppy(String name)
    {
        System.out.println("Passed Name is ": + name);  
    }

    public static void main(String []args)
    {
        // Following statement would create an object myPuppy
        Puppy myPuppy = new Puppy("tommy");
    }
}
```

Output:

3.9 is the largest number.
Output:

```java
// Class Declaration
public class Dog {
    // Instance Variables
    String name;
    String breed;
    int age;
    String color;

    // Constructor Declaration of Class
    public Dog(String name, String breed, int age, String color) {
        this.name = name;
        this.breed = breed;
        this.age = age;
        this.color = color;
    }

    // method 1
    public String getName() {
        return name;
    }
}
```

7. Write a program to demonstrate initialization of an object.

```java
// Class Declaration
public class Dog {

    // Instance Variables
    String name;
    String breed;
    int age;
    String color;

    // Constructor Declaration of Class
    public Dog(String name, String breed, int age, String color) {
        this.name = name;
        this.breed = breed;
        this.age = age;
        this.color = color;
    }

    // method 1
    public String getName() {
        return name;
    }
}
```
// method 2
public String getBreed()
{
    return breed;
}

// method 3
public int getAge()
{
    return age;
}

// method 4
public String getColor()
{
    return color;
}

@Override
public String toString()
{
    return("Hi my name is "+ this.getName()+
            "My breed, age and color are "+
            this.getBreed()+","+ this.getAge()+
            ","+ this.getColor());
}

public static void main(String[] args)
{
    Dog tuffy = new Dog("tuffy","papillon", 5,
    "white");
    System.out.println(tuffy.toString());
}
Output:

```
Hi my name is tuffy.
My breed, age, and color are papillon, 5, white
```

8. Write a program to declare and initialize an array.

```java
//main class
public class Testarray
{
    public static void main(String args[])
    {
        //declaration of array
        int a[]=new int[5];

        //initialization of an array
        a[0]=10;
        a[1]=20;
        a[2]=70;
        a[3]=40;
        a[4]=50;

        System.out.println("Array values are \n");
        //traversing array
        //length is the property of array
        for (int i=0;i<a.length;i++)
        {
            System.out.println (a[i]);
        }
    }
}
```
### Try yourself:
1. Write a java program to calculate the median of a given unsorted array of integers.
2. Write a java program to find a number that appears only once in a given array of integers.
3. Write a program to print the largest number in an array.

9. Write a program for accessing java array elements using for loop.

```java
//main class
public class GFG {
    public static void main (String[] args) {
        // declares an Array of integers.
        int[] arr;

        // allocating memory for 5 integers.
        arr = new int[5];

        // initialize the first elements of the array
        arr[0] = 10;

        // initialize the second elements of the array
        arr[1] = 20;

        //so on...
    }
}
```
arr[2] = 30;
arr[3] = 40;
arr[4] = 50;

// accessing the elements of the specified array
for (int i = 0; i<arr.length; i++)
    System.out.println("Element at index " + i + " : " + arr[i]);

Output:

Element at index 0 : 30
Element at index 1 : 20
Element at index 2 : 50
Element at index 3 : 40
Element at index 4 : 50

10. Write a program to add two matrices.

import java.util.Scanner;

class AddTwoMatrix
{
    public static void main(String args[])
    {
        int m, n, c, d;
        Scanner in = new Scanner (System.in);

        System.out.println ("Enter the number of rows and columns of matrix");
        m = in.nextInt ();
        n =in.nextInt ();

int first[][] = new int[m][n];
int second[][] = new int[m][n];
int sum[][] = new int[m][n];

System.out.println("Enter the elements of first matrix");
for (c = 0; c < m; c++)
  for (d = 0; d < n; d++)
    first[c][d] = in.nextInt();

System.out.println("Enter the elements of second matrix");
for (c = 0; c < m; c++)
  for (d = 0; d < n; d++)
    second[c][d] = in.nextInt();

for (c = 0; c < m; c++)
  for (d = 0; d < n; d++)
    sum[c][d] = first[c][d] + second[c][d];  //replace '+' with '-' to subtract matrices

System.out.println("Sum of the matrices:");
for (c = 0; c < m; c++)
  |
    for (d = 0; d < n; d++)
      System.out.print(sum[c][d]+"\t");
  |
System.out.println();
|
11. Write a program to subtract two matrices.

```java
import java.util.Scanner;

public class MatrixSubtraction
{
    public static void main(String[] args)
    {
        Scanner s = new Scanner(System.in);
        System.out.println("Enter the number of rows");
        int rows = s.nextInt();
        System.out.println("Enter the number of columns");
        int columns = s.nextInt();
    
        int[][] matrix1 = new int[rows][columns];
        int[][] matrix2 = new int[rows][columns];
        int[][] result = new int[rows][columns];

        System.out.println("Enter the elements of first matrix");
        for (int i = 0; i < rows; i++)
        {
            for (int j = 0; j < columns; j++)
            {
                matrix1[i][j] = s.nextInt();
            }
        }

        System.out.println("Enter the elements of second matrix");
        for (int i = 0; i < rows; i++)
        {
            for (int j = 0; j < columns; j++)
            {
                matrix2[i][j] = s.nextInt();
            }
        }

        for (int i = 0; i < rows; i++)
        {
            for (int j = 0; j < columns; j++)
            {
                result[i][j] = matrix1[i][j] - matrix2[i][j];
            }
        }

        System.out.println("Sum of the matrices:");
        for (int i = 0; i < rows; i++)
        {
            for (int j = 0; j < columns; j++)
            {
                System.out.print(result[i][j] + "");
            }
            System.out.println();
        }
    }
}
```
int matrix1[][] = new int[rows][columns];
int matrix2[][] = new int[rows][columns];
int sub[][] = new int[rows][columns];

System.out.println("Enter the elements of first matrix : ");

for (int i = 0; i < rows; i++) {
    for (int j = 0; j < columns; j++) {
        matrix1[i][j] = s.nextInt();
    }
}

System.out.println("Enter the elements of second matrix : ");

for (int i = 0; i < rows; i++) {
    for (int j = 0; j < columns; j++) {
        matrix2[i][j] = s.nextInt();
    }
}

for (int i = 0; i < rows; i++) {
    for (int j = 0; j < columns; j++) {
        sub[i][j] = matrix1[i][j] - matrix2[i][j];
    }
}

System.out.println("The subtraction of the two matrices is : ");

for (int i = 0; i < rows; i++) {
    for (int j = 0; j < columns; j++) {
        
    }
}
System.out.print ("\t" + sub[i][j]);
}
System.out.println (i);
} 
s.close();
}
}

Output:

12. Write a program to multiply two matrices.

```java
import java.util.Scanner;

public class MatrixMultiplication {

```
public static void main(String args[]) {
    int m, n, p, q, sum = 0, i, j, k;

    Scanner in = new Scanner(System.in);
    System.out.println("Enter the number of rows and columns of first matrix");
    m = in.nextInt();
    n = in.nextInt();

    int first[][] = new int[m][n];
    System.out.println("Enter elements of first matrix");
    for (i = 0; i < m; i++)
        for (j = 0; j < n; j++)
            first[i][j] = in.nextInt();

    System.out.println("Enter the number of rows and columns of second matrix");
    p = in.nextInt();
    q = in.nextInt();

    if (n != p)
        System.out.println("The matrices can’t be multiplied with each other.");
    else
        {
        int second[][] = new int[p][q];
        int multiply[][] = new int[m][q];
        }
System.out.println("Enter elements of second matrix");

for (i = 0; i < p; i++)
    for (j = 0; j < q; j++)
        second [i][j] = in.nextInt();

for (i = 0; i < m; i++)
    for (j = 0; j < q; j++)
    {
        for (k = 0; k < p; k++)
        {
            sum = sum + first[i][k]*second[k][j];
        }
        multiply [i][j] = sum;
        sum = 0;
    }

System.out.println ("Product of the matrices: ");

for (i = 0; i < m; i++)
    {
        for (j = 0; j < q; j++)
            System.out.print(multiply[i][j] +"\t");

    System.out.print ("\n");
    }
}
13. Write a program to print transpose of a matrix.

```java
import java.util.Scanner;
public class Transpose {
    public static void main(String args[]) {
        int i, j;
        System.out.println("Enter total rows and columns: ");
        Scanner s = new Scanner (System.in);
        int row = s.nextInt();
        int column = s.nextInt();
        int array[][] = new int[row][column];
        System.out.println ("Enter matrix: ");
```
for (i = 0; i < row; i++)
{
    for (j = 0; j < column; j++)
    {
        array[i][j] = s.nextInt();
        System.out.print(" ");
    }
    System.out.println("The above matrix before Transpose is ");
    for (i = 0; i < row; i++)
    {
        for (j = 0; j < column; j++)
        {
            System.out.println(array[i][j] + " ");
        }
    }
    System.out.println("The above matrix after Transpose is ");
    for (i = 0; i < column; i++)
    {
        for (j = 0; j < row; j++)
        {
            System.out.println(array[j][i] + " ");
        }
    }
}
NOTES

Output:

```
Enter total rows and columns: 
3
3
Enter matrix:
1 2 3
4 5 6
7 8 9
The above matrix before Transpose is
1 2 3
4 5 6
7 8 9
The above matrix after Transpose is
1 4 7
2 5 8
3 6 9
```

Try yourself:
1. Write a program to print sum of diagonal values of a square Matrix.
2. Write a program to find highest and lowest element of a Matrix.
3. Write a java program that searches a value in an m x n matrix.
4. Write a program to calculate area of a circle, a rectangle or a triangle depending on input using overloaded calculate function.

14. Write a program to check an integer is less than 20 using if.

```java
//main class
public class Test 
{
    public static void main(String args[]) 
    {
        
    }
```
```java
int x = 10;

if( x< 20 )
{
    System.out.print("Value of x is less than 20");
}

Output:

Value of x is less than 20

15. Write a program to check an integer is less than 20 using if and else and print suitable message.

   //main class
   public class Test
   {

       public static void main(String args[])
       {
           int x = 30;

           if( x< 20 )
           {
               System.out.print("Value of x is less than 20");
           }
           else
           {
               System.out.print("Value of x is greater than 20");
           }
       }
   }
```
16. Write a program to compare value of an integer variable using nested if and else if.

```java
//main class
class Test {
    public static void main(String args[]) {
        int x = 30;
        if (x == 10) {
            System.out.print("Value of X is 10");
        } else if (x == 20) {
            System.out.print("Value of X is 20");
        } else if (x == 30) {
            System.out.print("Value of X is 30");
        }
    }
}
```

Output: Value of x is greater than 20
else
{
    System.out.print("This is else statement");
}
}

Output:

Value of X is 30

17. Write a program to compare value of an integer variable using nested if.

    //main class
    public class Test
    {

        public static void main(String args[])
        {
            int x = 30;
            int y = 10;

            if( x == 30 )
            {
                if( y == 10 )
                {
                    System.out.print("X = 30 and Y = 10");
                }
            }
        }
    }
Try yourself:
1. Write a Program to convert a lowercase alphabet to uppercase or vice-versa
2. Write a Program to Check whether a year is Leap year or not
3. Write a Program to check whether a given character is uppercase or lowercase alphabet or a digit or a special character

18. Write a program to print grade of students using switch statements.

```java
//main class
public class Test {

    public static void main(String args[]) {
        // char grade = args[0].charAt(0);
        char grade = 'C';
        switch (grade) {
            case 'A' :
                System.out.println("Excellent!");
                break;
            case 'B' :
            case 'C' :
                { 
```
System.out.println("Well done");
    break;
}
case 'D' :
    {
    System.out.println("You passed");
    break;
    }
case 'F' :
    {
    System.out.println("Better try again");
    break;
    }
default :
    {
    System.out.println("Invalid grade");
    break;
    }
}
System.out.println("Your grade is " + grade);
}

Output:

Well done
Your grade is C

19. Write a program to print numbers from 10 to 20 using while loop.

//main class
public class Test {

20. Write a program to print numbers from 10 to 20 using for loop.

```java
public class Test {

    public static void main(String args[]) {

        System.out.print("value of x : ");
        System.out.print("\n");
    }
}
```

Output:

```
value of x : 10
value of x : 11
value of x : 12
value of x : 13
value of x : 14
value of x : 15
value of x : 16
value of x : 17
value of x : 18
value of x : 19
value of x : 20
```
for(int x = 10; x < 20; x = x + 1)
{
    System.out.print("value of x : "+ x);
    System.out.print("\n");
}
}

Output:

value of x : 10
value of x : 11
value of x : 12
value of x : 13
value of x : 14
value of x : 15
value of x : 16
value of x : 17
value of x : 18
value of x : 19
value of x : 20

21. Write a program to print numbers from 10 to 20 using do-while loop.

    public class Test
    {
        public static void main(String args[])
        {
            int x = 10;
            do
            {
                System.out.print("value of x : "+ x);
                x++;
            }
System.out.print("\n");
}
while( x < 20 );
}
}

Output:

```
value of x : 10
value of x : 11
value of x : 12
value of x : 13
value of x : 14
value of x : 15
value of x : 16
value of x : 17
value of x : 18
value of x : 19
value of x : 20
```

22. Write a program to print sum of array using for loop.

```java
public class SumArrayWithForLoop {
    public static void main(String[] args) {
        // array to sum
        int[] numbers = new int[]{ 10, 10, 10, 10};

        int sum = 0;
        for (int i=0; i<numbers.length ; i++) {
            System.out.print(numbers[i] + " ");
            sum += numbers[i];
        }
        System.out.print(" \n");
        System.out.println("sum is:");
        System.out.println(sum);
    }
}
```
\[
\text{sum} = \text{sum} + \text{numbers}[i];
\]

```java
System.out.println("Sum value of array elements is : " + sum);
```

Output:

```
Sum value of array elements is : 40
```

Try yourself:
1. Write a program to reverse a number.
2. Write a program to check whether a number is prime number or not.
3. Write a program to convert binary number to decimal number.
4. Write a program to print table of any number using do while loop.
5. Write a program to print Fibonacci Series (0, 1, 1, 2, 3, 5, 8, 13, 21, ...).
6. Write a program to print Table of any Number using for loop.

23. Write a program to illustrate the concept of function overloading.

```java
class MyClass
{
    int height;
```
MyClass ()
{
    System.out.println("bricks");
    height = 0;
}
MyClass(int i)
{
    System.out.println("Building new House that is " + i + " feet tall");
    height = i;
}
    void info()
{
    System.out.println("House is " + height + " feet tall");
}
    void info(String s)
{
    System.out.println(s + ": House is " + height + "feet tall");
}

public class MainClass
{
    public static void main(String[] args)
    {
    MyClass t = new MyClass(0);
    t.info();
    t.info("overloaded method");
    //Overloaded constructor:
    new MyClass();
    }
}
24. Write a program to overload a sum function.

```java
public class Calculation {
    void sum(int a, int b) {
        System.out.println(a+b);
    }
    void sum(int a, int b, int c) {
        System.out.println(a+b+c);
    }

    public static void main(String args[]) {
        Calculation cal = new Calculation();
        cal.sum(20, 30, 60);
        cal.sum(20, 20);
    }
}
```
25. Write a program to overload display function.

class DisplayOverloading
{
    public void disp(char c)
    {
        System.out.println(c);
    }
    public void disp(char c, int num)
    {
        System.out.println(c + " "+num);
    }
}
public class Sample
{
    public static void main(String args[])
    {
        DisplayOverloading obj = new DisplayOverloading();
        obj.disp('a');
        obj.disp('a',10);
    }
}
Output:

Try yourself:
1. Write a program that overloads comparison function where one function will compare integer values and another comparison function will compare float values.
2. Write a program to concatenate two strings and two characters using overloaded function.

26. Write a program demonstrate single level inheritance in java.

```java
class Animal
{
    void eat()
    {
        System.out.println("eating...");
    }
}
class Dog extends Animal
{
    void bark()
    {
        System.out.println("barking...");
    }
}
public class TestInheritance
{
    public static void main(String args[])
    {
NOTES

Dog d=new Dog();
d.bark();
d.eat();
}

Output:

barking...
eating...

27. Write a program demonstrate multiple inheritance in java.

class Animal
{
    void eat()
    {
        System.out.println("eating...");
    }
}
class Dog extends Animal
{
    void bark()
    {
        System.out.println("barking...");
    }
}
class BabyDog extends Dog
{
public class TestInheritance2 {
  public static void main(String args[]) {
    BabyDog d = new BabyDog();
    d.weep();
    d.bark();
    d.eat();
  }
}

Output:
weeping...
barking...
eating...

28. Write a program demonstrate hierarchical inheritance in java.

class Animal
{
  void eat()
  {
    System.out.println("eating...");
  }
}
class Dog extends Animal
{
    void bark()
    {
        System.out.println("barking...");
    }
}

class Cat extends Animal
{
    void meow()
    {
        System.out.println("meowing...");
    }
}

public class TestInheritance3
{
    public static void main(String args[])
    {
        Cat c=new Cat();
        c.meow();
        c.eat();
        //c.bark(); // C.T. Error
    }
}

Output:

weeping...
barking...
eating...
Try yourself:
1. Write a program to get and print student data using inheritance.
2. What will be the output of this program?
   ```java
class A {
    int i = 10;
}

class B extends A {
    int i = 20;
}

class MainClass {
    public static void main(String[] args) {
        A a = new B();
        System.out.println(a.i);
    }
}
```
3. What will be the output of this program?
   ```java
class A {
    {
        System.out.println(1);
    }
}
class B extends A {
    {
        System.out.println(2);
    }
```
class C extends B
{
    System.out.println(3);
}

public class MainClass
{
    public static void main(String[] args)
    {
        C c = new C();
    }
}

29. Write a program to demonstrate the concept of runtime polymorphism.

Animal.java
public class Animal
{
    public void sound()
    {
        System.out.println("Animal is making a sound");
    }
}

Horse.java
class Horse extends Animal
{
    @Override
    public void sound()
    {
        System.out.println("Neigh");
    }
}
public static void main(String args[]) { 
    Animal obj = new Horse();
    obj.sound();
}

Cat.java
public class Cat extends Animal 
{
    @Override
    public void sound() 
    { 
        System.out.println("Meow");
    }
    public static void main(String args[]) 
    { 
        Animal obj = new Cat();
        obj.sound();
    }
}

30. Write a program to demonstrate method overloading during runtime.

class Overload 
{
    void demo (int a) 
    {
        System.out.println ("a: "+ a);
    }
    void demo (int a, int b) 
    {
        System.out.println ("a and b: " + a + "," + b);
    }
    double demo (double a) 
    { 

```java
public class MethodOverloading {
    public static void main (String args [])
    {
        Overload Obj = new Overload();
        double result;
        Obj .demo(10);
        Obj .demo(10, 20);
        result = Obj .demo(5.5);
        System.out.println("O/P : "+ result);
    }
}
```

Output:
```
a: 10
a and b: 10,20
double a: 5,5
O/P : 30.25
```

31. Write a program to create and run a thread.

```java
class RunnableDemo implements Runnable
{
    private Thread t;
    private String threadName;

    RunnableDemo( String name)
    {
        threadName = name;
```
System.out.println("Creating " + threadName);
}

public void run()
{
    System.out.println("Running " + threadName);
    try
    {
        for(int i = 4; i > 0; i--)
        {
            System.out.println("Thread: " + threadName + ", " + i);
            // Let the thread sleep for a while.
            Thread.sleep(50);
        }
    }
    catch (InterruptedException e)
    {
        System.out.println("Thread " + threadName + " interrupted.");
    }
    System.out.println("Thread " + threadName + " exiting.");
}

public void start()
{
    System.out.println("Starting " + threadName);
    if(t == null)
    {
        t = new Thread(this, threadName);
        t.start();
    }
}

public class TestThread
{
public static void main(String args[]) {
    RunnableDemo R1 = new RunnableDemo( "Thread-1");
    R1.start();
    RunnableDemo R2 = new RunnableDemo( "Thread-2");
    R2.start();
}

Output:

Creating Thread-1
Starting Thread-1
Creating Thread-2
Starting Thread-2
Running Thread-1
Thread: Thread-1, 4
Running Thread-2
Thread: Thread-2, 4
Thread: Thread-1, 3
Thread: Thread-2, 3
Thread: Thread-1, 2
Thread: Thread-1, 2
Thread: Thread-2, 2
Thread: Thread-1, 1
Thread: Thread-2, 1
Thread-1 exiting.
Thread-2 exiting.

32. Write a program to create a thread by extending the thread class.

    // Java code for thread creation by extending
    // the Thread class
    class MultithreadingDemo extends Thread
    {
        public void run()
        {
            try
            {
                // Displaying the thread that is running
                System.out.println ("Thread" + Thread.currentThread().getId () + "is running");
            }
        }
    }

Self-Instructional Material
```java
// Throwing an exception
System.out.println("Exception is caught");
}
}
// Main Class
public class Multithread {
    public static void main(String[] args) {
        int n = 8; // Number of threads
        for (int i=0; i<n; i++) {
            MultithreadingDemo object = new MultithreadingDemo();
            object.start();
        }
    }
}
```

Output:

```
Thread 10 is running
Thread 11 is running
Thread 12 is running
Thread 13 is running
Thread 14 is running
Thread 15 is running
Thread 16 is running
Thread 17 is running
```
33. Write a program to create a thread by implementing the runnable interface.

```java
// Java code for thread creation by implementing
// the Runnable Interface
class MultithreadingDemo implements Runnable {
    public void run()
    {
        try
        {
            // Displaying the thread that is running
            System.out.println("Thread " + Thread.currentThread().getId() + " is running");
        }
        catch (Exception e)
        {
            // Throwing an exception
            System.out.println("Exception is caught");
        }
    }
}

// Main Class
public class Multithread {
    public static void main(String[] args)
    {
        int n = 8; // Number of threads
        for (int i=0; i<n; i++)
        {
            Thread object = new Thread(new MultithreadingDemo());
            object.start();
        }
    }
}```
Output:

```
Thread 10 is running
Thread 12 is running
Thread 11 is running
Thread 14 is running
Thread 15 is running
Thread 13 is running
Thread 17 is running
Thread 16 is running
```

34. Write a program to demonstrate the concept of abstract classes.

```java
abstract class Bank
{
    abstract int getRateOfInterest();
}
class SBI extends Bank
{
    int getRateOfInterest(){return 7;
}
}
class PNB extends Bank
{
    int getRateOfInterest(){return 8;
}
}

// Main Class
public class TestBank
{
    public static void main(String args[])
    {
        Bank b;
        b=new SBI();
        System.out.println("Rate of Interest is:");
```
"+b.getRateOfInterest()+" %");
b=new PNB();
System.out.println("Rate of Interest is:
"+b.getRateOfInterest()+" ");
}
}
}

Output:

```
Rate of Interest is: 7 %
Rate of Interest is: 8 %
```

35. Write a program to create an abstract class having constructor, data member and methods.

```java
abstract class Bike
{
Bike()
{
System.out.println("bike is created");
}
abstract void run();
void changeGear()
{
System.out.println("gear changed");
}

//Creating a Child class which inherits Abstract class
class Honda extends Bike
{
void run()
{
System.out.println("running safely..");
```
//Creating a Test class which calls abstract and non-abstract methods
public class TestAbstraction2
{
    public static void main(String args[])
    {
        Bike obj = new Honda();
        obj.run();
        obj.changeGear();
    }
}

Output:

bike is created
running safely..
gear changed

36. Write a program having try-catch block for exception handling.

// Main Class
public class TestTrycatch2
{
    public static void main(String args[])
    {
        try
        {
            int data=50/0;
        }
        catch (ArithmeticException e)
        {
            System.out.println(e.getMessage());
        }
    }
}
37. Write a program having multiple try-catch block for exception handling.

```java
class TestMultipleCatchBlock {
    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");
        }
    }
}
```

Output:
```
java.lang.ArithmeticException: / by zero
rest of the code...
```
catch(Exception e) {
  System.out.println("common task completed");
}
System.out.println("rest of the code..."};
}

Output:

task1 is completed
rest of the code...

38. Write a program having nested try-catch block for exception handling.

class Excep6 {
  public static void main(String args[]) {
    try {
      try {
        System.out.println("going to divide");
        int b = 39 / 0;
      }
      catch(ArithmeticException e) {System.out.println(e); }
    }
    catch(ArithmeticException e) {System.out.println(e); }
    try {  
      
                  

int a[]=new int[5];
a[5]=4;

NOTES

39. Write a program having try-catch with finally block.

public class TestFinallyBlock2
{
    public static void main(String args[])
    {
        try
        {
            int data=25/0;
            System.out.println(data);
        }
        catch(ArithmeticException e)
        {
            System.out.println(e);
        }
        System.out.println("other statement");
        }
        catch(Exception e)
        {
            System.out.println("handed");
        }
        System.out.println("normal flow..");
    }
}
finally
{
    System.out.println("finally block is always executed");
}
System.out.println("rest of the code...");

Output:

java.lang.ArithmeticException: / by zero
finally block is always executed
rest of the code...

40. Write a program to demonstrate the concept of throwing an exception.

public class TestThrow1
{
    static void validate(int age)
    {
        if(age<18)
            throw new ArithmeticException("not valid");
        else
            System.out.println("welcome to vote");
    }
    public static void main(String args[])
    {
        validate(13);
        System.out.println("rest of the code...");
    }
}
Output:

```java
import java.io.*;
// Main Class
public class CopyFile
{
    public static void main(String args[]) throws IOException
    {
        FileInputStream in = null;
        FileOutputStream out = null;
        try
        {
            in = new FileInputStream("input.txt");
            out = new FileOutputStream("output.txt");
            int c;
            while ((c = in.read()) != -1)
            {
                out.write(c);
            }
        }
        finally
        {
            if (in != null)
            {
                in.close();
            }
            if (out != null)
            {
                out.close();
            }
        }
    }
}
```

41. Write a program to copy data of one file to another file.

```java
import java.io.*;
// Main Class
public class CopyFile
{
    public static void main(String args[]) throws IOException
    {
        FileInputStream in = null;
        FileOutputStream out = null;
        try
        {
            in = new FileInputStream("input.txt");
            out = new FileOutputStream("output.txt");
            int c;
            while ((c = in.read()) != -1)
            {
                out.write(c);
            }
        }
        finally
        {
            if (in != null)
            {
                in.close();
            }
            if (out != null)
            {
                out.close();
            }
        }
    }
}
```
import java.io.*;
// Main Class
public class CopyFile {
    public static void main(String args[]) throws IOException {
        FileReader in = null;
        FileWriter out = null;
        try {
            in.close();
            if (out != null)
                out.close();
        }
        catch (Exception e) {
            System.out.println(e.getMessage());
        }
    }
}

Output:

42. Write a program to copy data of one file to another file using character streams.

import java.io.*;
// Main Class
public class CopyFile {
    public static void main(String args[]) throws IOException {
        FileReader in = null;
        FileWriter out = null;
        try {
            in.close();
            if (out != null)
                out.close();
        }
        catch (Exception e) {
            System.out.println(e.getMessage());
        }
    }
}
try {
    in = new FileReader("input.txt");
    out = new FileWriter("output.txt");
    int c;
    while ((c = in.read()) != -1) {
        out.write(c);
    }
} finally {
    if (in != null) {
        in.close();
    }
    if (out != null) {
        out.close();
    }
}

Output:
43. Write a program to write a string in file.

```java
import java.io.FileOutputStream;

public class FileOutputStreamExample {
    public static void main(String args[]) {
        try {
            FileOutputStream fout = new FileOutputStream("D:\testout.txt");
            String s = "Welcome to javaTpoint."

            //converting string into byte array
            byte b[] = s.getBytes();
            fout.write(b);
            fout.close();
            System.out.println("success...");
        } catch (Exception e) {
            System.out.println(e);
        }
    }
}
```

Output:

Success...
NOTES

Try yourself:
1. Write a Java program to read a file content line by line.
2. Write a Java program to read a plain text file.
3. Write a java program to read a file line by line and store it into a variable.

Java AWT (Abstract Window Toolkit) is an API to develop GUI or window-based applications in java. The java.awt package provides classes for AWT API such as TextField, Label, TextArea, RadioButton, CheckBox, Choice, List etc. Following are the steps to create an applet.

Step 1: Write a Java code using text editor (notepad).

44. Write a program to Create applet to print Hello World.

import java.applet.Applet;
import java.awt.Graphics;

// HelloWorld class extends Applet
public class HelloWorld extends Applet
{
    // Overriding paint() method
    @Override
    public void paint(Graphics g)
    {
        g.drawString("Hello World", 20, 20);
    }
}

Step 2: Save the file as HelloWorld.java.

Step 3: Compiling Applets:

Javac HelloWorld.java
Step 4: Running Applet from console:

```
java HelloWorld
```

**Note:** Running `HelloWorld` with the `java` command will generate an error because it is not an application.

```java
java HelloWorld
```

One difference between an application and an Applet is that applications must have a `main()`. Our Applet does not, so we see the error message as:

```
Exception in thread "main" java.lang.NoSuchMethodError:
main
```

We need to create our HTML code for the Hello World Applet:

**Step 1:** Type in the following HTML code.

```html
<html>
<head>
<title>Hello World</title>
</head>
<body>
<p>Hello World Applet</p>
<applet code="HelloWorld.class" width=300 height=200>
</applet>

</body>
</html>
```

**Step 2:** save file as `HelloWorld.html`

**Step 3:** Run the `HelloWorld` Applet with `appletviewer`.

```
appletviewer HelloWorld.html
```
45. Write a program to draw an Arc in Applet Window.

```java
import java.applet.Applet;
import java.awt.Color;
import java.awt.Graphics;

public class DrawArcExample extends Applet
```

Note: You can also run your applet in your browser window. For the URL, type in the path for your HTML file.
public void paint(Graphics g) {
  //set color to red
  setForeground(Color.red);

  //this will draw an arc of width 50 & height 100 at (10,10)
  g.drawArc(10,10,50,100,10,45);

  //draw filled arc
  g.fillArc(100,10,100,100,0,90);
}

Output:
46. Write a program to draw 3D rectangle and square.

```java
import java.applet.Applet;
import java.awt.Color;
import java.awt.Graphics;

public class Draw3DRectanglesExample extends Applet {

    public void paint(Graphics g) {

        g.setColor(Color.green);
        // this will draw a 3-D rectangle of width 50 & height 100 at (10,10)
        g.draw3DRect(10, 10, 50, 100, true);
        // this will draw a 3-D square
        g.draw3DRect(100, 100, 50, 50, true);

        g.setColor(Color.orange);
        g.fill3DRect(10, 150, 50, 100, true);
        // this will draw a filled 3-D square
        g.fill3DRect(100, 200, 50, 50, true);
    }
}
```
Output:

Try yourself:
1. Write a program to create different shapes using applet.
2. Write a program to fill colors in shapes using applet.

47. Write a program for implementation of simple network using java.

Problem 1: Socket Client
import java.net.*;
import java.io.*;

public class GreetingClient
{

    public static void main(String [] args)
    {
        String serverName = args[0];
        int port = Integer.parseInt(args[1]);
        try
        {  

            
        }  
    
}
System.out.println("Connecting to " + serverName + " on port " + port);
Socket client = new Socket(serverName, port);

NOTES

System.out.println("Just connected to " +
client.getRemoteSocketAddress());
OutputStreamoutToServer = client.getOutputStream();
DataOutputStream out = new DataOutputStream(outToServer);
out.writeUTF("Hello from " + client.getLocalSocketAddress());
InputStreaminFromServer = client.getInputStream();
DataInputStream in = new DataInputStream(inFromServer);
System.out.println("Server says " + in.readUTF());
close();
}
catch (IOException e)
{
e.printStackTrace();
}
}

Problem 2: Socket Server
import java.net.*;
import java.io.*;

public class GreetingServer extends Thread
{
    private ServerSocket serverSocket;

    public GreetingServer(int port) throws IOException
    {
        serverSocket = new ServerSocket(port);
        serverSocket.setSoTimeout(10000);
    }
public void run()
{
    while (true)
    {
        try
        {
            System.out.println("Waiting for client on port "+
serverSocket.getLocalPort() + "...");
            Socket server = serverSocket.accept();

            System.out.println("Just connected to "+ server.getRemote
SocketAddress());
            DataInputStream in = new DataInputStream(server.
getInputStream());

            System.out.println(in.readUTF());
            DataOutputStream out = new DataOutputStream(server.
getOutputStream());
            out.writeUTF("Thank you for connecting to "+ server.
getLocalSocketAddress()+ "\nGoodbye!");
            server.close();
        }
        catch (SocketTimeoutException s)
        {
            System.out.println("Socket timed out!");
            break;
        }
        catch (IOException e)
        {
            e.printStackTrace();
            break;
        }
    }
}
```java
public static void main(String[] args) {
    int port = Integer.parseInt(args[0]);
    try {
        Thread t = new GreetingServer(port);
        t.start();
    } catch (IOException e) {
        e.printStackTrace();
    }
}
```

Output:

```
$ java GreetingClient localhost 8080
Connecting to localhost on port 8080.
Just connected to localhost/127.0.0.1:8080
Server says Thank you for connecting to /127.0.0.1:8080
Goodbye!
```
M.Sc. [Information Technology]
313 14
LAB: OBJECT ORIENTED PROGRAMMING AND JAVA
I - Semester

ALAGAPPA UNIVERSITY
[Accredited with 'A+' Grade by NAAC (CGPA:3.64) in the Third Cycle and Graded as Category-I University by MHRD-UGC]
KARAIKUDI – 630 003
DIRECTORATE OF DISTANCE EDUCATION

LAB: OBJECT ORIENTED PROGRAMMING AND JAVA
I - Semester