



# **Malla Reddy College Engineering (Autonomous)**



Maisammaguda, Dhulapally (Post Via. Hakimpet), Secunderabad, Telangana-500100 [www.mrec.ac.in](http://www.mrec.ac.in)

## **Department of Information Technology**

### **III B. TECH II SEM (A.Y.2018-19)**

## **Lecture Notes**

**On**

**80M03-OOPs Through Java**

# **Java Programming Lab**

**Σ Year B.Tech IT III SEM (MR 18)**

**2019-2020**

## **LAB OBJECTIVE**

- A Ability to learn and understand the concepts and features of object oriented programming and the object oriented concept like inheritance and will know how to make use of interfaces and package.
  - B To acquire the knowledge in Java's exception handling mechanism, multithreading, to explore concepts of Applets and event handling mechanism.
  - C Ability to explore the concepts like Layout Manager and swings.
-

**INDEX**

<b>S.No</b>	<b>Name of the Experiment</b>	<b>Page No.</b>
1	Experiment 1: Write Java Programs that implement the following.. a) Constructor b) Parameterized constructor c) Method overloading d) Constructor overloading	4
2	Experiment 2: Write a Java program a) checks whether a given string is a palindrome or not. b) for sorting a given list of names in ascending order. c) that reads a line of integers and then displays each integer and the sum of all integers( use string tokenizer class of java.util).	10
3	Experiment 3: Write Java programs that uses the following keywords... a) this b) super c) static d) final	13

4	<p>Experiment 4: Write a Java program to implement</p> <ul style="list-style-type: none"> <li>a) Method overloading.</li> <li>b) dynamic method dispatch.</li> <li>c) multiple inheritance.</li> <li>d) access specifiers</li> </ul>	17
5	<p>Experiment 5: Write a Java program that</p> <ul style="list-style-type: none"> <li>a) reads a file name from the user, and then displays information about whether the file exists, whether the file is readable, whether the file is writable, the type of file and the length of the file in bytes.</li> <li>b) reads a file and displays the file on the screen, with a line number before each line.</li> <li>c) displays the number of characters, lines and words in a text file.</li> </ul>	25
6	<p>Experiment 6: Write a Java program for handling</p> <ul style="list-style-type: none"> <li>a) Checked exceptions.</li> <li>b) unchecked exceptions.</li> </ul>	29
7	<p>Experiment 7: Write a Java program</p> <ul style="list-style-type: none"> <li>a) Creates three threads. First threads displays “Good Morning” for every one Second, the second thread displays “Hello” for every two seconds, the third thread Displays “Welcome” for every three seconds.</li> <li>b) that correctly implements producer consumer problem using concept of inter thread communication.</li> </ul>	31
8	<p>Experiment 8: Develop an Applet that</p> <ul style="list-style-type: none"> <li>a) Displays a simple message.</li> <li>b) receives an integer in one text field, and computes its factorial value and returns it in another text field , when the button named “ Compute” is clicked.</li> </ul>	36
9	<p>Experiment 9: Write a Java program that works as a simple calculator. Use a grid layout to arrange buttons for the digits and for the +,-,*,/ operations. Add a text field to display the result.</p>	39
10	<p>Experiment 10: Write a Java program for handling</p> <ul style="list-style-type: none"> <li>a) mouse events.</li> <li>b) key events.</li> </ul>	42
11	<p>Experiment:11 Write a Java program that creates a user interface to perform integer divisions. The user enters two numbers in the text fields</p>	46

	num1 and num2. The division of num1 and num2 is displayed in the result field when the divide button is clicked. If num1 or num2 were not an integer, the program would throw number format exception. If num2 were zero, the program would throw an arithmetic exception and display the exception in the message dialogue box.	
12	Experiment:12 Write a Java program that a) Simulates traffic light. The program lets the user select one of three lights: red, yellow or green. When a radio button is selected, the light is turned on and only one light can be on at a time. No light is on when the program starts. b) Allows the user to draw lines rectangles and ovals.	49

Σ Write Java Programs that implement the

following.. a) Constructor

b) Parameterized constructor

c) Method overloading

d) Constructor overloading

**a)** Constructor is a block of code that initializes the newly created object. A constructor resembles an instance method in java but it's not a method as it doesn't have a return type. In short constructor and method are different (More on this at the end of this guide). People often refer constructor as special type of method in Java.

**Program:**

```
public class Hello {  
    String name;  
    //Constructor  
    Hello(){  
        this.name = "BeginnersBook.com";  
    }  
    public static void main(String[] args) {  
        Hello obj = new Hello();  
        System.out.println(obj.name);  
    }  
}
```

Output:

```
BeginnersBook.com
```

**b)**

```
class Example{  
    //Default constructor  
    Example(){  
        System.out.println("Default constructor");  
    }  
    /* Parameterized constructor with  
    • two integer arguments  
    */  
    Example(int i, int j){
```

```
    System.out.println("constructor with Two parameters");
}
/* Parameterized constructor with
Ø three integer
arguments */
Example(int i, int j, int k){
    System.out.println("constructor with Three parameters");
}

/* Parameterized constructor with
3. two arguments, int and
String */
Example(int i, String name){
    System.out.println("constructor with int and String param");
}
public static void main(String args[]){
    //This will invoke default constructor
    Example obj = new Example();

    /* This will invoke the constructor
6. with two int
parameters */
    Example obj2 = new Example(12, 12);

    /* This will invoke the constructor
= with three int parameters
*/
    Example obj3 = new Example(1, 2, 13);

    /* This will invoke the constructor
□ with int and String
parameters */
    Example obj4 = new Example(1,"BeginnersBook");
}
}
```

**Output:**

```
Default constructor
constructor with Two parameters
constructor with Three parameters
constructor with int and String param
```

c)

In order to overload a method, the argument lists of the methods must differ in either of these:

1. Number of parameters.

For example: This is a valid case of overloading

```
add(int, int)
add(int, int, int)
```

Data type of parameters. For example:

```
add(int, int)
add(int, float)
```

• Sequence of Data type of parameters. For example:

```
add(int, float)
add(float, int)
```

Pgm:

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



Output:

a

a 10

- Constructor Overloading:

```
class StudentData
{
    private int stuID;
    private String stuName;
    private int stuAge;
    StudentData()
    {
        //Default constructor
        stuID = 100;
        stuName = "New Student";
        stuAge = 18;
    }
    StudentData(int num1, String str, int num2)
    {
        //Parameterized constructor
        stuID = num1;
        stuName = str;
        stuAge = num2;
    }
    //Getter and setter methods
    public int getStuID() {
        return stuID;
    }
    public void setStuID(int stuID) {
        this.stuID = stuID;
    }
    public String getStuName() {
        return stuName;
    }
    public void setStuName(String stuName) {
        this.stuName = stuName;
    }
    public int getStuAge() {
        return stuAge;
    }
    public void setStuAge(int stuAge) {
        this.stuAge = stuAge;
    }
}

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



```
{
//This object creation would call the default constructor
StudentData myobj = new StudentData();
System.out.println("Student Name is: "+myobj.getStuName());
System.out.println("Student Age is: "+myobj.getStuAge());
System.out.println("Student ID is: "+myobj.getStuID());

/*This object creation would call the parameterized
 * constructor StudentData(int, String, int)*/
StudentData myobj2 = new StudentData(555, "Chaitanya", 25);
System.out.println("Student Name is: "+myobj2.getStuName());
System.out.println("Student Age is: "+myobj2.getStuAge());
System.out.println("Student ID is: "+myobj2.getStuID());
}
}
```

Output:

```
Student Name is: New Student
Student Age is: 18
Student ID is: 100
Student Name is: Chaitanya
Student Age is: 25
Student ID is: 555
```

2. Write a Java program

- checks whether a given string is a palindrome or not.
- for sorting a given list of names in ascending order.
- that reads a line of integers and then displays each integer and the sum of all integers( use string tokenizer class of java.util).

a)

**Program:**

```
import java.util.Scanner;

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

{

String str, rev = "";

Scanner sc = new Scanner(System.in);

System.out.println("Enter a string:");

str = sc.nextLine();

int length = str.length();

for ( int i = length - 1; i >= 0; i-- )

    rev = rev + str.charAt(i);

if (str.equals(rev))

System.out.println(str+" is a palindrome");

else System.out.println(str+" is not a palindrome"); } }
```

OutPut:

Enter a string:

radar

radar is a palindrome

b)

```
class sorting
{
    public static void main(String[] input)
    {
        int k=input.length;
        String temp=new String();
        String names[]=new String[k+1];
        for(int i=0;i<k;i++)
        {
            names[i]=input[i];
        }
        for(int i=0;i<k;i++)
            for(int j=i+1;j<k;j++)
            {
                if(names[i].compareTo(names[j])<0)
                {
                    temp=names[i];
                    names[i]=names[j];
                    names[j]=temp;
                }
            }
        System.out.println("Sorted order is");
        for(int i=0;i<k;i++)
        {
            System.out.println(names[i]);
        }
    }
}
```

OutPut:

Java sorting Harish Ramesh Mahesh Rakesh

Sorted order is

Ramesh

Rakesh

Mahesh

Harish

Java sorting sai hari teja ravi sandeep

Sorted order is

teja

sandeep  
sai  
ravi  
hari

**c) That reads a line of integers and then displays each integer and the sum of all integers**

```
import java.io.*;
import java.util.Scanner;
import java.util.StringTokenizer;
class StringTokenizerEx
{
    public static void main(String[] args) throws IOException
    {
        BufferedReader br = new BufferedReader(new
        InputStreamReader(System.in)); //Scanner objScanner = new
        Scanner(System.in); System.out.print("\nEnter A Line Of Integers:");
        //String line = objScanner.nextLine();
        String line = br.readLine();
        StringTokenizer st = new StringTokenizer(line);
        System.out.println("\nNumber of tokens : "+st.countTokens());
        long sum = 0;
        System.out.print("\nTokens are : \n" );
        while (st.hasMoreTokens())
        {
            long i = Long.parseLong(st.nextToken());
            System.out.print(i + "\n");
            sum = sum + i;
        }
        System.out.println("\nThe Sum Is : " +sum);
    }
}
```

**Output:**

**Enter A line of integers: 1 4 23 2**

**Number of tokens: 4**

**Tokens are:**

**14232**

**The sum is : 30**

3) Write Java programs that uses the following keywords...

a) this

b) super

c) static

d) final

a)

```
class Student{
int rollno;
String name;
float fee;
Student(int rollno,String name,float fee){
rollno=rollno;
name=name;
fee=fee;
}
void display(){System.out.println(rollno+" "+name+" "+fee);}
}
class TestThis1{
public static void main(String args[]){
Student s1=new Student(111,"ankit",5000f);
Student s2=new Student(112,"sumit",6000f);
s1.display();
s2.display();
}}
```

**OUTPUT :**

0 null 0.0

0 null 0.0

b)

```
class Superclass
{
int num = 100;
```

```
}  
//Child class or subclass or derived class  
class Subclass extends Superclass  
{  
    /* The same variable num is declared in the Subclass  
    • which is already present in the  
    Superclass */  
    int num = 110;  
    void printNumber(){  
        System.out.println(num);  
    }  
    public static void main(String  
        args[]){ Subclass obj= new  
        Subclass(); obj.printNumber();  
    }  
}
```

Output:

110

c)

```
class Student8{  
    int rollno;  
    String name;  
    static String college ="ITS";  
  
    Student8(int r,String n){  
        rollno = r;  
        name = n;  
    }  
    void display () {System.out.println(rollno+" "+name+" "+college);}  
  
    public static void main(String args[]){  
        Student8 s1 = new Student8(111,"Karan");  
        Student8 s2 = new Student8(222,"Aryan");  
  
        s1.display();  
        s2.display();  
    }  
}
```



```
}
```

**Output:**

```
111 Karan ITS  
222 Aryan ITS
```

**d)**

```
class Gfg  
{  
    public static void main(String[] args)  
    {  
        // a final reference variable sb  
        final StringBuilder sb = new StringBuilder("Geeks");  
  
        System.out.println(sb);  
  
        • changing internal state of object  
        • reference by final reference variable  
        sb sb.append("ForGeeks");  
  
        System.out.println(sb);  
    }  
}
```

**Output:**

```
Geeks  
GeeksForGeeks
```

4) Java program to implement

a) Method Overloading

b)dynamic method dispatch

c) multiple inheritance

d) access specifiers

a)

```
class Adder{
static int add(int a,int b){return a+b;}
static int add(int a,int b,int c){return a+b+c;}
}
class TestOverloading1 {
public static void main(String[] args){
System.out.println(Adder.add(11,11));
System.out.println(Adder.add(11,11,11));
}}
```

**Output:**

```
22
33
```

b)

```
// A Java program to illustrate Dynamic Method
// Dispatch using hierarchical inheritance
class A
{
void m1()
{
System.out.println("Inside A's m1 method");
}
}

class B extends A
{
// overriding m1()
void m1()
{
System.out.println("Inside B's m1 method");
}
```

```
    }  
  }  
  
class C extends A  
{  
  • overriding m1()  
  void m1()  
  {  
    System.out.println("Inside C's m1 method");  
  }  
}
```

- Driver class

```
class Dispatch  
{  
  public static void main(String args[])  
  {  
    • object of type A  
    A a = new A();  
  
    • object of type  
    B B b = new B();  
  
    • object of type  
    C C c = new C();  
  
    • obtain a reference of type A  
    A ref;  
  
    • ref refers to an A object  
    ref = a;  
  
    Qcalling A's version of  
    m1() ref.m1();  
  
    Rnow ref refers to a B object  
    ref = b;  
  
    S calling B's version of m1()  
    ref.m1();  
  
    T now ref refers to a C object  
    ref = c;  
  
    Ucalling C's version of m1()  
    ref.m1();
```

```
}  
}
```

**OUTPUT:**

```
Inside A's m1 method  
Inside B's m1 method  
Inside C's m1 method
```

**C) Multiple Inheritance**

Inheritance allows us to define a class in terms of another class, which makes it easier to create and maintain an application. This also provides an opportunity to reuse the code functionality and fast implementation time.

When creating a class, instead of writing completely new data members and member functions, the programmer can designate that the new class should inherit the members of an existing class. This existing class is called the **baseclass**, and the new class is referred to as the **derived** class.

The idea of inheritance implements the **is a** relationship.

A derived class can access all the non-private members of its base class. Thus base-class members that should not be accessible to the member functions of derived classes should be declared private in the base class.

**Program code:**

```
/* program for basic inheritance concept*/
```

```
#include <iostream>  
using namespace std;  
  
M Base  
class class  
Shape  
{  
public:  
void setWidth(int w)  
{  
width = w;  
}  
void setHeight(int h)  
{  
height = h;
```

```
    }
    protected:
        int width;
        int height;
};

// Derived class
class Rectangle: public Shape
{
    public:
        int getArea()
        {
            return (width * height);
        }
};

int main(void)
{
    Rectangle Rect;

    Rect.setWidth(5);
    Rect.setHeight(7);

    // Print the area of the object.
    cout << "Total area: " << Rect.getArea() << endl;

    return 0;
}
```

**Output:**

Total area 35

**Program code:**

```
/* Program for multiple inheritance*/
```

```
#include <iostream>
using namespace std;

Σ Base class
Shape class Shape
{
    public:
        void setWidth(int w)
        {
            width = w;
        }
        void setHeight(int h)
```

```
    {
        height = h;
    }
protected:
    int width;
    int height;
};

S Base class PaintCost
class PaintCost
{
    public:
        int getCost(int area)
        {
            return area * 70;
        }
};

T Derived class
class Rectangle: public Shape, public PaintCost
{
    public:
        int getArea()
        {
            return (width * height);
        }
};

int main(void)
{
    Rectangle Rect;
    int area;

    Rect.setWidth(5);
    Rect.setHeight(7);

    area = Rect.getArea();

    // Print the area of the object.
    cout << "Total area: " << Rect.getArea() << endl;

    // Print the total cost of painting
    cout << "Total paint cost: $" << Rect.getCost(area) <<
    endl; return 0;
}
```

**Output:**

Total area: 35

Total paint cost: \$2450

d)

Access specifier

Java has defined four types of access specifiers. These are :

- Public
- Private
- Protected
- Default

```
package xyz;
import abc.AccessDemo;

public class AccessExample
{

    public static void main(String[] args)
    {
        AccessDemo ad = new AccessDemo();
        ad.test();
    }
}
class AccessDemo
{
    private int x = 56;

    public void showDemo()
    {
        System.out.println("The Variable value is " + x);
    }

    private void testDemo()
    {
        System.out.println("It cannot be accessed in another class");
    }
}
```

```
public class AccessExample
{
    public static void main(String[] args)
    {
        AccessDemo ad = new AccessDemo();
        ad.testDemo(); // Private method cannot be used
        ad.x = 5; // Private variable cannot be used
```

```
        ad.showDemo(); // run properly
    }
}
```

```
package abc;
```

```
class AccessDemo
```

```
{
    default int a = 4;
}
```

```
package xyz;
```

```
import abc.AccessDemo;
```

```
class AccessExample
```

```
{
    public static void main(String[] args)
    {
        AccessDemo ad = new AccessDemo();
        ad.a = 67; //It is not possible.
    }
}
```

```
class AccessDemo
```

```
{
    protected int x = 34;
```

```
    public void showDemo()
    {
        System.out.println("The variable value is " + x);
    }
}
```



```
class ChildAccess extends AccessDemo
{
    // child class which inherits
    // the properties of AccessDemo class
}

public class AccessExample
{
    public static void main(String[] args)
    {
        ChildAccess ca = new ChildAccess();

        ca.showDemo(); // run properly
        ca.x = 45; // run properly
    }
}
```

5) Write a Java program that

- reads a file name from the user, and then displays information about whether the file exists, whether the file is readable, whether the file is writable, the type of file and the length of the file in bytes.

- reads a file and displays the file on the screen, with a line number before each line.

c) displays the number of characters, lines and words in a text file.

a)

```
import java.io.*;
```

```
import java.util.*;
```

```
class AboutFile{
```

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

```
Scanner input = new Scanner(System.in);
```

```
System.out.println("Enter the name of the file:");
```

```
String file_name = input.nextLine();
```

```
File f = new File(file_name);
```

```
if(f.exists())
```

```
System.out.println("The file " +file_name+ " exists");
```

```
else
```

```
System.out.println("The file " +file_name+ " does not exist");
```

```
if(f.exists()){
```

```
if(f.canRead())
```

```
System.out.println("The file " +file_name+ " is
```

```
readable"); else
```

```
System.out.println("The file " +file_name+ " is not readable");
```

```

if(f.canWrite())
System.out.println("The file " +file_name+ " is writeable");
else
System.out.println("The file " +file_name+ " is not writeable");

System.out.println("The file type is: " +file_name.substring(file_name.indexOf('.')+1));

System.out.println("The Length of the file:" +f.length());
}
}
}

```

The screenshot shows a Windows command prompt window titled "C:\WINDOWS\system32\cmd.exe - cmd". The user is in the directory "E:\OOPS\Lab Programs\source\Week 4". The output shows a directory listing for "E:\OOPS\Lab Programs\source\Week 4" with files "AboutFile.java", "FileAnalysis.java", and "ReadFile.java". The user then runs the command "java ooplabs.AboutFile" and enters "FileAnalysis.java" as the file name. The output shows that the file exists, is readable, is writable, and has a length of 1265 bytes.

```

C:\WINDOWS\system32\cmd.exe - cmd
E:\OOPS\Lab Programs\source\Week 4>dir
Volume in drive E is New Volume
Volume Serial Number is FCBF-47B5

Directory of E:\OOPS\Lab Programs\source\Week 4

03/17/2002  03:11 PM    <DIR>          -
03/17/2002  03:11 PM    <DIR>          ..
03/17/2002  03:10 PM                1,150 AboutFile.java
03/17/2002  03:11 PM                1,265 FileAnalysis.java
03/17/2002  03:11 PM                1,005 ReadFile.java
               3 File(s)                3,420 bytes
               2 Dir(s)          7,875,526,656 bytes free

E:\OOPS\Lab Programs\source\Week 4>java ooplabs.AboutFile
Enter the name of the file:
FileAnalysis.java
The file FileAnalysis.java exists
The file FileAnalysis.java is readable
The file FileAnalysis.java is writeable
The file type is: java
The Length of the file:1265

E:\OOPS\Lab Programs\source\Week 4>

```

b)

```

import java.io.*;
class linenum
{
    public static void main(String[] args)throws IOException
    {
        FileInputStream fil;
        LineNumberInputStream line;

```

```

        int i;
        try
        {
            fil=new FileInputStream(args[0]);
            line=new LineNumberInputStream(fil);
        }
        catch(FileNotFoundException e)
        {
            System.out.println("No such file found");
            return;
        }
        do
        {
            i=line.read();
            if(i=='\n')
            {
                System.out.println();
                System.out.print(line.getLineNumber()+" ");
            }
            else
                System.out.print((char)i);
        }while(i!=-1);
        fil.close();
        line.close();
    }
}

```

**Output:**

Demo.java

class Demo

LL {

MM public static void main(java Demo beta gamma delta)

NN {

OO int n = 1 ;

PP System.out.println("The word is " + args[ n ] );

QQ }

C)

**import** java.io.\*;

**class** wordcount

{

```
public static int words=0;
public static int lines=0;
public static int chars=0;
public static void wc(InputStreamReader isr)throws
IOException {
    int c=0;
    boolean lastwhite=true;
    while((c=isr.read())!=-1)
    {
        chars++;
        if(c=='\n')
            lines++;
        if(c=='\t' || c==' ' || c=='\n')
            ++words;
        if(chars!=0)
            ++chars;
    }
}
public static void main(String[] args)
{
    FileReader fr;
    try
    {
        if(args.length==0)
        {
            wc(new InputStreamReader(System.in));
        }
        else
        {
            for(int i=0;i<args.length;i++)
            {
                fr=new FileReader(args[i]);
                wc(fr);
            }
        }
    }
    catch(IOException ie)
    {
        return;
    }
    System.out.println(lines+" "+words+" "+chars);
}
```

Output:This is II CSE

1432

Draw the frequency response

1458

- Write a Java program for handling a) Checked exceptions.  
b) unchecked exceptions.

• **Predefined Exception:** The most general of exceptions are subclasses of the standard type RuntimeException. Since java.lang is implicitly imported into all Java programs, most exceptions derived from RuntimeException are automatically available.

```
class Ex1
{
    void div()
    {
        int b=10/0;
    }
    void div1()
    {
        div();
    }
    void div2()
    {
        try
        {
            div();
        }
        catch(Exception e)
        {
            System.out.println(e);
        }
    }
    public static void main(String[] args)
    {
        Ex1 e=new Ex1();
        e.div2();
    }
}
```

O/P:

Java.lang..ArithmeticException:/by zero

**User defined Exception:** If you are creating your own Exception that is known as custom exception or user-defined exception. Java custom exceptions are used to customize the exception according to user need.

```
class MyException extends Exception
{
```

```
        public String toString()
        {
            return("myException");
        }
    }
class MyExceptionDemo
{
    static void compute(int age) throws MyException
    {
        if(age<18)
            throw new MyException();
        else
            System.out.println("elgible");
    }
    public static void main(String[] args)
    {
        try
        {
            compute(24);
            compute(18);
        }
        catch(MyException e)
        {
            System.out.println(e);
        }
    }
}
```

**O/P:**

MyException

7)Write a JAVA program

Creates three threads. First threads displays “Good Morning “for every one Second, the second thread displays “Hello” for every two seconds, the third thread Displays “Welcome” for every three seconds.

that correctly implements producer consumer problem using concept of inter thread communication

a)

```
class A extends Thread
```

```
{
    synchronized public void run()
    {
        try
        {
            while(true)
            {
                sleep(1000);
                System.out.println("good morning");
            }
        }
        catch(Exception e)
        { }
    }
}
```

```
class B extends Thread
```

```
{
    synchronized public void run()
    {
        try
        {
            while(true)
            {
                sleep(2000);
                System.out.println("hello");
            }
        }
        catch(Exception e)
        { }
    }
}
```

```
class C extends Thread
```





```
{
    synchronized public void run()
    {
        try
        {
            while(true)
            {
                sleep(3000);
                System.out.println("welcome");
            }
        }
        catch(Exception e)
        { }
    }
}
class ThreadDemo
{
    public static void main(String args[])
    {
        A t1=new A();
        B t2=new B();
        C t3=new C();
        t1.start();
        t2.start();
        t3.start();
    }
}
```

**Output:**

```
E:\javamani>java ThreadDemo
good morning
good morning
hello
good morning
welcome
good morning
hello
good morning
welcome
hello
good morning
good morning
hello
good morning
welcome
```

```
good morning
hello
good morning
good morning
hello
welcome
good morning
```

b)

```
class Q
{
    int n;
    boolean valueSet=false;
    synchronized int get()
    {
        if(!valueSet)
            try
            {
                wait();
            }
            catch(InterruptedException e)
            {
                System.out.println("InterruptedException caught");
            }
        System.out.println("Got:"+n);
        valueSet=false;
        notify();
        return n;
    }
    synchronized void put(int n)
    {
        if(valueSet)
            try
            {
                wait();
            }
            catch(InterruptedException e)
            {
                System.out.println("InterruptedException caught");
            }
        this.n=n;
        valueSet=true;
        System.out.println("Put:"+n);
        notify();
    }
}
```

```
class Producer implements Runnable
{
    Q q;
    Producer(Q q)
    {
        this.q=q;
        new Thread(this,"Producer").start();
    }
    public void run()
    {
        int i=0;
        while(true)
        {
            q.put(i++);
        }
    }
}
class Consumer implements Runnable
{
    Q q;
    Consumer(Q q)
    {
        this.q=q;
        new Thread(this,"Consumer").start();
    }
    public void run()
    {
        while(true)
        {
            q.get();
        }
    }
}
class ProdCons
{
    public static void main(String[] args)
    {
        Q q=new Q();
        new Producer(q);
        new Consumer(q);
        System.out.println("Press Control-c to stop");
    }
}
```

Output:

Put:0

Press Control-c to stop

Got:0

Put:1

Got:1

Put:2

Got:2

Put:3

Got:3

Put:4

Got:4

Put:5

Got:5

Put:6

Got:6

8. Develop an Applet that

\* Displays a simple message.

\* receives an integer in one text field, and computes its factorial value and returns it in another text field , when the button named “ Compute” is clicked.

a)

```
* import java.awt.*;
* import java.applet.*;
* /*
* <applet code="sim" width=300 height=300>
* </applet>
* */
* public class sim extends Applet
* {
9         String msg=" ";
10        public void init()
11        {
12                msg+="init()--->";
13                setBackground(Color.orange);
14        }
15        public void start()
16        {
17                msg+="start()--->";
18                setForeground(Color.blue);
19        }
20        }
21        public void paint(Graphics g)
22        {
23                msg+="paint()--->";
24                g.drawString(msg,200,50);
25        }
26 }
```

o/p:

```
b)
import java.awt.*;
import java.awt.event.*;
import java.applet.*;

/*<applet code="Factorial" width=500 height=200></applet>*/

public class Factorial extends Applet implements ActionListener
{
    TextField input,output;
    Button compute;
    int fact=0;
    public void init()
    {
        compute=new Button("Compute");
        Label inp=new Label("Enter any number :",Label.RIGHT);
        Label opt=new Label("Factorial of the given number is :
        ",Label.RIGHT); input=new TextField(5);
        output=new TextField(10);
        input.setBackground(Color.pink);
        output.setBackground(Color.pink);
        add(inp);
        add(input);
        add(opt);
        add(output);
        add(compute);
        output.setText("0");
        output.setEditable(false);
        input.addActionListener(this);
        output.addActionListener(this);
        compute.addActionListener(this);
    }
    public void actionPerformed(ActionEvent ae)
    {
        String str=ae.getActionCommand();
        if(str.equals("Compute"))
        {
            fact=1;
            int n=Integer.parseInt(input.getText());
            if(n<=12)
            {
                for(int i=n;i>=2;i--)
                fact=fact*i;
                output.setText(""+fact);
            }
        }
    }
}
```

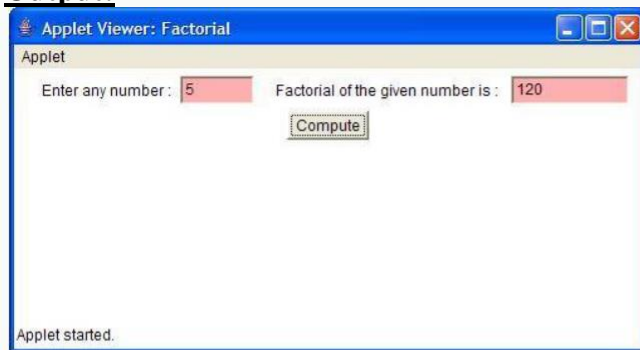




```
        else
        fact=-1;

    }
    repaint();

}
public void paint(Graphics g)
{
    if(fact==-1) {
        output.setText("0");
        g.drawString("Sorry number exceeds greater than 12",10,100); }
}
}
```

**Output:**

C Write a Java program that works as a simple calculator. Use a grid layout to arrange buttons for the digits and for the +,-,\*,/ operations. Add a text field to display the result.

```
import java.awt.*;
import java.awt.event.*;
import java.applet.*;
/*<applet code="Calculator1" width=300 height=300></applet>*/
public class Calculator1 extends Applet implements ActionListener
{
    TextField t;
    Button b[]=new Button[15];
    Button b1[]=new Button[6];
    String op2[]={ "+", "-", "*", "%", "=", "C" };
    String str1="";
    int p=0,q=0;
    String oper;
    public void init()
    {
        setLayout(new GridLayout(5,4));
        t=new TextField(20);
        setBackground(Color.pink);
        setFont(new Font("Arial",Font.BOLD,20));
        int k=0;
        t.setEditable(false);
        t.setBackground(Color.white);
        t.setText("0");
        for(int i=0;i<10;i++)
        {
            b[i]=new Button(""+k);
            add(b[i]);
            k++;
            b[i].setBackground(Color.pink);
            b[i].addActionListener(this);
        }
        for(int i=0;i<6;i++)
        {
            b1[i]=new Button(""+op2[i]);
            add(b1[i]);
            b1[i].setBackground(Color.pink);
            b1[i].addActionListener(this);
        }
        add(t);
    }
}
```

```
public void actionPerformed(ActionEvent ae)
{
    String str=ae.getActionCommand();

    if(str.equals("+")){ p=Integer.parseInt(t.getText());

        oper=str;
        t.setText(str1="");
    }
    else if(str.equals("-")){ p=Integer.parseInt(t.getText());
        oper=str;
        t.setText(str1="");
    }
    else if(str.equals("*")){ p=Integer.parseInt(t.getText());
        oper=str;
        t.setText(str1="");
    }
    else if(str.equals("%")){ p=Integer.parseInt(t.getText());
        oper=str;

        t.setText(str1="");
    }
    else if(str.equals("=")) { str1="";
        if(oper.equals("+")) {
            q=Integer.parseInt(t.getText());
            t.setText(String.valueOf((p+q)));}

        else if(oper.equals("-")) {
            q=Integer.parseInt(t.getText());
            t.setText(String.valueOf((p-q))); }

        else if(oper.equals("*")){
            q=Integer.parseInt(t.getText());
            t.setText(String.valueOf((p*q))); }

        else if(oper.equals("%")){
            q=Integer.parseInt(t.getText());
            t.setText(String.valueOf((p%q))); }
    }

    else if(str.equals("C")){ p=0;q=0;
        t.setText("");
        str1="";
        t.setText("0");
    }
}
```

```
else{ t.setText(str1.concat(str));  
      str1=t.getText();  
    }  
  
  }  
  
}
```

**Output:**

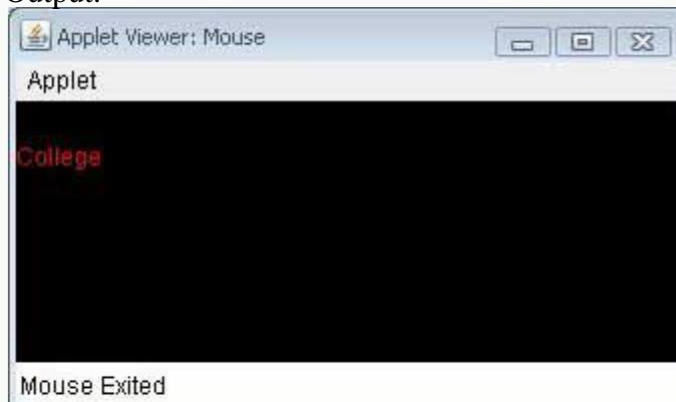
1. Write a Java Program for  
handling a) Mouse events  
b) key events

a)

```
import java.awt.*;
import java.awt.event.*;
import java.applet.*;
/*
<applet code="Mouse" width=500 height=500>
</applet>
*/
public class Mouse extends Applet
implements MouseListener,MouseMotionListener
{
    int X=0,Y=20;
    String msg="MouseEvents";
    public void init()
    {
        addMouseListener(this);
        addMouseMotionListener(this);
        setBackground(Color.black);
        setForeground(Color.red);
    }
    public void mouseEntered(MouseEvent m)
    {
        setBackground(Color.magenta);
        showStatus("Mouse Entered");
        repaint();
    }
    public void mouseExited(MouseEvent m)
    {
        setBackground(Color.black);
        showStatus("Mouse Exited");
        repaint();
    }
    public void mousePressed(MouseEvent m)
    {
        X=10;
        Y=20;
        msg="NEC";
        setBackground(Color.green);
        repaint();
    }
    public void mouseReleased(MouseEvent m)
```

```
{
    X=10;
    Y=20;
    msg="Engineering";
    setBackground(Color.blue);
    repaint();
}
public void mouseMoved(MouseEvent m)
{
    X=m.getX();
    Y=m.getY();
    msg="College";
    setBackground(Color.white);
    showStatus("Mouse Moved");
    repaint();
}
public void mouseDragged(MouseEvent m)
{
    msg="CSE";
    setBackground(Color.yellow);
    showStatus("Mouse Moved"+m.getX()+" "+m.getY());
    repaint();
}
public void mouseClicked(MouseEvent m)
{
    msg="Students";
    setBackground(Color.pink);
    showStatus("Mouse Clicked");
    repaint();
}
public void paint(Graphics g)
{
    g.drawString(msg,X,Y);
}
}
```

Output:



□ Key events **import**

```
java.awt.*; import  
java.awt.event.*; import  
java.applet.*; /*
```

```
<applet code="Key" width=300 height=400>  
</applet>  
*/
```

```
public class Key extends  
Applet implements KeyListener
```

```
{  
    int X=20,Y=30;  
    String msg="KeyEvents--->";  
    public void init()  
    {  
        addKeyListener(this);  
        requestFocus();  
        setBackground(Color.green);  
        setForeground(Color.blue);  
    }  
    public void keyPressed(KeyEvent k)  
    {  
        showStatus("KeyDown");  
        int key=k.getKeyCode();  
        switch(key)  
        {  
            case KeyEvent.VK_UP:  
                showStatus("Move to Up");  
                break;  
            case KeyEvent.VK_DOWN:  
                showStatus("Move to Down");  
                break;  
            case KeyEvent.VK_LEFT:  
                showStatus("Move to Left");  
                break;  
            case KeyEvent.VK_RIGHT:  
                showStatus("Move to Right");  
                break;  
        }  
        repaint();  
    }  
    public void keyReleased(KeyEvent k)  
    {  
        showStatus("Key Up");  
    }  
}
```

```
public void keyTyped(KeyEvent k)
{
    msg+=k.getKeyChar();
    repaint();
}
public void paint(Graphics g)
{
    g.drawString(msg,X,Y);
}
}
```





□ Write a Java program that creates a user interface to perform integer divisions. The user enters two numbers in the text fields num1 and num2. The division of num1 and num2 is displayed on the result field when the divide button is clicked. If num1 and num2 were not an integer, the program would throw number format exception. If num2 was zero , the program would throw an arithmetic exception and display the exception in message dialogue box.

```
import java.awt.*;
import java.awt.event.*;
import java.applet.*;
/*<applet code="Div"width=230 height=250>
</applet>*/
public class Div extends Applet implements ActionListener
{
String msg;
TextField num1,num2,res;Label l1,l2,l3;
Button div;
public void init()
{
l1=new Label("Number 1");
l2=new Label("Number 2");
l3=new Label("result");
num1=new TextField(10);
num2=new TextField(10);
res=new TextField(10);
div=new Button("DIV");
div.addActionListener(this);
add(l1);
add(num1);
add(l2);
add(num2);
add(l3);
add(res);
add(div);
}
public void actionPerformed(ActionEvent ae)
{
String arg=ae.getActionCommand();
if(arg.equals("DIV"))
{
String s1=num1.getText();
String s2=num2.getText();
int num1=Integer.parseInt(s1);
int num2=Integer.parseInt(s2);
```



```
if(num2==0)
{
try
{
System.out.println(" ");
}
catch(Exception e)
{
System.out.println("ArithmeticException"+e);
}
msg="Arithmetic";
repaint();
}
else if((num1<0)||(num2<0))
{
try
{
System.out.println("");
}
catch(Exception e)
{
System.out.println("NumberFormat"+e);
}
msg="NumberFormat";
repaint();
}
else
{
int num3=num1/num2;
res.setText(String.valueOf(num3));
}
}
}
public void paint(Graphics g)
{
g.drawString(msg,30,70);
}
}
```



Output

---

Number 1  Number 2  result

---

12. Write a JAVA program that

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

b)allows the user to draw lines rectangles and ovals.

a)

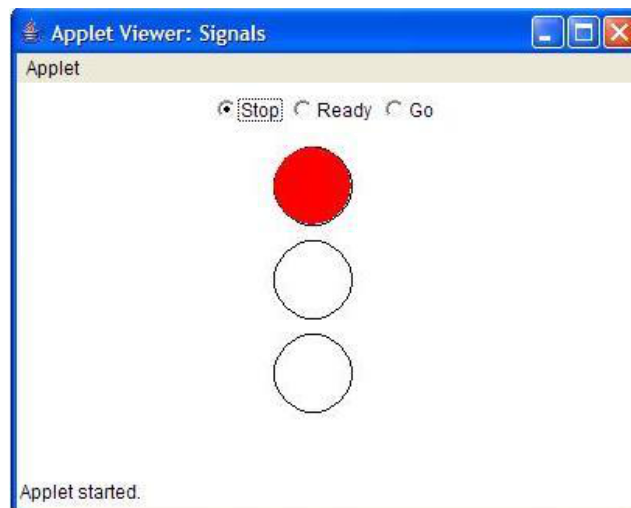
```
import java.applet.*;
import java.awt.*;
import java.awt.event.*;
/*<applet code="Signals" width=400 height=250></applet>*/
public class Signals extends Applet implements ItemListener
{
    String msg="";
    Checkbox stop,ready,go;
    CheckboxGroup cbg;
    public void init()
    {
        cbg = new CheckboxGroup();
        stop = new Checkbox("Stop", cbg, false);
        ready = new Checkbox("Ready", cbg, false);
        go= new Checkbox("Go", cbg, false);
        add(stop);
        add(ready);
        add(go);
        stop.addItemListener(this);
        ready.addItemListener(this);
        go.addItemListener(this);
    }

    public void itemStateChanged(ItemEvent ie)
    {
        repaint();
    }

    public void paint(Graphics g)
    {
        msg=cbg.getSelectedCheckbox().getLabel();
        g.drawOval(165,40,50,50);
        g.drawOval(165,100,50,50);
    }
}
```

```
g.drawOval(165,160,50,50);

if(msg.equals("Stop"))
{
    g.setColor(Color.red);
    g.fillOval(165,40,50,50);
}
else if(msg.equals("Ready"))
{
    g.setColor(Color.yellow);
    g.fillOval(165,100,50,50);
}
else
{
    g.setColor(Color.green);
    g.fillOval(165,160,50,50);
}
}
}
```

**Output:**

b)

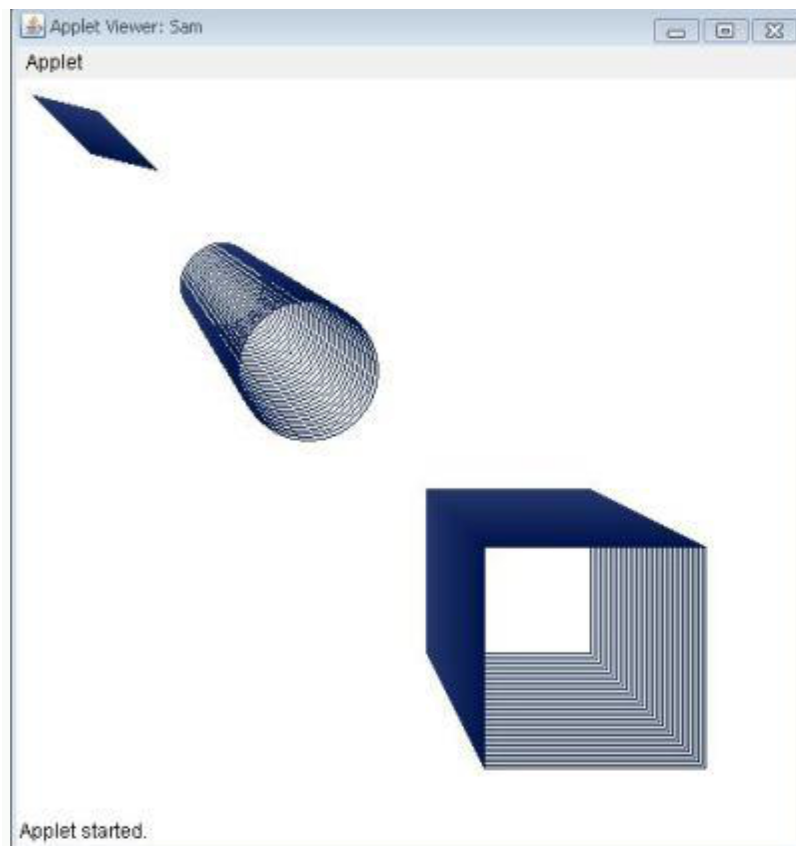
```
import java.awt.*;
import java.applet.*;
```

/\*

```
<applet code="Sujith" width=200 height=200>
```

```
</applet>
*/
public class Sujith extends Applet
{
    public void paint(Graphics g)
    {
        for(int i=0;i<=250;i++)
        {
            Color c1=new Color(35-i,55-i,110-i);
            g.setColor(c1);
            g.drawRect(250+i,250+i,100+i,100+i);
            g.drawOval(100+i,100+i,50+i,50+i);
            g.drawLine(50+i,20+i,10+i,10+i);
        }
    }
}
```

Output:





**MREC (A)**  
**No:50**

**Page**





