

# LAB MANUAL

PROGRAMMING LAB-JAVA

|  |  |
| --- | --- |
| **AUTHORS** | |
| **Dr.V.Sathyavathy,** | **Ms. V.Nithya,** |
| Head of the Department, | Assistant Professor, |
| Department of Computer Technology, | Department of Computer Technology, |
| KG College of Arts and Science, | KG College of Arts and Science, |
| Coimbatore. | Coimbatore. |
| [sathyavathy.v@kgcas.com](mailto:sathyavathy.v@kgcas.com) | [nithya.v@kgcas.com](mailto:nithya.v@kgcas.com) |

Programming Lab-Java

© Dr.V.Sathyavathy, Ms.V.Nithya First Published: 2021

Published by: Inam Pathippagam, Coimbatore – 641016. Pages: 43

ISBN: 978-93-92293-61-0



**SYLLABUS**

1. Write a Java Applications to extract a portion of a character string and print the extracted string.
2. Write a Java Program to implement the concept of multiple inheritance using Interfaces.
3. Write a Java Program to create an Exception called payout-of-bounds and throw the exception.
4. Write a Java Program to implement the concept of multithreading with the use of any three multiplication tables and assign three different priorities to them.
5. Write a Java Program to draw several shapes in the created windows.
6. Write a Java Program to create a frame with four text fields name, street, city and pin code with suitable tables. Also add a button called my details. When the button is clicked its corresponding values are to be appeared in the text fields.
7. Write a Java Program to demonstrate the Multiple Selection List-box.
8. Write a Java Program to create a frame with three text fields for name, age and qualification and a text field for multiple lines for address
9. Write a Java Program to create Menu Bars and pull down menus.
10. Write a Java Program to create frames which respond to the mouse clicks. For each events with mouse such as mouse up, mouse down, etc., the corresponding message to be displayed.
11. Write a Java Program to draw circle, square, ellipse and rectangle at the mouse click positions.
12. Write a Java Program which open an existing file and append text to that file.

# PREFACE

One of the most significant advantages of Java is **its ability to move easily from one computer system to another.** The ability to run the same program on many different systems is crucial to World Wide Web software, and Java succeeds at this by being platform-independent at both the source and binary levels. Java is **used to build applications and platforms for a number of devices**, including computers, laptops, gaming consoles, Blu-ray players, car navigation systems, medical monitoring devices, parking meters, lottery terminals and smartphones.

# INDEX

|  |  |  |
| --- | --- | --- |
| **S. No.** | **TITLE** | **PAGE No.** |
| **1.** | **STRING EXTRACTION** | **1** |
| **2.** | **MULTIPLE INHERITANCE** | **3** |
| **3.** | **PAYOUT OF BOUNDS EXCEPTION** | **7** |
| **4.** | **MULTITHREADING** | **10** |
| **5.** | **SHAPES DEMO** | **14** |
| **6.** | **TEXT FIELD DEMO** | **17** |
| **7.** | **MULTIPLE SELECTION LIST BOX** | **21** |
| **8.** | **TEXT AREA DEMO** | **24** |
| **9.** | **MENU DEMO** | **27** |
| **10.** | **MOUSE EVENTS** | **29** |
| **11.** | **DRAWING SHAPES USING MOUSE CLICKS** | **34** |
| **12.** | **RANDOM ACCESS FILES** | **37** |

**Ex.no:01 STRING EXTRACTION**

# Aim:

To write a java program to extract a portion of a character string and print the extracted string.

# Algorithm:

Step 1: Start the process.

Step 2: Import all the java packages from the java library. Step 3: Create a class and name it as ‘Extract String’.

Step 4: Declare the variables to get the input and to be stored.

Step 5: Get the string input from the user and store it in the variable.

Step 6: Get the index values from the user and store it in the declared variable. Step 7: Assign the integer values to the substring

Step 8: Compile and run the program.

Step 9: The substring of the given string has been displayed in the output screen. Step 10: Stop the process.

# Source code:

import java.io.\*;

import java.util.Scanner; public class ExtractString

{

public static void main(String args[])

{

String str,substr ; int idx1,idx2;

Scanner in = new Scanner(System.in); System.out.println("Enter the string :"); str = in.next();

System.out.println("Enter the value of index 1:"); idx1 = in.nextInt();

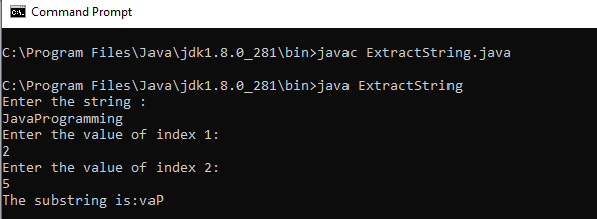
System.out.println("Enter the value of index 2:"); idx2 = in.nextInt();

substr = str.substring(idx1,idx2); System.out.println("The substring is:"+substr);

}

}

# Output:



**Ex.no:02 MULTIPLE INHERITANCE**

# Aim:

To write a java program to implement the concepts of multiple inheritance using interfaces.

# Algorithm:

Step 1: Start the process.

Step 2: Import the required java package from the java library. Step 3: Define a class student with required data.

Step 4: Derive a class test from the student that includes data for the class test. Step 5: Create a interface sports and assign the value for sports.

Step 6: Create display function to give the total marks.

Step 7: Get the total marks and print it using the println method.

Step 8: Create a main class to create an object and invoke the class using object.

Step 9: The given details will display on the output screen once the program has been executed. Step 10: Stop the process.

**Source code:** import java.io.\*; class student

{

int rno;

String name="sindhu"; void getno(int n)

{

rno=n;

}

void putno()

{

System.out.println("rno:"+no); System.out.println("name:" + name);

}

}

class test extends student

{

int mark1,mark2;

void getmarks(int m1,int m2)

{

mark1=m1; mark2=m2;

}

public void Putmarks()

{

System.out.println("mark1:"+mark1); System.out.println("mark2:"+mark2);

}

}

interface sports

{

int sportsmark=75; void putmarks();

}

class result extends test implements sports

{

int total;

public void putmarks()

{

System.out.println("sportsmark="+sportsmark);

}

void display()

{

total=mark1+mark2+sportsmark; putno();

Putmarks(); putmarks();

System.out.println("Total marks:"+total);

}

}

class MultipleInheritance

{

public static void main(String args[])

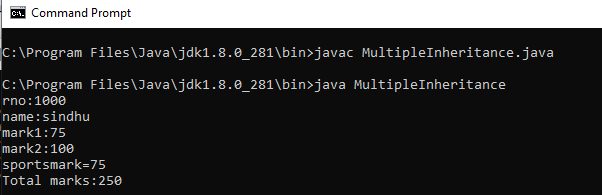
{

result stud=new result(); stud.getno(1000); stud.getmarks(75,100); stud.display();

}

}

# Output:



**Ex.no:03 PAYOUT OF BOUNDS EXCEPTION**

# Aim:

To write a java program to handle exceptions by using PayoutofBound exception.

# Algorithm:

Step 1: Start the process.

Step 2: Create a class Payout and under main function declare the integer variable payment. Step 3: Create an object in using Datainputstream to get the input values.

Step 4: Read the input value using Integer.parseInt and store that value in the variable payment. Step 5: Inside the try block check if payment is greater than 1000 or not if it greater than 1000 the payoutofBoundException to catch block otherwise display the basic pay.

Step 6: Inside the catch block caught the exception and create a classpayoutofBound Exception extends from Exception.

Step 7: under that class pass the exception Message. Step 8: Stop the process.

**Source code:** import java.io.\*; import java.lang.\*;

class ExceptionClass

{

public static void main(String args[]) throws IOException

{

int payamt;

DataInputStream in=new DataInputStream(System.in); System.out.println("\n\npay out of bound exception"); System.out.println("\* \* \*");

System.out.println("\n enter a basic pay amount"); payamt =Integer.parseInt(in.readLine());

try { if(payamt>1000)

throw new payoutofBoundException("Basic pay is out of bound"); else

System.out.println("\n given basic pay is:"+payamt); } catch(Exception e) {

System.out.println("caught:"+e);

}

} }

class payoutofBoundException extends IOException

{

payoutofBoundException(String message)

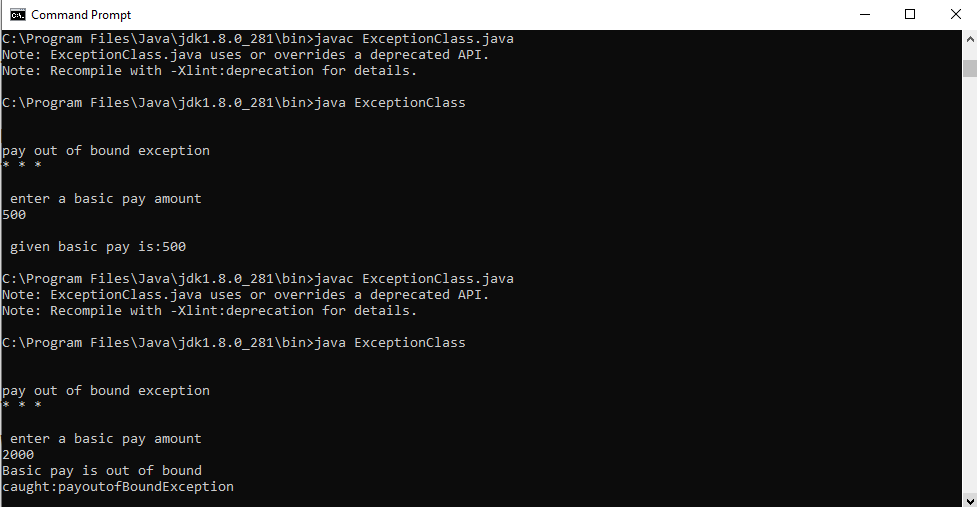
{

System.out.println(message);

}

}

# Output:



**Ex.no:04 MULTITHREADING**

# Aim:

Write a Java program to illustrate the use of thread classes with priority.

# Algorithm:

Step 1: Start the process.

Step 2: Create a class A extends from the thread class and define run()

Step 3: Under the run() use the for loop to display the multiple of second table. Step 4: Create a class B extends from the thread class and define run().

Step 5: Under the run() use for loop to display the multiples of the third table . Step 6: Create a new class C extends from thread class and define run().

Step 7: Under the run() use for loop to display the multiples of fifth tables.

Step 8: Create a ThreadTest and under the main function create an object A,B,C for the class A,B and C and setpriority for all the objects.

Step 9: start method is used to start the object A,B,C and display the output. Step 10: Stop the process.

# Source code:

import java.io.\*;

import java.lang.Thread; class A extends Thread

{

public void run()

{

int i=0,j=2,k; for(i=1;i<=4;i++)

{

k=i\*j;

System.out.println("From thread A:"+i+"\*"+j+"="+k);

}

System.out.println("Exit fromA");

}}

class B extends Thread

{

public void run() { int a,b=3,c;

for(a=1;a<=4;a++) { c=a\*b;

System.out.println("From thread B: "+a+"\*"+b+"="+c); } System.out.println("Exit from B"); }

}

class C extends Thread

{

public void run() { int x,y=5,z; for(x=1;x<=4;x++) { z=x\*y;

System.out.println("From thread C:"+x+"\*"+y+"="+z); } System.out.println("Exit from C"); }

}

class ThreadTest

{

public static void main(String args[])

{

int x=0;

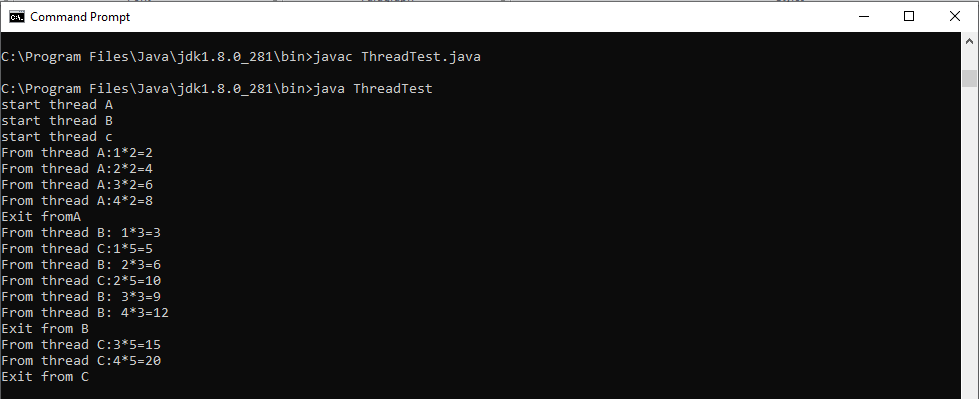
A threadA=new A(); B threadB=new B(); C threadC=new C();

threadC.setPriority(Thread.MAX\_PRIORITY); threadB.setPriority(threadA.getPriority()+x); threadC.setPriority(Thread.MIN\_PRIORITY); threadB.setPriority(threadA.getPriority()+x); threadC.setPriority(Thread.MIN\_PRIORITY); System.out.println("start thread A"); threadA.start();

System.out.println("start thread B"); threadB.start(); System.out.println("start thread c"); threadC.start();

} }

# Output:



**Ex.no:05 SHAPES DEMO**

# Aim:

To write a java program to draw Several Shapes in the created windows.

# Algorithm:

Step 1: Start the process

Step 2: Import the program .awt and .applet for using the applet function.

Step 3: Create a class Shape and extends from the Applet and define the function paint().

Step 4: Under the paint() pass the parameter (Graphics g) by using the object g call the different functions of applet.

Step 5: Using object g we call drawRect(), drawRoundRect(), drawLine(), drawOval(), fillOval() using the suitable coordinates.

Step 6: Display the different shapes in windows.

Step 7: The Applet tag </applet> is used to set the width and height for the Applet window. Step 8: Stop the process.

**Source code:** import java.awt.\*; import java.applet.\*;

public class shapes extends Applet

{

int x1[]={60,240,440,40};

int y1[]={60,240,40,40};

int n1=4;

public void paint(Graphics g)

{

g.drawPolygon(x1,y1,n1); g.drawLine(20,20,160,280); g.drawRect(300,200,160,280);

g.fillRect(200,200,80,40); g.fillRoundRect(40,320,20,60,10,10); g.drawRoundRect(20,300,160,100,20,20); g.drawOval(550,400,100,40);

}

}

/\*<html>

<head>

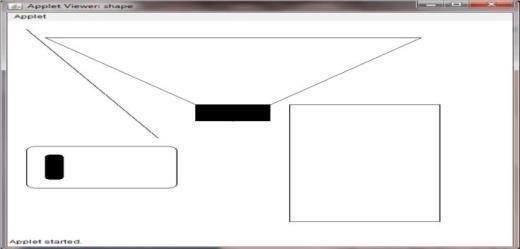
<applet code="shapes",height=400 width=400>

</applet>

</head>

</html>\*/

# Output:



**Ex.no:06 TEXT FIELD DEMO**

# Aim:

To create a frame with 4 fields Name, Street, City, Pin with suitable labels also add a button called details when the button is click the corresponding values are to be appeared in the field.

# Algorithm:

Step1: Start the process

Step2: Import the statement java.awt and java.applet for using the applet functions and graphics Step3: Declare a class ButtonClick and extends from Applet and implements the ActionListener Step4: Declare the Textfield as t1,t2,t3,t4 and declare Label as l1,l2,l3,l4 and button b1

Step5: Assign the new flowlayout and textfields,label and buttons in layout

Step6: Add the Actionlistener to the button b1 and also use this pointer

Step7: Under the Actionperformed function declare the object ae for the actionevent Step8: Use if condition to check whether the getsource is equal to b1

Step9: Settext tp all the textfields and display the result using Appletviewers Step10: Stop the process

**Source code:**

import java.awt.\*; import java.awt.event.\*;

public class ButtonClick extends Frame implements ActionListener

{

TextField t1,t2,t3,t4;

Label l1,l2,l3,l4;

Button b1;

Buttonclick()

{

setLayout(new GridLayout(4,2)); t1=new TextField(30);

t2=new TextField(30);

t3=new TextField(30); t4=new TextField(30);

l1=new Label("name:",Label.LEFT); l2=new Label("street:",Label.LEFT);

l3=new Label("city:",Label.LEFT); l4=new Label("place:",Label.LEFT); b1=new Button("my details");

add(l1);

add(t1);

add(l2);

add(t2);

add(l3);

add(t3);

add(l4);

add(t4);

setLayout(new FlowLayout(FlowLayout.CENTER)); b1.addActionListener (this);

add(b1);

}

public void actionPerformed(ActionEvent ae) { if(ae.getSource()==b1) {

t1.setText("Vivek Sabu"); t2.setText("Nehru Nagar"); t3.setText("Texas City"); t4.setText("20");

} }

public static void main(String arg[]) { Buttonclick m= new Buttonclick(); m.setSize(600,300); m.addWindowListener(new WindowAdapter(){ public void windowClosing(WindowEvent e)

{

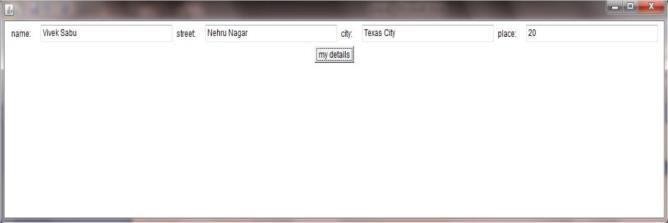
System.exit(0); }

});

m.show();

} }

# Output:



**Ex.no:07 MULTIPLE SELECTION LIST BOX**

# Aim:

To write a Java program to demonstrate the Multiple selection list box.

# Algorithm:

Step 1: Start the process.

Step 2: The applet tag<applet> is used to set the width and height for the Applet window. Step 3: Declare a class MultipleSelectionList extends from the class applet.

Step 4: Declare the list as list1,Textfield text1 and button butt. Step 5: Under the function init() add list for multiple options. Step 6: Declare an array of string Selection[].

Step 7: Use if condition to check whether the getSource is equal to butt. Step 8: SetText the textfield to outstring variable.

Step 9: Display the results. Step 10: Stop the process.

# Source code:

import java.applet.Applet; import java.awt.\*;

import java.awt.event.\*;

public class MultipleSelectionList extends Applet implements ActionListener

{

List list1; TextField Text1; Button butt; public void init()

{

Text1=new TextField(40); add(Text1);

list1=new List(3,true); list1.add("Windows NT"); list1.add("Windows Vista"); list1.add("Windows XP"); list1.add("LINUX");

list1.add("UNIX");

list1.add("Xenix"); add(list1);

butt=new Button("Show Selection"); butt.addActionListener(this); add(butt);

}

String Selections[];

public void actionPerformed(ActionEvent e) { String outstring=new String("you selected"); if(e.getSource()==butt)

{

Text1.setText(""); Selections=list1.getSelectedItems();

for(int loopIndex=0;loopIndex<Selections.length;loopIndex++)

{ outstring+=" "+Selections[loopIndex]; } Text1.setText(outstring); }

} }

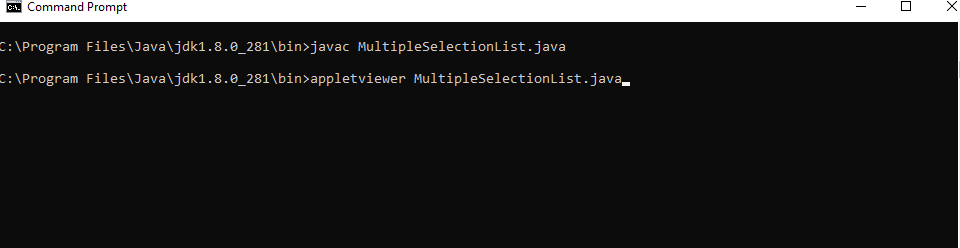
/\*

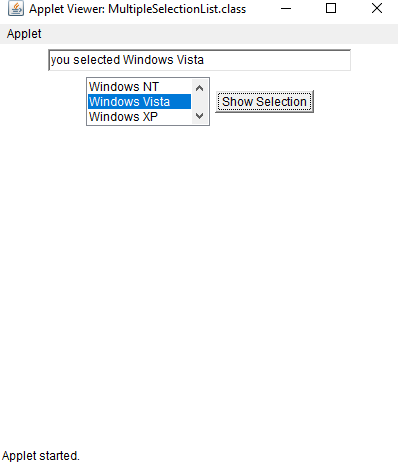
<html>

<applet code = MultipleSelectionList.class width=400 height=400> </applet>

</html> \*/

# Output:





**Ex.no:08 TEXTAREA DEMO**

# Aim:

To create a frame with 3 text fields name, age, qualification and a text field for multiple

lines for address.

# Algorithm:

Step 1: Start the process

Step 2: Import the statement java.awt and java.applet

Step 3: Declare a class personaldetails which extends the class Frame

Step 4: Declare the TextField as t1, t2, t3, t4, Label as l1, l2, l3, l4 and TextArea ta. Step 5: Set Layout to new GridLayout.

Step 6: Under main function create an object for personaldetails. Step 7:Use windowClosing() function to exit from the event.

Step 8: Call the show() using the object. Step 9: Stop the process.

# Source code:

import java.awt.\*; import java.awt.event.\*;

public class personaldetails extends Frame

{

TextField t1,t2,t3,t4; Label l1,l2,l3,l4; TextArea ta; personaldetails()

{

setLayout(new GridLayout(4,2)); t1=new TextField(30);

t2=new TextField(30); t3=new TextField(30); ta=new TextArea(" ",2,5);

l1=new Label("Name",Label.LEFT); l2=new Label("Age",Label.LEFT);

l3=new Label("Qualification",Label.LEFT);

l4=new Label("Address",Label.LEFT); add(l1);

add(t1);

add(l2);

add(t2);

add(l3);

add(t3);

add(l4);

add(ta);

}

@SuppressWarnings("deprecation") public static void main(String [] args)

{

personaldetails m=new personaldetails(); m.setSize(300,250);

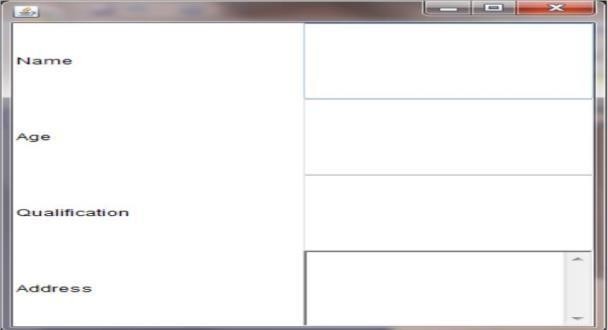
m.addWindowListener(new WindowAdapter() { public void windowClosing(WindowEvent e) { System.exit(0); }

} );

m.show(); }

}

# Output:



**Ex.no:09 MENU DEMO**

# Aim:

To write a Java program to create menu bar and pulldown menus.

# Algorithm:

Step 1: Start the process. Step 2: Import the packages.

Step 3: Declare the class Menudemo extends class applet. Step 4: Under the class declare the init function.

Step 5: Under the function init() declare height and width. Step 6: Create object for Frame and an object for Menubar.

Step 7: Create an object for Menu and add the menuitems with the add(). Step 8: Create another object edit to add checkbox menu items.

Step 9: The applet tag </applet> is used to set the width and height of the applet window. Step 10: Stop the process.

# Source code:

import java.awt.\*; import java.applet.\*;

public class Menudemo extends Applet { public void init()

{

int width=Integer.parseInt(getParameter("width")); int height=Integer.parseInt(getParameter("height")); Frame f=new Frame("demo Frame"); f.resize(width,height);

MenuBar mbar=new MenuBar(); f.setMenuBar(mbar);

Menu fi=new Menu("File"); fi.add(new MenuItem("New")); fi.add(new MenuItem("Open")); fi.add(new MenuItem("Close")); fi.add(new MenuItem("-"));

fi.add(new MenuItem("new")); mbar.add(fi);

Menu edit=new Menu("Edit");

edit.add(new MenuItem("Cut")); edit.add(new MenuItem("Copy")); edit.add(new MenuItem("paste")); edit.add(new MenuItem("-")); Menu sub=new Menu("Special"); sub.add(new MenuItem("First")); sub.add(new MenuItem("Second")); sub.add(new MenuItem("Third")); edit.add(sub);

edit.add(new CheckboxMenuItem("Debug")); edit.add(new CheckboxMenuItem("Testing")); mbar.add(edit);

f.show();

}

} /\*<html>

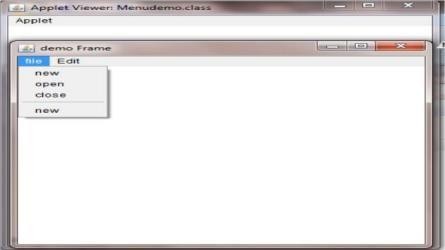
<head>

<applet code="Menudemo.class" width=400 height=400>

</applet>

</head> </html>\*/

# Output:



**Ex.No:10 MOUSE EVENTS**

# Aim:

To create a frame which respond to the mouse click for each event with the mouse such

as mouseup, down,etc,.,. The corresponding message to be displayed.

# Algorithm:

Step 1: Start the process.

Step 2: The applet tag is used to set the width and height for the Applet window.

Step 3: Create a class MouseEvents extends from Applet implements by MouseListener, MouseMotionListener.

Step 4: Declare the string s and under this method to add mouseListener and mousemotionlistener.

Step 5: Declare the function mouseclicked(), mouseentered(), mousefield(), mouseExited(),MousePressed(), MouseDraged(), MouseMoved(). Under this function assign the entered, clicked, excited, released, dragged and moved toS amd repaint().

Step 6: Under the paint() use the method drawstring and pass the parameter. Step 7: Stop the Process.

# Source code:

import java.applet.\*; import java.awt.\*; import java.awt.event.\*;

public class MouseEvents extends Applet implements MouseListener, MouseMotionListener

{

String msg = " "; int x,y,mousex = 0, mousey = 0;

public void init()

{

addMouseListener(this); addMouseMotionListener(this); }

public void mousePressed(MouseEvent me) { x=50;

y=100;

msg="Mouse Pressed"; repaint();

}

public void MouseClicked(MouseEvent me) { x=50;

y=100;

msg="Mouse Clicked"; repaint();

}

public void MouseExited(MouseEvent me) { x=50; y=100;

msg="Mouse Exited"; repaint();

}

public void MouseReleased(MouseEvent me) { x=50; y=100;

msg="Mouse Released"; repaint();

}

public void mouseEntered(MouseEvent me) { x=50;

y=100;

msg="mouse Entered"; repaint();

}

public void mouseDragged(MouseEvent me) { msg="";

showStatus("Dragged at"+me.getX()+","+me.getY()); repaint();

}

public void mouseMoved(MouseEvent me) { showStatus("Moved at"+me.getX()+","+me.getY()); repaint();

}

public void paint(Graphics g) {

g.drawString(msg,x,y); }

}

/\*<html>

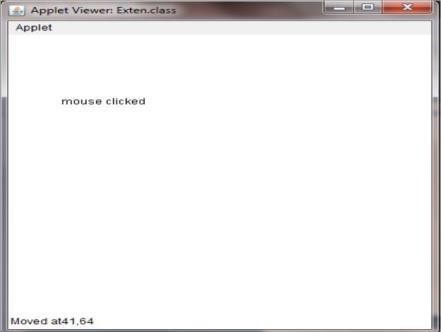
<head>

<applet code="MouseEvents", width=400 height=400></applet>

</head>

</html>\*/

# Output:



**Ex.no:11 DRAWING SHAPES USING MOUSE CLICKS**

# AIM:

To write a java program to draw circle, square, eclipse and rectangle at the mouse clicks positions.

# ALGORITHM:

Step 1: Start the process.

Step 2: Import the necessary java packages from java library. Step 3: Declare a class Mouseclick extended to applet.

Step 4: Declare x,y,e as an integer.

Step 5: Using the init() method add the mouselistener and mouseadapter.

Step 6: In mouse pressed event give c++ and getme.getx() to assigning x and getme.gety() to assigning y.

Step 7: In paint() method give if condition and draw oval using object g. Step 8: Give else if condition drawRect() and drawOval() using object g. Step 9: In html file give the applet code to invoke class.

Step 10: Output will be displayed on the output screen once the program has been executed. Step 11: Stop the process.

**Source code:** import java.applet.\*; import java.awt.\*;

import java.awt.event.\*;

public class Mouseclick extends Applet { int x=0,y=0,c=0;

public void init() { addMouseListener(new MouseAdapter()

{

public void mousePressed(MouseEvent me) { c++;

x = me.getX();

y = me.getY(); repaint();

}

});

}

public void paint(Graphics g) { if(c%4==1) g.drawOval(x,y,150,150);

else if(c%4==2) g.drawRect(x,y,150,150); else if(c%4==3) g.drawOval(x,y,100,150); else if(c%4==0) g.drawRect(x,y,100,150);

}

}

/\*<html>

<head>

<title>Mouse click</title>

</head>

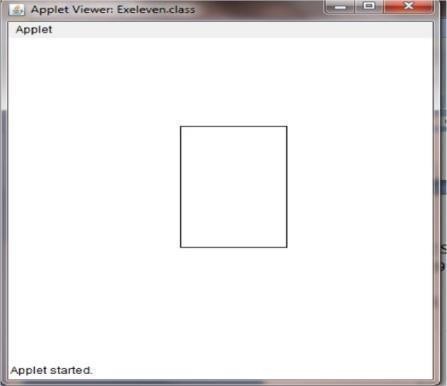
<body>

<applet code = Mouseclick.class width=400 height=400> </applet>

</body> </html>

\*/

# Output:



**Ex.no:12 RANDOM ACCESS FILES**

# AIM:

To write a java program which open an existing and append text to the file.

# ALGORITHM:

Step 1: Start the process.

Step 2: Include all the necessary package from the java library. Step 3: It is a random access file so declare the file as r file.

Step 4: Create an object for the file and the text file should be declared in try. Step 5: Throw the random access file.

Step 6: Catch IOE and the object. Step 7: Print the IOException.

Step 8: Execute the program.

Step 9: Continue to run if the error is 0. Step 10: Stop the program.

# Source code:

import java.io.\*;

class RandomAccessFile1 {

public static void main(String args[]) { RandomAccessFile rFile;

try

{

rFile=new RandomAccessFile("city.txt","rw"); rFile.seek(rFile.length()); rFile.writeBytes("\nCHENNAI"); rFile.close();

}

catch(IOException ioe)

{ System.out.println(ioe); }

} }

# Output:

