

OBJECT ORIENTED PROGRAMMING **LAB**

SUBMITTED BY:
BLESSY P ROY
S2 MCA
ROLL NO:214

LABCYCLE 5

PROGRAM 1

AIM

Program to draw Circle, Rectangle, Line in Applet

ALGORITHM

Step.1: Start

Step.2: Define a class 'appshape' that extends Applet class.

Step.3: Draw a line, rectangle and circle using drawLine, drawRect and drawOval methods of Graphics class respectively.

Step.4: Stop


PROGRAM CODE

```
import java.awt.*;
import java.applet.*;
public class appshape extends Applet
{
    public void paint(Graphics g)
    {
        g.setColor(Color.red);
        g.drawLine(10,10,250, 10);
        g.drawRect(220, 150, 200, 200);
        g.drawOval(20,20,200,120);
    }
}
/*
<applet code = "appshape.class" width = "420" height = "320"
border="5"></applet>
*/
```

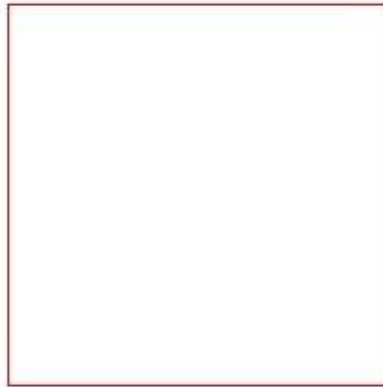
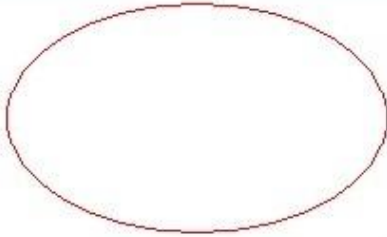
RESULT

The above program is successfully executed and obtained the output

OUTPUT

 Applet Viewer: appshape.class

Applet



PROGRAM 2

AIM

Program to find maximum of three numbers using AWT.

ALGORITHM

Step.1: Start.

Step.2: Define a class 'large' that extends Applet class and implements ActionListener interface.

Step.3: Using TextField class object, construct the required no. of Text Fields wide enough to hold the values entered by the user.

Step.4: Using Button class object, construct a labeled button that sends an instance of ActionEvent.

Step.5: Call addActionListener() method to send events from the button to the new listener.

Step.6: Get the string values from textfields and then parse them as integers.

Step.7: Compare each value using if-else statements to find the maximum value and set the result accordingly.

Step.8: Stop

PROGRAM CODE

```
import java.applet.*;
import java.awt.*;
import java.awt.event.*;
public class large extends Applet implements
ActionListener
{
    TextField t1,t2,t3,t4;
    Button b1;
    public void init()
    {
        setLayout(null);
        t1 = new TextField(15);
        t1.setBounds(100,25,50,20);
        t2 = new TextField(15);
        t2.setBounds(100,50,50,20);
        t3 = new TextField(15);
        t3.setBounds(100,75,50,20);
        t4 = new TextField("Ans");
        t4.setBounds(175,40,50,20);
```

	<pre>b1 = new Button("Find"); b1.setBounds(175,65,50,30); add(t1); add(t2); add(t3); add(t4); add(b1); b1.addActionListener(this); } public void actionPerformed(ActionEvent e) { int i,j,k; i = Integer.parseInt(t1.getText()); j=Integer.parseInt(t2.getText()); k=Integer.parseInt(t3.getText()); if(i<j) { if(j<k) t4.setText(""+k); else t4.setText(""+j); } else t4.setText(""+i); } } /* <applet code="large.class" height=500 width=800> </applet> */</pre>
--	---

RESULT

The above program is successfully executed and obtained the output.

OUTPUT



PROGRAM 3

AIM

Find the percentage of marks obtained by a student in 5 subjects. Display a happy face if he secures above 50% or a sad face if otherwise

ALGORITHM

Step.1: Start.

Step.2: Define a class 'smile' that extends Applet class and implements ActionListener interface.

Step.3: Using TextField class object, construct textfields to receive marks of 5 subjects from the user.

Step.4: Using Button class object, construct a labeled button that sends an instance of ActionEvent.

Step.5: Call addActionListener() method to send events from the button to the new listener.

Step.6: Get the string values from textfields and then parse them as float values.

Step.7: Calculate the percentage:

$$\text{percent} = ((m1+m2+m3+m4+m5)*100)/500$$

Step.8: Define a paint() method that contains functions from Graphics class to display a happy face if student secures above 50% or a sad face if otherwise

Step.9: Stop.

PROGRAM CODE

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

public class smile extends Applet implements ActionListener {
    Label l1,l2,l3,l4,l5,l6;
    TextField t1,t2,t3,t4,t5,t6;
    Button b;
    public void init(){
        l1 = new Label(" MARK 1:");
        t1 = new TextField();
        l2 = new Label(" MARK 2 :");
        t2 = new TextField();
```


	<pre> l3 = new Label(" MARK 3 :"); t3 = new TextField(); l4 = new Label("MARK 4 :"); t4 = new TextField(); l5 = new Label("MARK 5 :"); t5 = new TextField(); l6 = new Label("PERCENTAGE:"); t6 = new TextField(); b = new Button("RESULT"); setLayout(null); l1.setBounds(450,50,70,20); t1.setBounds(520,50,100,20); l2.setBounds(450,80,70,20); t2.setBounds(520,80,100,20); l3.setBounds(450,110,70,20); t3.setBounds(520,110,100,20); l4.setBounds(450,140,70,20); t4.setBounds(520,140,100,20); l5.setBounds(450,170,70,20); t5.setBounds(520,170,100,20); l6.setBounds(450,200,100,20); t6.setBounds(550,200,100,20); b.setBounds(450,290,80,30); add(l1); add(l2); add(l3); add(l4); add(l5); add(l6); add(t1); add(t2); add(t3); add(t4); add(t5); add(t6); add(b); b.addActionListener(this); } public void actionPerformed(ActionEvent e){ float m1, m2,m3, m4,m5,percent; </pre>
--	---

```

m1= Float.parseFloat(t1.getText());
m2= Float.parseFloat(t2.getText());
m3= Float.parseFloat(t3.getText());
m4= Float.parseFloat(t4.getText());
m5= Float.parseFloat(t5.getText());

percent=((m1+m2+m3+m4+m5)*100)/500;

t6.setText(String.valueOf(percent));
repaint();
}
public void paint(Graphics g){

float p;
p= Float.parseFloat(t6.getText());

if(p> 50.0) {
    g.setColor(Color.YELLOW);
    g.fillOval(0,0,300,300);
    g.setColor(Color.BLACK);
    g.fillOval(80,75,30,30);
    g.fillOval(190,75,30,30);
    g.setColor(Color.black);
    g.fillArc (75,100,150,150,0,-180);
}
else {
    g.setColor(Color.YELLOW);
    g.fillOval(0,0,300,300);
    g.setColor(Color.BLACK );
    g.fillOval(80,75,30,30);
    g.fillOval(190,75,30,30);
    g.setColor(Color.black);
    g.drawArc(75,150,150,150,0,180);

}
}
}
/*
<applet code="smile.class" border="2" width="500"
height="500">
</applet>
*/

```

RESULT

The above program is successfully executed and obtained the output

OUTPUT

Applet Viewer: smile.class

Applet



MARK 1:

MARK 2:

MARK 3:

MARK 4:

MARK 5:

PERCENTAGE:

RESULT

Applet Viewer: smile.class

Applet



MARK 1:

MARK 2:

MARK 3:

MARK 4:

MARK 5:

PERCENTAGE:

RESULT

PROGRAM 4

AIM

Using 2D graphics commands in an Applet, construct a house. On mouse click event, change the colour of the door from blue to red

ALGORITHM

Step.1: Start.

Step.2: Define a class 'house' that extends Applet and implements MouseListener.

Step.3: Define methods to add MouseListener to the panel.

Step.4: Using getX() and getY() methods, get the coordinates of the door to repaint when the MousePressed event occurs.

Step.5: Stop.

PROGRAM CODE

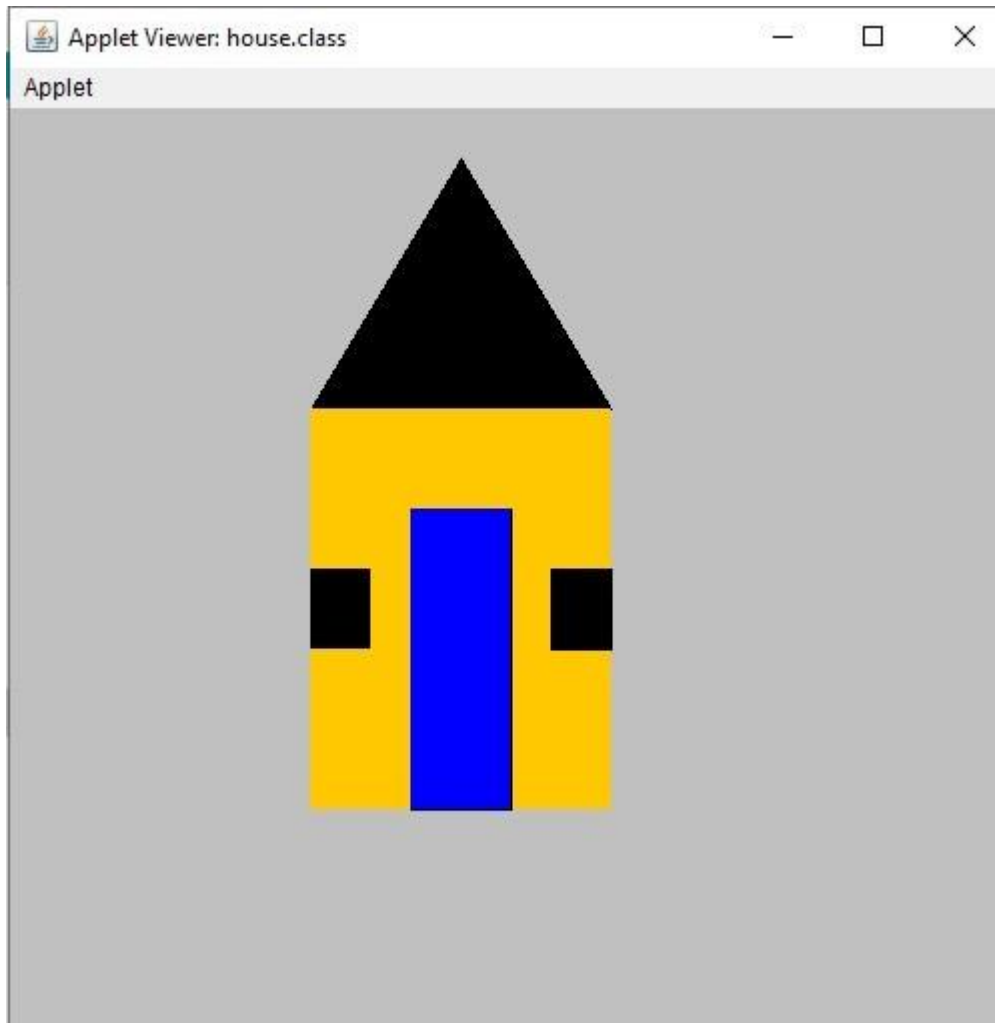
```
import java.awt.*;
import java.applet.*;
import java.awt.event.*;
/*
<applet code="house.class" width="500" height="700"
border="2">
</applet>
*/
public class house extends Applet implements MouseListener
{
    int a,b;
    public void init()
    {
        addMouseListener( this);
        setBackground(Color.lightGray);
    }
    public void paint(Graphics g)
    {
        int c[]={150,300,225};
        int d[]={150,150,25};
        g.drawPolygon(c,d,3);
        g.setColor(Color.black);
        g.fillPolygon(c,d,3);
        g.setColor(Color.orange);
        g.fillRect(150,150,150,200);    //House
        g.drawRect(150, 230,30,40);
        g.setColor(Color.black);
```

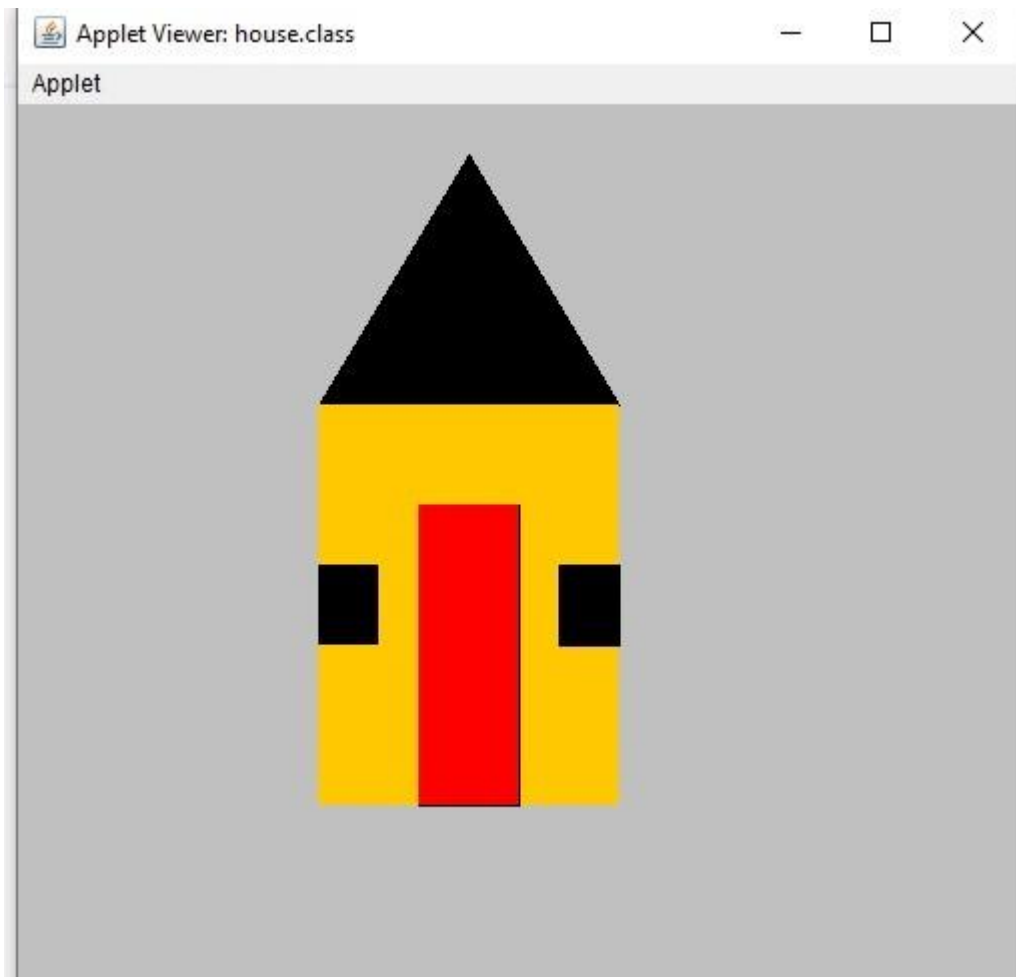
	<pre> g.fillRect(150,230,30,40); //Windows g.drawRect(270, 230,30,40); g.setColor(Color.black); g.fillRect(270,230,30,40); g.drawRect(200, 200,50,150); //Door g.setColor(Color.red); g.fillRect(200,200,50,150); if(a>200 && a<300 && b>200 && b<300) { g.setColor(Color.blue); g.fillRect(200, 200, 50, 150); } } public void mouseClicked(MouseEvent e) { } public void mouseEntered(MouseEvent e) { } } public void mouseExited(MouseEvent e) { } public void mousePressed(MouseEvent e) { a=e.getX(); b=e.getY(); repaint(); } public void mouseReleased(MouseEvent e) { } } </pre>
--	---

RESULT

The above program is successfully executed and obtained the output

OUTPUT





PROGRAM 5

AIM

Implement a simple calculator using AWT components

ALGORITHM

Step.1: Start.

Step.2: Define a class 'calcu' that extends Frame and implements ActionListener interface.

Step.3: Using TextField class object, construct the required no. of textfields wide enough to hold the values entered by the user.

Step.4: Using Label class object, construct and provide the appropriate labels.

Step.5: Using Button class object, construct labeled buttons that send the instances of ActionEvent.

Step.6: Call addActionListener() method to send events from the button to the new listener.

Step.7: Get the string values from textfields and then parse them as integers.

Step.8: Perform various methods to add, subtract, multiply and divide those integers.

Step.9: Stop.

PROGRAM CODE

```
import java.applet.*;
import java.awt.event.*;
import java.awt.*;
//<applet code="calcu" height=300 width=200></applet>
public class calcu extends Applet implements ActionListener
{
    TextField t;
    Button b[];
    Panel p;
    String c,
arr[]={"1","2","3","+","4","5","6","-","7","8","9","*","0",".", "=",
"/"};
    double n1,n2,ans;
    public void init()
    {
        t= new TextField(20);
        b=new Button[16];
        p= new Panel();
```

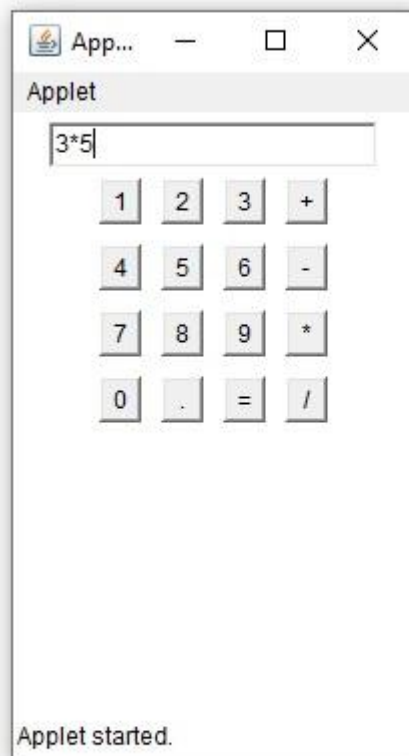

	<pre> for(int i=0;i<16;i++) b[i]=new Button(arr[i]); p.setLayout(new GridLayout(4, 4, 10, 10)); p.setSize(700,1500); } public void start() { add(t); for(int i=0;i<16;i++) { p.add(b[i]); b[i].addActionListener(this); } add(p); } public void actionPerformed(ActionEvent ae) { switch (ae.getActionCommand()) { case "1": insert("1"); break; case "2": insert("2"); break; case "3": insert("3"); break; case "4": insert("4"); break; case "5": insert("5"); break; case "6": insert("6"); break; case "7": insert("7"); break; case "8": insert("8"); break; case "9": insert("9"); break; case "0": insert("0"); break; case ".": insert("."); break; case "+": addition(); break; case "-": subtract(); break; case "*": multiply(); break; case "/": divide(); break; case "=": cal(); break; } } void insert(String n) { if (t.getText().equals("")) t.setText(n); else t.setText(t.getText()+n); } void addition() { n1=Double.parseDouble(t.getText()); </pre>
--	---

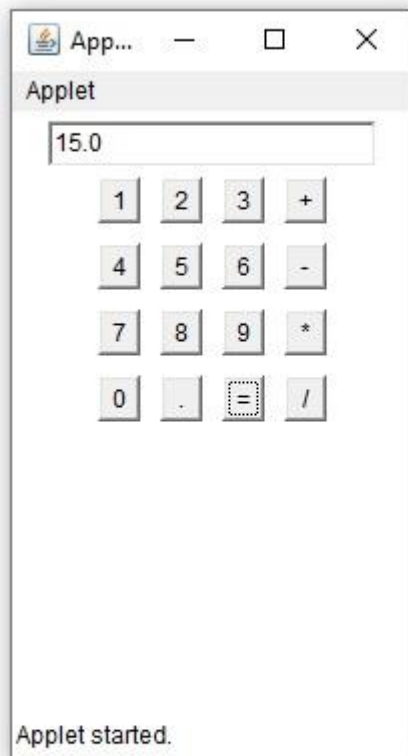
	<pre> t.setText(t.getText()+"+"); c="+"; } void subtract() { n1=Double.parseDouble(t.getText()); t.setText(t.getText()+"-"); c="-"; } void multiply() { n1=Double.parseDouble(t.getText()); t.setText(t.getText()+"*"); c="*"; } void divide() { n1=Double.parseDouble(t.getText()); t.setText(t.getText()+"/"); c="/"; } void cal() { n2=Double.parseDouble(t.getText().substring(t.getText().index Of(c)+1,t.getText().length())); if(c.equals("+")) { t.setText(Double.toString(n1+n2)); n1=n1+n2; } if(c.equals("-")) { t.setText(Double.toString(n1-n2)); n1=n1-n2; } if(c.equals("*")) { t.setText(Double.toString(n1*n2)); n1=n1*n2; } if(c.equals("/")) { t.setText(Double.toString(n1/n2)); n1=n1/n2; } } } </pre>
--	---

RESULT

The above program is successfully executed and obtained the output

OUTPUT





PROGRAM 6

AIM

Develop a program that has a Choice component which contains the names of shapes such as rectangle, triangle, square and circle. Draw the corresponding shapes for given parameters as per user's choice

ALGORITHM

Step.1: Start the program.

Step.2: Define a class 'choice' that extends Applet class and implements ItemListener interface.

Step.3: Declare a new constructor of the Choice class to create an empty Choice menu.

Step.4: Use add() method to include items in the menu.

Step.5: Using getSelectedItem() method, get the item chosen by the user from the menu and repaint accordingly.

Step.6: Stop the program.

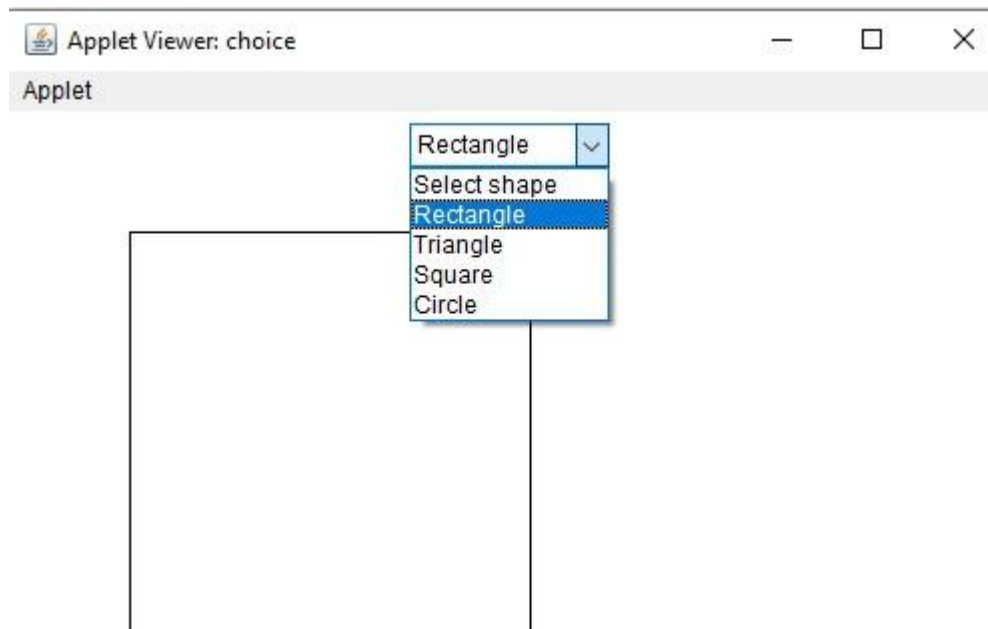
<u>PROGRAM CODE</u>
<pre>import java.applet.Applet; import java.awt.*; import java.awt.Graphics; import java.awt.event.*; //<applet code="choice" height=300 width=500></applet> public class choice extends Applet implements ItemListener { Choice choice; int rectX; int rectY; int rectWidth ; int rectHeight; String shape; int Selection; public void init() { // Create the choice and add some choices choice = new Choice(); choice.addItem("Select shape");</pre>

	<pre> choice.addItem("Rectangle"); choice.addItem("Triangle"); choice.addItem("Square"); choice.addItem("Circle"); add(choice); choice.addItemListener(this); } public void itemStateChanged (ItemEvent e) { // set new selection index Selection = choice.getSelectedIndex(); repaint(); } public void paint(Graphics g) { super.paint(g); if (Selection == 1) { g.drawRect(60,60,200,200); } if (Selection == 2) { g.drawLine(120, 130, 280, 130); g.drawLine(120, 130, 200, 65); g.drawLine(200, 65, 280, 130); } if (Selection == 3) { g.drawRect(150,150,100,100); } if (Selection ==4) { g.drawOval(150,150,140,140); } } } </pre>
--	--

RESULT

The above program is successfully executed and obtained the output

OUTPUT



Applet started

PROGRAM 7

AIM

Develop a program to handle all mouse events and window events

ALGORITHM

Step 1: Start

Step 2: Define a class q7 that extends Applet class and implements MouseListener interface

Step 3: Define methods to add MouseListener to the panel

Step 4: Using getX() and getY() methods, get the location (or movements) of mouse pointer on the panel. Use them to display the necessary message in the output.

Step 5: Define another class WindowEvents that extends Applet class and implements WindowListener interface.

Step 6: Define methods to add WindowListener to the panel

Step 7: Display the appropriate message in the output

Step 8: Stop

PROGRAM CODE

```
import java.awt.*;
import java.applet.*;
import java.awt.event.*;
/*<applet code="q7" width=300 height=300>
</applet>*/
public class q7 extends Applet implements
MouseListener,MouseMotionListener
{
int mx=0;
int my=0;
String msg="";
public void init()
{
addMouseListener(this);
addMouseMotionListener(this);
}
public void mouseClicked(MouseEvent me)
{
mx=20;
my=40;
msg="Mouse Clicked";
repaint();
}
```



```

}
public void mousePressed(MouseEvent me)
{
mx=30;
my=60;
msg="Mouse Pressed";
repaint();
}
public void mouseReleased(MouseEvent me)
{
mx=30;
my=60;
msg="Mouse Released";
repaint();
}
public void mouseEntered(MouseEvent me)
{
mx=40;
my=80;
msg="Mouse Entered";
repaint();
}
public void mouseExited(MouseEvent me)
{
mx=40;
my=80;
msg="Mouse Exited";
repaint();
}
public void mouseDragged(MouseEvent me)
{
mx=me.getX();
my=me.getY();
showStatus("Currently mouse dragged"+mx+" "+my);

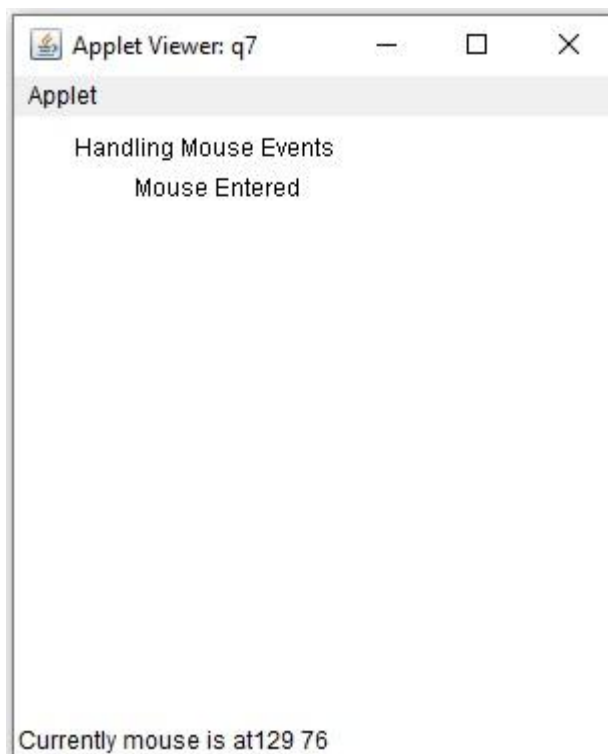
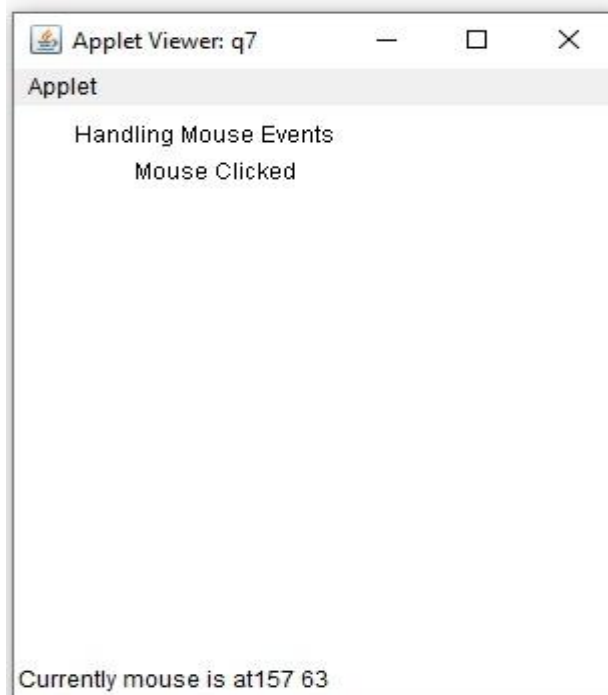
repaint(); }
public void mouseMoved(MouseEvent me)
{
mx=me.getX();
my=me.getY();
showStatus("Currently mouse is at"+mx+" "+my);
repaint();
}
public void paint(Graphics g)
{
g.drawString("Handling Mouse Events",30,20);
g.drawString(msg,60,40);
}}

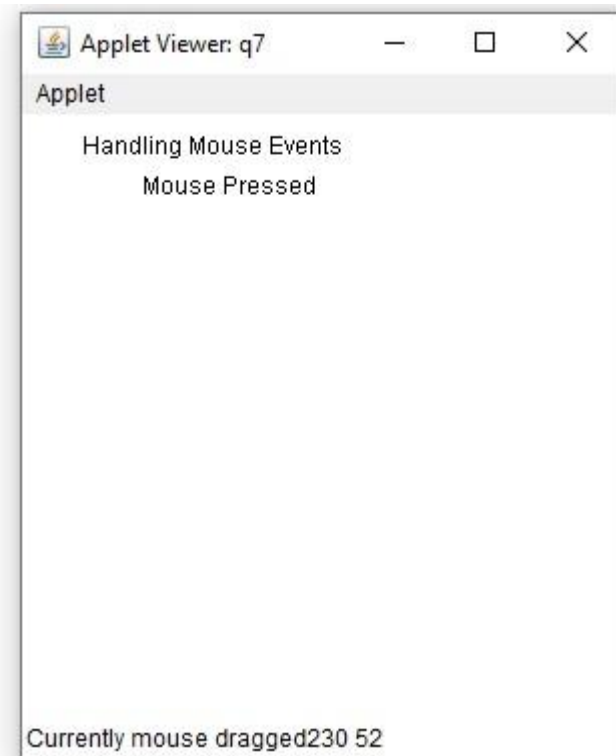
```

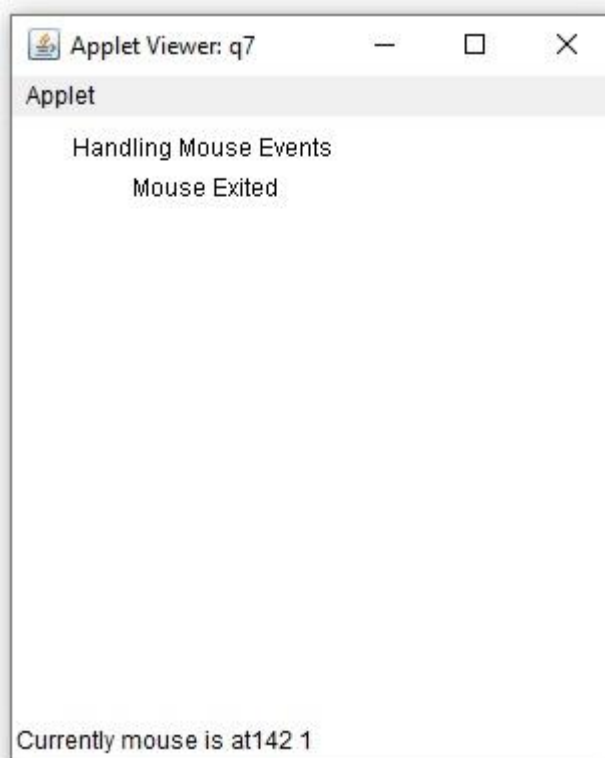
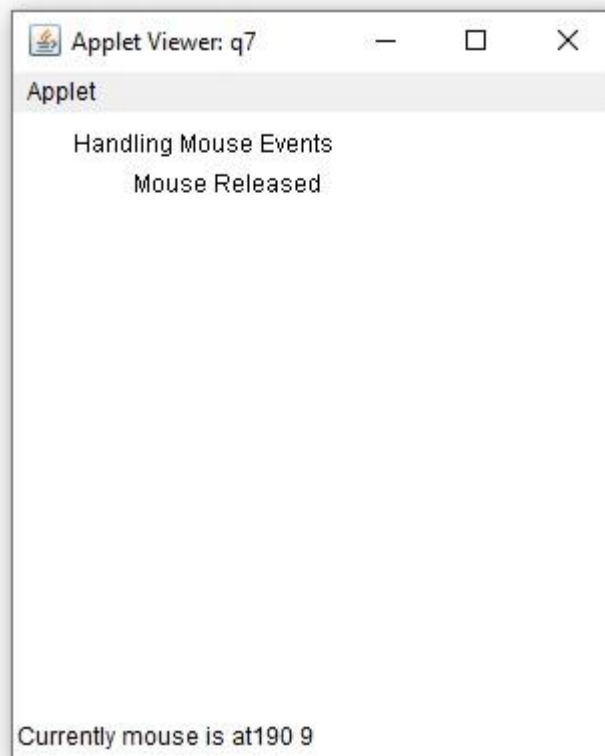
RESULT

The above program is successfully executed and obtained the output

OUTPUT







PROGRAM 8

AIM

Develop a program to handle Key events

ALGORITHM

Step.1: Start.

Step.2: Define a class q8 that extends Applet and implements KeyListener.

Step.3: Define methods to add KeyListener to the panel which will have the following methods:

void keyTyped(KeyEvent e) – Invoked when a key has been typed.

void keyPressed(KeyEvent e) - Invoked when a key has been pressed.

void keyReleased(KeyEvent e) - Invoked when a key has been released.

Step.4: Using getKeyChar(), get the unicode and character representation of the key pressed. Use them to display the necessary message in the output.

Step.5: Stop.

PROGRAM CODE

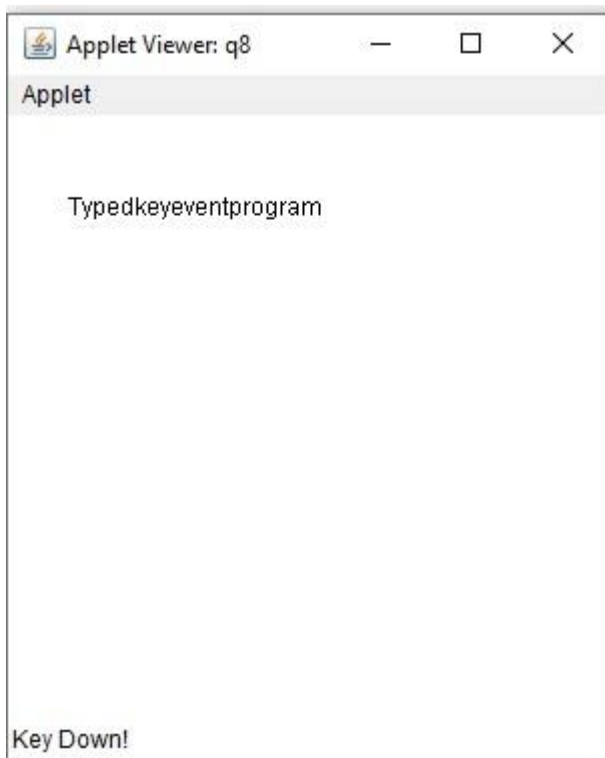
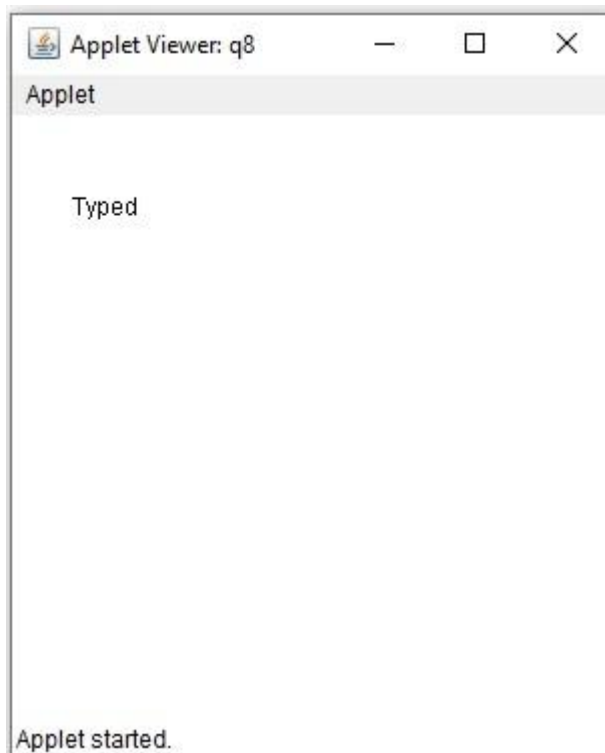
```
import java.awt.*;
import java.awt.event.*;
import java.applet.*;
/*<applet code="q8" width=300 height=300></applet>*/
public class q8 extends Applet implements KeyListener
{
    String msg="Typed";
    int x=30,y=50;
    public void init()
    {
        addKeyListener(this);
        requestFocus();
    }
    public void keyTyped(KeyEvent ke)
    {
        msg+=ke.getKeyChar();
        repaint();
    }
    public void keyReleased(KeyEvent ke)
    {
        showStatus("Key Up!");
    }
}
```

	<pre>public void keyPressed(KeyEvent ke) { showStatus("Key Down!"); } public void paint(Graphics G) { G.drawString(msg,x,y); } }</pre>
--	--

RESULT

The above program is successfully executed and obtained the output

OUTPUT



LABCYCLE 6

PROGRAM 1

AIM

Program to list the sub directories and files in a given directory and also search for a file name.

ALGORITHM

Step 1: Start

Step 2: Create a class named 'files' that implements FilenameFilter interface

Step 3: Create an object for the class File to initialize its constructor with the file source

Step 4: Using list(), get the names of all the files present in the directory

Step 5: Create an object for the FilenameFilter interface that contains the method Boolean accept (File dir, String fname) to test if a specified file should be included in the file list or not

Step 6: Filter accordingly and store the file names to the list

Step 7: Display the list

Step 8: Stop

PROGRAM CODE

```
import java.io.File;
import java.io.FilenameFilter;
public abstract class files implements FilenameFilter {
    public static void main(String[] args) {
        File f = new File("C:\\Users\\bibin\\Desktop\\co5");
        String[] flist = f.list();
        System.out.println("LISTING OUT THE FILES:\n");
        for(String str : flist) {
            System.out.println(str);
        }
        System.out.println("\nSEARCHING FOR FILENAMES  
STARTING WITH 'c':\n");

        FilenameFilter filter = new FilenameFilter() {
            public boolean accept(File dir, String fname) {
                return fname.startsWith("c");
            }
        };

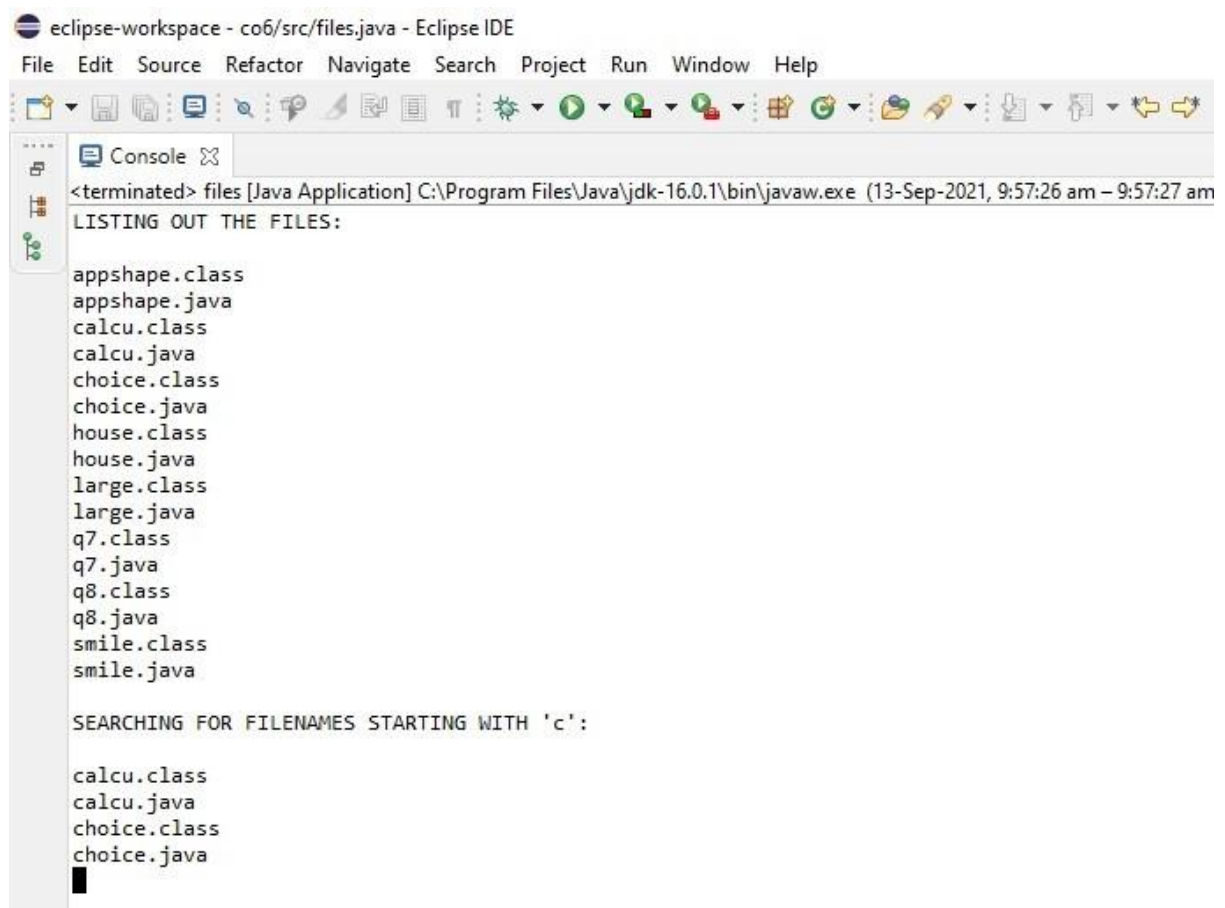
        String[] search = f.list(filter);
        if(search == null) {
```

	<pre> System.out.println("File does not exist. . "); } else { for(int i=0; i<search.length;i++) { String fn = search[i]; System.out.println(fn); } } } } } </pre>
--	--

RESULT

The above program is successfully executed and obtained the output.

OUTPUT



```

eclipse-workspace - co6/src/files.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
<terminated> files [Java Application] C:\Program Files\Java\jdk-16.0.1\bin\javaw.exe (13-Sep-2021, 9:57:26 am – 9:57:27 am)
LISTING OUT THE FILES:
appshape.class
appshape.java
calcu.class
calcu.java
choice.class
choice.java
house.class
house.java
large.class
large.java
q7.class
q7.java
q8.class
q8.java
smile.class
smile.java

SEARCHING FOR FILENAMES STARTING WITH 'c':
calcu.class
calcu.java
choice.class
choice.java

```

PROGRAM 2

AIM

Write a program to write a file, the read from the file and display the contents on the console.

ALGORITHM

Step 1: Start.

Step 2: Create a class named 'display'.

Step 3: Create an object of the class File to initialize its constructor with the file source.

Step 4: Create and use an object for the FileWriter class to write the file.

Step 5: Create and use an object for the BufferedReader class to read the stream of characters the specified file.

Step 6: Display the contents read from the file on the console.

Step 7: Stop.

PROGRAM CODE

```
import java.io.BufferedReader;
import java.io.FileReader;
import java.io.FileWriter;
import java.io.IOException;

public class display {

    public static void main(String[] args) {
        try {
            FileWriter writer = new FileWriter("file.txt", true);
            writer.write("OOPS LAB\n");
            writer.write("Done\n");
            writer.close();
            FileReader reader = new FileReader("file.txt");
            BufferedReader bufferedReader = new
            BufferedReader(reader);

            String line;
            System.out.println("Data read from the file");
            while ((line = bufferedReader.readLine()) != null) {
                System.out.println(line);
            }
            reader.close();

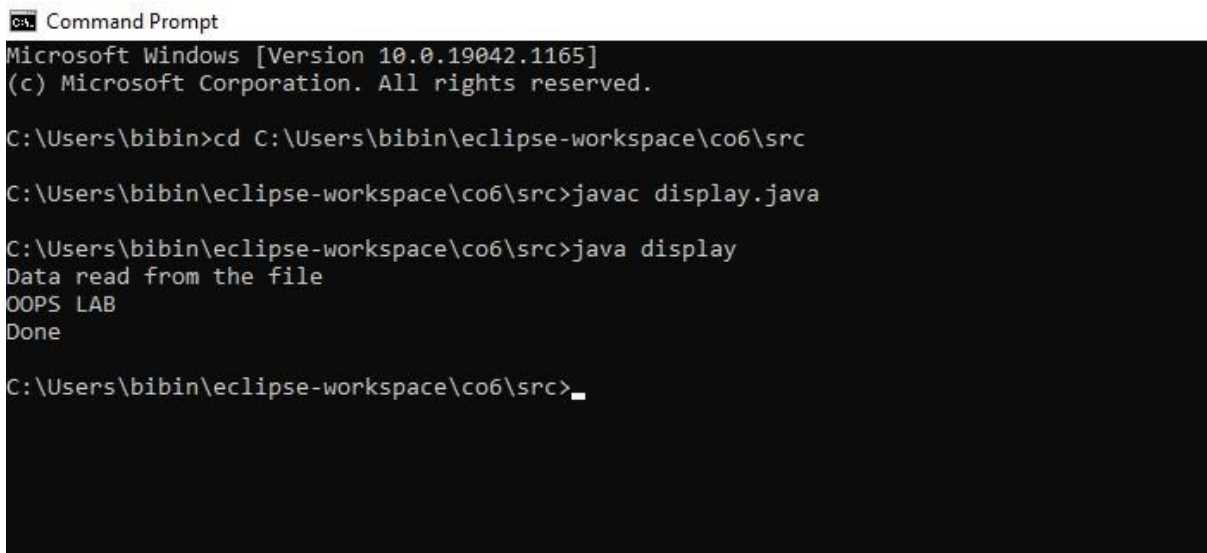
        } catch (IOException e) {
```

	<pre> System.out.println("Error Occurred!!!!!!"); } } }</pre>
--	--

RESULT

The above program is successfully executed and obtained the output.

OUTPUT



PROGRAM 3

AIM

Write a program to copy one file to another.

ALGORITHM

Step 1: Start.

Step 2: Create a class named 'filecopy'.

Step 3: Create and use an object for the BufferedReader class to read the stream of characters from the specified file.

Step 4: Create and use an object for the FileWriter class to write the stream of characters read by the BufferedReader, to the file. while ((s = br.readLine()) != null) { fw.write(s); }

Step 5: Stop

PROGRAM CODE

```
import java.io.*;
import java.io.BufferedReader;
import java.io.FileReader;
import java.io.FileWriter;
import java.io.IOException;

public class filecopy {

    public static void main(String[] args) {

        try {
            FileReader fr = new
FileReader("C:\\Users\\bibin\\eclipse-workspace\\co6\\src\\orgi
nal.txt");
            BufferedReader br = new
BufferedReader(fr);

            FileWriter fw = new
FileWriter("C:\\Users\\bibin\\eclipse-workspace\\co6\\src\\copy.
txt", true);

            String s;


            while ((s = br.readLine()) != null) { //
read a line
                fw.write(s+"\n"); // write to
output file
            }
            fw.flush();
        }
    }
}
```

	<pre> } br.close(); fw.close(); System.out.println("file copied"); } catch (IOException e) { e.printStackTrace(); } } </pre>
--	---

RESULT

The above program is successfully executed and obtained the output.


OUTPUT

 original - Notepad

File Edit Format View Help

hello,
oops lab



 copy - Notepad

File Edit Format View Help

hello,
oops lab



PROGRAM 4

AIM

Write a program that reads from a file having integers. Copy even numbers and odd numbers to separate files.

ALGORITHM

Step 1: Start

Step 2: Create a class named 'numbers'

Step 3: Create an object for the class File to initialize its constructor with the given file.

Step 4: Get user inputs via the console, for the integers to be inserted into the file.

Step 5: Using an object for the FileWriter class, write those integers into the file.

Step 6: Using objects for the FileOutputStream class, create two separate files to store even and odd integers respectively and copy the integers accordingly to separate files just created.

Step 7: Stop

PROGRAM CODE

```
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.IOException;


public class numbers {
    public static void main(String args[]) throws
IOException {
        FileInputStream fr = new
FileInputStream("C:\\Users\\bibin\\eclipse-workspace\\co6\\src
\\numbers.txt");
        FileOutputStream fw1 = new
FileOutputStream("even.txt");
        FileOutputStream fw2 = new
FileOutputStream("odd.txt");
        System.out.println("***Copied even numbers and
odd numbersto separate files***");
        int i;
        while((i=fr.read()) != -1)
        {
            if(i%2==0)
            fw1.write(i);
            else
            fw2.write(i);
```

	<pre>} fr.close(); fw1.close(); fw2.close(); } }</pre>
--	--

RESULT


The above program is successfully executed and obtained the output.

OUTPUT

 numbers - Notepad

File Edit Format View Help


```
2  
1  
3  
7  
8  
4  
6  
5
```

 odd - Notepad

File Edit Format View Help

```
█  
1  
3  
7
```

```
5
```


 even - Notepad

File Edit Format View Help

2

8

4

6

PROGRAM 5

AIM

Client server communication using Socket – TCP/IP

ALGORITHM

Step 1: Start

Step 2: To create the Client application, create an instance of ClientSocket class

Step 3: To create the Server application, create an instance of ServerSocket class

Step 4: Stop

PROGRAM CODE

client.java

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

public class client {
    public static void main(String args[]) throws Exception{
        try {
            Socket s = new Socket ("localhost", 2665);
            PrintWriter pw = new
PrintWriter(s.getOutputStream(), true);
            pw.println("Hello Server!! How are you?");

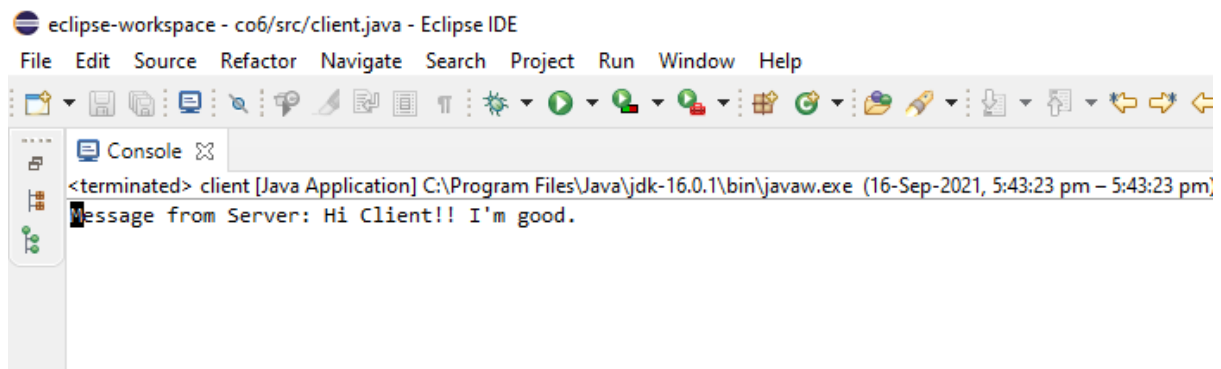
            //Client is reading from its InputStream
            InputStreamReader isr = new
InputStreamReader(s.getInputStream());
            BufferedReader br = new BufferedReader(isr);
            String str= br.readLine();
            System.out.println("Message from Server: "+str);
            pw.close();
            s.close();
        }
        catch(Exception e) {
            System.out.println("An error occured..." +e);
        }
    }
}
```

server.java	<pre> import java.net.*; import java.io.*; public class server { public static void main(String[] args) throws Exception { // TODO Auto-generated method stub try { ServerSocket ss = new ServerSocket(2665); System.out.println("Server is waiting for the client....."); Socket s = ss.accept(); System.out.println("CONNECTION ESTABLISHED !!!"); InputStreamReader isr = new InputStreamReader(s.getInputStream()); BufferedReader br = new BufferedReader(isr); String str = br.readLine(); System.out.println("Message from Client: "+str); //Server is responding through its OutputStream PrintWriter pw = new PrintWriter(s.getOutputStream(), true); pw.println("Hi Client!! I'm good."); pw.close(); } catch(Exception e) { System.out.println("An error occured.." +e); } } } </pre>
--------------------	--

RESULT

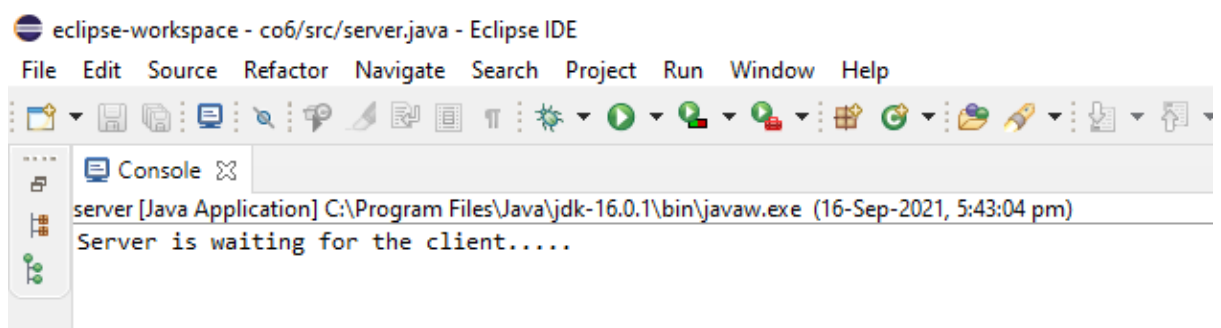
The above program is successfully executed and obtained the output.

OUTPUT



The screenshot shows the Eclipse IDE interface with the title bar "eclipse-workspace - co6/src/client.java - Eclipse IDE". The menu bar includes File, Edit, Source, Refactor, Navigate, Search, Project, Run, Window, and Help. The toolbar contains various icons for file operations, search, and development. The console window, titled "Console", displays the following output:

```
<terminated> client [Java Application] C:\Program Files\Java\jdk-16.0.1\bin\javaw.exe (16-Sep-2021, 5:43:23 pm - 5:43:23 pm)
Message from Server: Hi Client!! I'm good.
```



The screenshot shows the Eclipse IDE interface with the title bar "eclipse-workspace - co6/src/server.java - Eclipse IDE". The menu bar includes File, Edit, Source, Refactor, Navigate, Search, Project, Run, Window, and Help. The toolbar contains various icons for file operations, search, and development. The console window, titled "Console", displays the following output:

```
server [Java Application] C:\Program Files\Java\jdk-16.0.1\bin\javaw.exe (16-Sep-2021, 5:43:04 pm)
Server is waiting for the client.....
```

PROGRAM 6

AIM

Client Server communication using Datagram Socket – UDP

ALGORITHM

Step 1: Start

Step 2: Create the Client application

Step 3: Create the Server application

Step 4: Stop

PROGRAM CODE

ClientUdp.java	<pre>import java.io.*; import java.net.*; class Client{ public static void main(String[] args) throws IOException { DatagramSocket client= new DatagramSocket(); InetAddress add=InetAddress.getByName("localhost"); String str ="Hello...Server"; byte[] bufBytes = str.getBytes(); DatagramPacket datagramPacket=new DatagramPacket(bufBytes,bufBytes.length,add,4220); client.send(datagramPacket); client.close(); } }</pre>
ServerUdp.java	<pre>import java.io.*; import java.net.*; class Server { public static void main(String[] args) throws IOException { DatagramSocket server=new DatagramSocket(4220); byte[] buf=new byte[256]; DatagramPacket packet=new DatagramPacket(buf,buf.length);</pre>

	<pre>server.receive(packet); String reply =new String(packet.getData()); System.out.println("\n Client Says : "+reply); server.close(); } }</pre>
--	---

RESULT

The above program is successfully executed and obtained the output

OUTPUT

```
Microsoft Windows [Version 10.0.19041.264]
(c) 2020 Microsoft Corporation. All rights reserved.

E:\java\co6>javac ServerUdp.java

E:\java\co6>java ServerUdp

Client Says : Hello...Server
```

```
Microsoft Windows [Version 10.0.19041.264]
(c) 2020 Microsoft Corporation. All rights reserved.

E:\java\co6>javac Client.java

E:\java\co6>javac ClientUdp.java

E:\java\co6>java ClientUdp

E:\java\co6>
```