

OBJECT ORIENTED PROGRAMMING LAB

Co5 & Co6

Jomin K Mathew
TKM20MCA2021
Roll.no: 20MCA221

Program 1

10/05/2021

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.

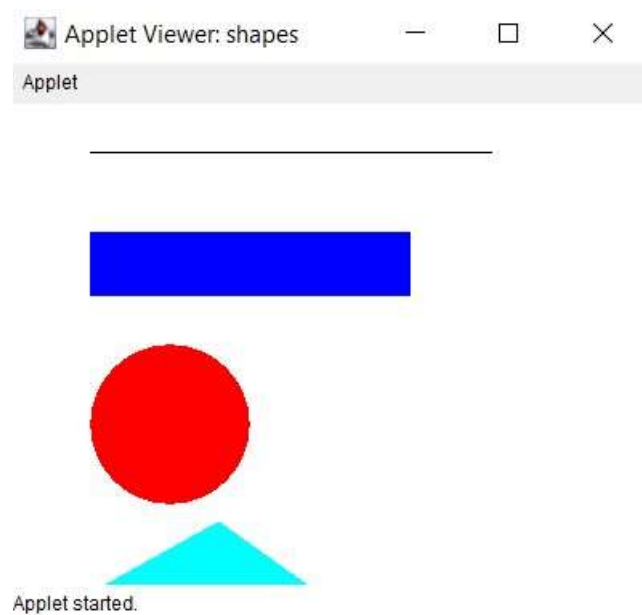
Step.4: Stop

Source Code:

```
import java.applet.Applet;
import java.awt.*;
public class shapes extends Applet{
public void paint(Graphics g){
g.drawLine(50,30, 300, 30);
g.setColor(Color.BLUE);
g.fillRect(50, 80, 200, 40);
g.setColor(Color.RED);
g.fillOval(50,150,100,100);
```

```
g.setColor(Color.CYAN);  
int x1[]={40, 130, 200, 50};  
int y1[]={310, 260, 310, 310};  
int n1=4;  
g.fillPolygon(x1, y1, n1);  
}  
}
```

Output:



Program 2

10/05/2021

Aim: Program to draw Circle, Rectangle, Line in Applet.

Algorithm:

Step.1: Start.

Step.2: Define a class 'largest' 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

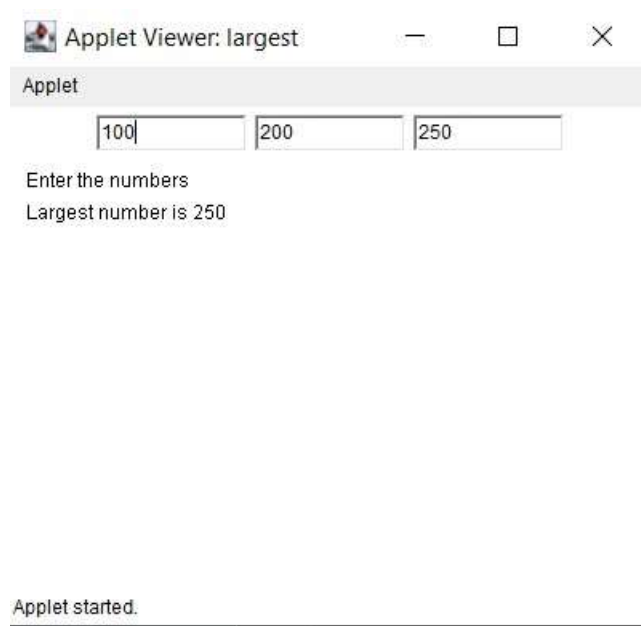
Source Code:

```
import java.awt.*;  
import java.applet.*;  
public class largest extends Applet  
{
```

```
TextField Txt1,Txt2,Txt3;
public void init(){
    Txt1 = new TextField(10);
    Txt2 = new TextField(10);
    Txt3 = new TextField(10);
    add(Txt1);
    add(Txt2);
    add(Txt3);
    Txt1.setText("num1");
    Txt2.setText("num2");
    Txt3.setText("num3");
}
public void paint(Graphics g){
    int a, b, c,result;
    String str;
    g.drawString("Enter the numbers",10,50);
    str=Txt1.getText();
    a=Integer.parseInt(str);
    str=Txt2.getText();
    b=Integer.parseInt(str);
    str=Txt3.getText();
    c=Integer.parseInt(str);
```

```
if (a>b) {  
    if (a>c)  
        result=a;  
    else  
        result=c;  
}  
else{  
    if (b>c)  
        result=b;  
    else  
        result=c;  
}  
g.drawString("Largest number is "+result,10,70);  
}  
public boolean action(Event e, Object o){  
    repaint();  
    return true;  
}  
}
```

Output:



Program 3

10/05/2021

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.

Source Code:

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

public class smile extends Applet implements ActionListener
{
    TextField t1,t2,t3,t4,t5,t6;
    Button b;
    Label l1,l2,l3,l4,l5,l6;

    public void init(){
        l1=new Label("mark1");
        //l1.setBounds(100,100,200,20);

        t1= new TextField(5);
        //t1.setBounds(100,50,200,20);

        l2=new Label("mark2");
        //l2.setBounds(100,130,100,30);

        t2= new TextField(5);
        //t2.setBounds(100,80,100,20);

        l3=new Label("mark3");
        //l3.setBounds(100,160,100,20);

        t3= new TextField(5);
        //t3.setBounds(100,120,100,20);
```

```
l4=new Label("mark4");
//l4.setBounds(100,130,100,30);
t4= new TextField(5);
//t4.setBounds(100,80,100,20);
l5=new Label("mark5");
//l5.setBounds(100,130,100,30);
t5= new TextField(5);
//t5.setBounds(100,80,100,20);
l6=new Label("result");
//l6.setBounds(100,200,100,20);
t6=new TextField(5);
t1.setBounds(210,40,100,20);
t2.setBounds(210,80,100,20);
t3.setBounds(210,120,100,20);
t4.setBounds(210,80,100,20);
t5.setBounds(210,120,100,20);
t6.setBounds(210,140,100,20);
l1.setBounds(100,40,100,20);
l2.setBounds(100,80,100,20);
l3.setBounds(100,120,100,20);
l4.setBounds(100,140,100,20);
l5.setBounds(100,120,100,20);
```

```
l6.setBounds(100,140,100,20);
b=new Button("find");
b.setBounds(230,150,60,50);
//t4.setBounds(100,200,100,20);
add(l1);
add(t1);
add(l2);
add(t2);
add(l3);
add(t3);
add(l4);
add(t4);
add(l5);
add(t5);
add(b);
b.addActionListener(this);
add(l6);
add(t6);
}

public void actionPerformed(ActionEvent e){
int mark1=0;
int mark2=0;
```

```
int mark3=0;
int mark4=0;
int mark5=0;
int total=0;
mark1= Integer.parseInt(t1.getText());
mark2= Integer.parseInt(t2.getText());
mark3= Integer.parseInt(t3.getText());
mark4= Integer.parseInt(t2.getText());
mark5= Integer.parseInt(t3.getText());
if(e.getSource()==b){
total=(mark1+mark2+mark3+mark4+mark5)/5;
t6.setText(String.valueOf(total));
}
}
```

@Override

```
public void paint(Graphics g){
int mark1=0;
int mark2=0;
int mark3=0;
int mark4=0;
int mark5=0;
int total=0;
```

```
mark1= Integer.parseInt(t1.getText());
mark2= Integer.parseInt(t2.getText());
mark3= Integer.parseInt(t3.getText());
mark4= Integer.parseInt(t2.getText());
mark5= Integer.parseInt(t3.getText());
total=(mark1+mark2+mark3+mark4+mark5)/5;
g.setColor(Color.yellow);
g.fillOval(20,20,150,150); // For face
g.setColor(Color.black);
g.fillOval(50,60,15,25); // Left Eye
g.fillOval(120,60,15,25); // Right Eye
int x[] = {95,85,106,95};
int y[] = {85,104,104,85};
g.drawPolygon(x, y, 4); // Nose
g.fillPolygon(x, y, 4);
// g.drawLine(50,126,60,116); // Smile arc1
//g.drawLine(128,115,139,126); // Smile arc2
if(total > 50){
g.drawArc(55,95,78,50,0,-180); // Smile
}
else
{
```

```
g.drawArc(55,120,78,50,0,180); // Smile  
}  
}}
```

Output:



Program 4

10/05/2021

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

Algorithm:

Step.1: Start.

Step.2: Define a class 'MouseClicked' 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.

Source Code:

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

public class MouseClick extends Applet implements
MouseListener
{
```

```
int a,b;

public void init()
{
    addMouseListener(this);
}

public void paint(Graphics gp)
{
    int [] x = {170, 280, 300,150};
    int [] y = {100, 100, 150, 150};
    gp.setColor(Color.cyan);
    gp.fillRect(150, 150, 150, 200); //House
    gp.drawRect(200, 200, 50, 150); // Door
    gp.setColor(Color.pink);
    gp.fillRect(200,200,50,150);
    gp.setColor(Color.gray);
    gp.fillPolygon(x,y,4);
    if(a>200 && a<300 && b>200 && b<300)
    {
        gp.setColor(Color.orange);
        gp.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)  
{  
}  
}
```

Output:



Applet started.

Program 5

10/05/2021

Aim: Implement a simple calculator using AWT components.

Algorithm:

Step.1: Start.

Step.2: Define a class 'calculator' 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.

Source Code:


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

class calculator implements ActionListener {
    Frame f = new Frame();
    Label l1 = new Label("1st Integer");
    Label l2 = new Label("2nd Integer");
    Label l3 = new Label("Result");
    TextField t1 = new TextField();
    TextField t2 = new TextField();
    TextField t3 = new TextField();
    Button b1 = new Button("Add");
    Button b2 = new Button("Sub");
    Button b3 = new Button("Mul");
    Button b4 = new Button("Div");
    calculator() {
        l1.setBounds(50, 100, 100, 20);
        l2.setBounds(50, 150, 100, 20);
        l3.setBounds(50, 200, 100, 20);
        t1.setBounds(200, 100, 100, 20);
        t2.setBounds(200, 150, 100, 20);
        t3.setBounds(200, 200, 100, 20);
```

```
b1.setBounds(50, 250, 50, 20);
b2.setBounds(110, 250, 50, 20);
b3.setBounds(170, 250, 50, 20);
b4.setBounds(230, 250, 50, 20);
f.add(l1);
f.add(l2);
f.add(l3);
f.add(t1);
f.add(t2);
f.add(t3);
f.add(b1);
f.add(b2);
f.add(b3);
f.add(b4);
b1.addActionListener(this);
b2.addActionListener(this);
b3.addActionListener(this);
b4.addActionListener(this);
f.setLayout(null);
f.setVisible(true);
f.setSize(500, 500);
}
```

```
public void actionPerformed(ActionEvent e) {  
    int i = Integer.parseInt(t1.getText());  
    int j = Integer.parseInt(t2.getText());  
    if (e.getSource() == b1) {  
        t3.setText(String.valueOf(i + j));  
    }  
    if (e.getSource() == b2) {  
        t3.setText(String.valueOf(i - j));  
    }  
    if (e.getSource() == b3) {  
        t3.setText(String.valueOf(i * j));  
    }  
    if (e.getSource() == b4) {  
        t3.setText(String.valueOf(i / j));  
    }  
}  
  
public static void main(String[] args)  
{  
    new calculator();  
}
```

Output:



—

□

×

1st Integer

16

2nd Integer

5

Result

21

Add

Sub

Mul

Div

Program 6

10/05/2021

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 'Shape' 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.

Source Code:

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

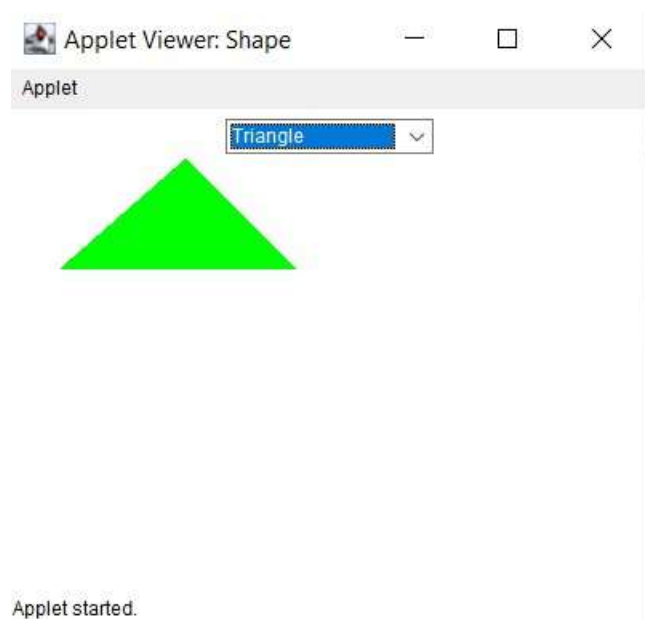


```
public class Shape extends Applet implements ItemListener
{
    Choice choice;
    int rectX;
    int rectY;
    int rectWidth ;
    int rectHeight;
    String shape;
    int Selection;
    public void init()
    {
        choice = new Choice();
        choice.addItem("Choose the shape");
        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.setColor(Color.BLUE);
        g.fillRect(70,70,200,130);
        g.drawRect(70,70,200,130);
    }
    if (Selection == 2)
    {
        int xArray[] = {110,180,30,110};
        int yArray[] = {30,100,100,30};
        g.setColor(Color.green);
        g.fillPolygon(xArray,yArray,4);
    }
    if (Selection == 3)
    {
```

```
g.setColor(Color.BLACK);  
g.fillRect(150,150,100,100);  
g.drawRect(150,150,100,100);  
}  
if (Selection ==4)  
{  
g.setColor(Color.red);  
g.fillOval(130,100,190,190);  
g.drawOval(130,100,190,190);  
}  
}  
}
```

Output:



Program 7

10/05/2021

Aim: Develop a program to handle all mouse events and window events.

Algorithm:

Step 1: Start

Step 2: Define a class 'MouseEvent' 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

Source Code:

```
import java.awt.*;  
import java.awt.event.MouseEvent;  
import java.awt.event.MouseListener;
```

```
import java.awt.event.WindowEvent;
import java.awt.event.WindowListener;

public class MouseWindowEvent extends Frame implements
MouseListener{

    Label l;

    MouseWindowEvent(){
        addMouseListener(this);
        l=new Label();
        l.setBounds(20,50,100,20);
        add(l);
        setSize(300,300);
        setLayout(null);
        setVisible(true);
    }

    public void mouseClicked(MouseEvent e) {
        l.setText("Mouse Clicked");
    }

    public void mouseEntered(MouseEvent e) {
        l.setText("Mouse Entered");
    }

    public void mouseExited(MouseEvent e) {
        l.setText("Mouse Exited");
    }
}
```

```

}

public void mousePressed(MouseEvent e) {
l.setText("Mouse Pressed");
}

public void mouseReleased(MouseEvent e) {
l.setText("Mouse Released");
}

public static void main(String[] args) {
new MouseListenerExample();
new WindowExample();
}
}

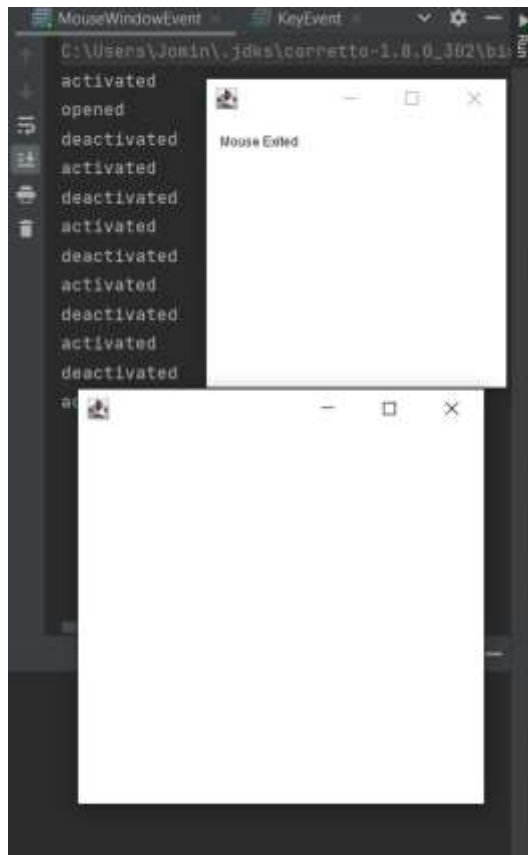
class WindowExample1 extends Frame implements
WindowListener{
WindowExample1(){
addWindowListener(this);
setSize(400,400);
setLayout(null);
setVisible(true);
}

public void windowActivated(WindowEvent arg0) {
System.out.println("activated");
}
}

```

```
}  
  
public void windowClosed(WindowEvent arg0) {  
    System.out.println("closed");  
}  
  
public void windowClosing(WindowEvent arg0) {  
    System.out.println("closing");  
    dispose();  
}  
  
public void windowDeactivated(WindowEvent arg0) {  
    System.out.println("deactivated");  
}  
  
public void windowDeiconified(WindowEvent arg0) {  
    System.out.println("deiconified");  
}  
  
public void windowIconified(WindowEvent arg0) {  
    System.out.println("iconified");  
}  
  
public void windowOpened(WindowEvent arg0) {  
    System.out.println("opened");  
}  
}
```

Output:



Program 8

10/05/2021

Aim: Develop a program to handle Key events.

Algorithm:

Step.1: Start.

Step.2: Define a class 'KeyEvents' 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.

Source Code:

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

```
import java.awt.event.KeyEvent;

public class KeyEvents extends Frame implements
KeyListener{
    Label l;
    TextArea area;
    KeyEvents(){
        l=new Label();
        l.setBounds(20,50,100,20);
        area=new TextArea();
        area.setBounds(20,80,300, 300);
        area.addKeyListener(this);
        add(l);add(area);
        setSize(400,400);
        setLayout(null);
        setVisible(true);
    }
    public void keyPressed(KeyEvent e) {
        l.setText("Key Pressed");
    }
    public void keyReleased(KeyEvent e) {
        l.setText("Key Released");
    }
}
```

```
public void keyTyped(KeyEvent e) {  
    l.setText("Key Typed");  
}  
  
public static void main(String[] args) {  
    new KeyListenerExample();  
}  
}
```

Output:



Program 9

10/05/2021

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 'Listfile' 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

Source Code:

```
package packoops;  
import java.io.File;
```

```

class Listfile {
public static void main(String[] args) {
File file = new File("C:\\Users\\jomin\\Desktop");
String[] fileList = file.list();
for(String str: fileList) {
System.out.println(str);
}
}
}
}

```

Output:

```

<terminated> Listfile [Java Application] C:\Users\jomin\p2\p
1.jpg
1.pdf
2.jpg
2.pdf
3.jpg
3.pdf
4.jpg
Among Us.url
Basic Commands
Course Outcome 1 & 2.pdf
desktop.ini
Doc1.pdf
JOMIN K MATHEW.pdf
Lab
New folder
PICO PARKClassic Edition.url
Ringtone.mp3
Ringtone2.mp3
Screenshot 2021-09-07 123003.jpg
Teacher-Day-010-1-24-june-2019.webp
Time Table.jpg
Tree
vi command
~$jomin.pdf
~$urse Outcome 2.docx
~WRL1175.tmp

```

Program 10

10/05/2021

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

Algorithm:

Step 1: Start.

Step 2: Create a class named 'write'.

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.

Source Code:

```
import java.io.File;
import java.io.IOException;
import java.io.FileWriter;
import java.util.Scanner
public class write
```

```
{  
public static void main(String[] args) {  
try {  
Scanner re=new Scanner(System.in);  
FileWriter obj = new FileWriter("D:\\programming\\file.txt");  
System.out.println("Write mode ON....." +  
""");  
String a= re.nextLine();  
obj.write(a);  
obj.close();  
System.out.println("Successfully written");  
File obk = new File("D:\\programming\\file.txt");  
System.out.println("Read mode ON....." +  
""");  
Scanner reada=new Scanner(obk);  
while(reada.hasNextLine()){  
String data= reada.nextLine();  
System.out.println(data);  
}  
reada.close();  
} catch (IOException e) {  
System.out.println("Error Occurred");
```

```
e.printStackTrace();
```

```
}
```

```
}
```

```
}
```

Output:

```
Write mode ON.....  
This is the program to perform file handling in java.....  
Successfully written  
Read mode ON.....  
This is the program to perform file handling in java.....  
  
Process finished with exit code 0  
|
```


Program 11

10/05/2021

Aim: Write a program to copy one file to another.

Algorithm:

Step 1: Start.

Step 2: Create a class named 'copy'.

Step 3: Create and use an object for the FileReader 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 FileReader, to the file.

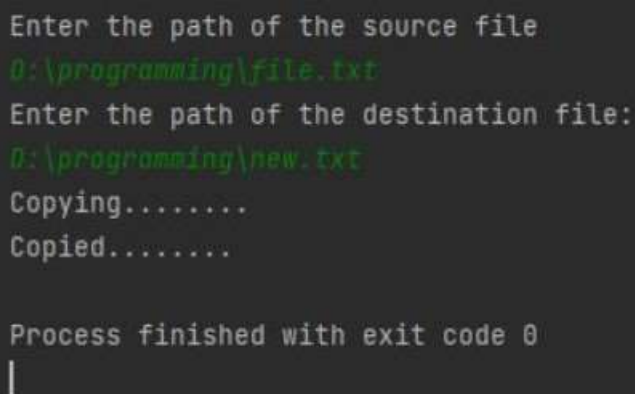
Step 5: Stop

Source Code:

```
import java.io.*;
import java.util.*;
public class copy {
    public static void main(String[] args) throws Exception{
        Scanner re = new Scanner(System.in);
        System.out.println("Enter the path of the source file");
        String sc=re.next();
        System.out.println("Enter the path of the destination file:");
```

```
String dc=re.next();
System.out.println("Copying.....");
FileReader ra= new FileReader(sc);
FileWriter wr =new FileWriter(dc,true);
int c;
while((c= ra.read()) !=-1){
wr.write(c);
}
System.out.println("Copied.....");
ra.close();
wr.close();
}
}
```

Output:



```
Enter the path of the source file
D:\programming\file.txt
Enter the path of the destination file:
D:\programming\new.txt
Copying.....
Copied.....

Process finished with exit code 0
|
```

Program 12

10/05/2021

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 'evenodd'

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

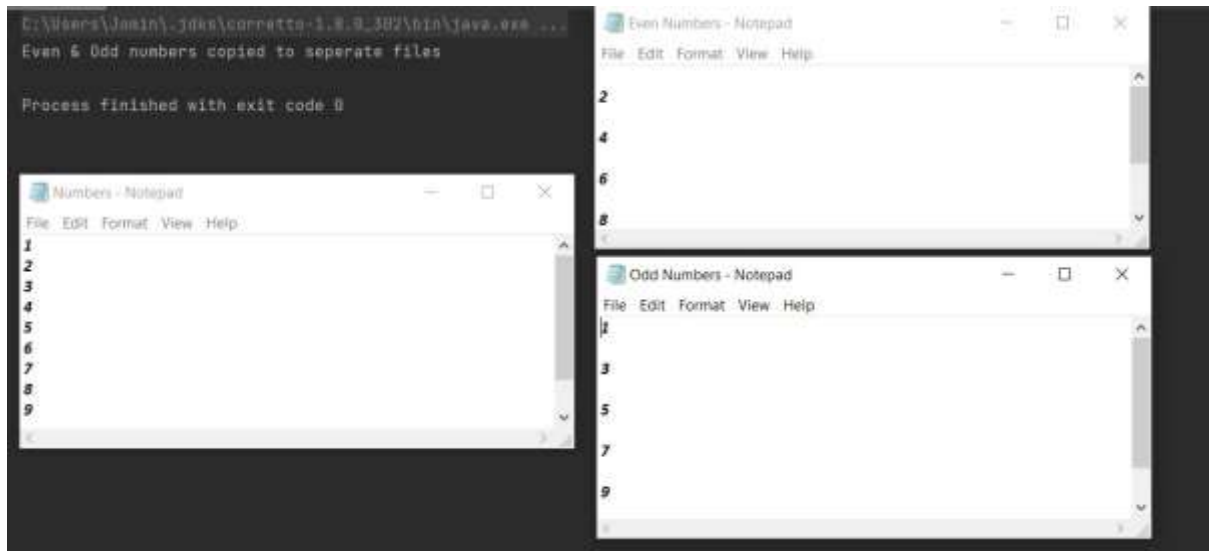
Source Code:

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

```
class evenodd {  
    public static void main(String[] args) throws IOException {  
        FileInputStream fr = new FileInputStream("D:\\MCA\\Sem  
        2\\oops lab\\Course Outcomes\\CO6\\prgrm4\\Numbers.txt");  
        FileOutputStream fw1 = new  
        FileOutputStream("D:\\MCA\\Sem 2\\oops lab\\Course  
        Outcomes\\CO6\\prgrm4\\Even Numbers.txt");  
        FileOutputStream fw2 = new  
        FileOutputStream("D:\\MCA\\Sem 2\\oops lab\\Course  
        Outcomes\\CO6\\prgrm4\\Odd Numbers.txt");  
        System.out.println("Even & Odd numbers copied to seperate  
        files");  
        int i;  
        while((i=fr.read()) != -1)  
        {  
            if(i%2==00)  
                fw1.write(i);  
            else  
                fw2.write(i);  
        }  
        fr.close();  
        fw1.close();  
        fw2.close();  
    }  
}
```

}

Output:



Program 13

10/05/2021

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

Source Code:

server.java

```
import java.net.*;
import java.io.*;
class server {
public static void main(String[] args) throws Exception {
try {
ServerSocket ss = new ServerSocket(2665);
System.out.println("Server is waiting .....");
```

```

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);
PrintWriter pw = new PrintWriter(s.getOutputStream(), true);
pw.println("Hello Client!!.");
pw.close();
}
catch(Exception e) {
System.out.println("An error occured.." +e);
}
}
}

```

client.java

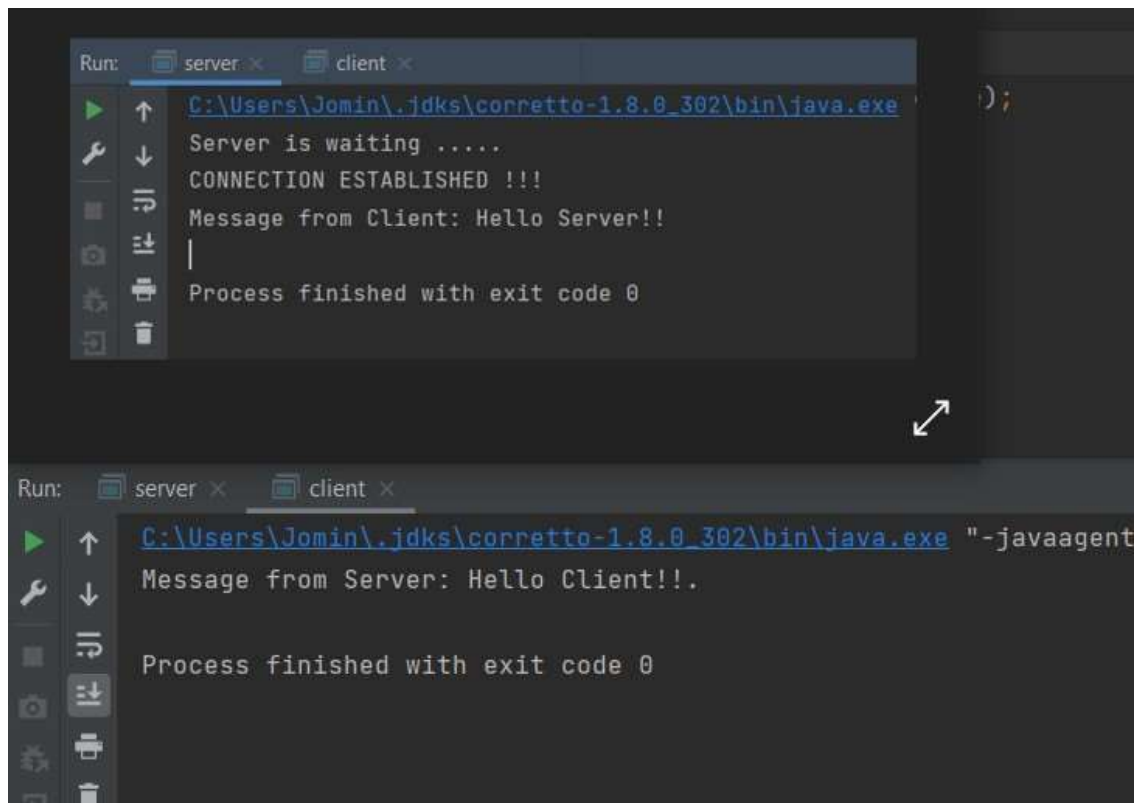
```

import java.net.*;
import java.io.*
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!!");
//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);
}
}
}
```


Output:



The image displays two screenshots of an IDE's Run console, showing the execution of a Java server and client program.

Top Screenshot (Server Output):

- Run: server x client x
- Command: `C:\Users\Jomin\.jdk\corretto-1.8.0_302\bin\java.exe`
- Output:
 - Server is waiting
 - CONNECTION ESTABLISHED !!!
 - Message from Client: Hello Server!!
 - Process finished with exit code 0

Bottom Screenshot (Client Output):

- Run: server x client x
- Command: `C:\Users\Jomin\.jdk\corretto-1.8.0_302\bin\java.exe -javaagent`
- Output:
 - Message from Server: Hello Client!!.
 - Process finished with exit code 0

Program 14

10/05/2021

Aim: Client Server communication using DatagramSocket – UD.

Algorithm:

Step 1: Start

Step 2: Create the Client application

Step 3: Create the Server application

Step 4: Stop

Source Code:

serverudp.java

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

public class serverudp {
    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);
        server.receive(packet);
```

```
String reply =new String(packet.getData());
System.out.println("\n Client Says : "+reply);
server.close();
}}
```

clientudp.java

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

public class clientudp {
    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();
    }
}
```

Output:

The image displays two terminal windows from an IDE. The top window, titled 'clientudp', shows the command `C:\Users\Jomin\jdk\corretto-1.8.0_302\bin\java.exe ...` and the output `Process finished with exit code 0`. The bottom window shows the command `C:\Users\Jomin\jdk\corretto-1.8.0_302\bin\java.exe ...` and the output of a network interaction: `Client Says : Hello` followed by `Server` and a long string of exclamation marks representing received data. Both windows conclude with `Process finished with exit code 0`. A double-headed arrow points from the top window to the bottom window, indicating a sequence of operations.