**Source Code:**

abstract class Shape

{

public abstract void Noofsides();

}

class Rectangle extends Shape

{

public void Noofsides()

{

System.out.println("No. Of Sides In Rectangle : 4");

}

}

class Triangle extends Shape

{

public void Noofsides()

{

System.out.println("No. Of Sides In Triangle : 3");

}

}

class Hexagon extends Shape

{

public void Noofsides()

{

System.out.println("No. Of Sides In Hexagon : 6");

}

}

class Polymorphism

{

public static void main(String[] args)

{

Rectangle r = new Rectangle();

Triangle t = new Triangle();

Hexagon h = new Hexagon();

r.Noofsides();

t.Noofsides();

h.Noofsides();

}

}

**Text

Description automatically generatedOutput :**

**Source Code:**

public class GarbageCollector

{

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

{

Test g1 = new Test();

Test g2 = new Test();

System.out.println(g1);

System.out.println(g2);

g1 = null;

System.gc();

g2 = null;

Runtime.getRuntime().gc();

}

protected void finalize() throws Throwable

{

System.out.println("Garbage Collector is called");

System.out.println("Object is Garbage Collected :" +this);

}

}

**Output:**

Graphical user interface, text, application

Description automatically generated

**Source Code:**

import java.io.FileReader;

import java.io.FileWriter;

import java.io.IOException;

public class fileHandling

{

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

{

int ch;

FileReader fr = new FileReader("Sample.txt");

FileWriter fw = new FileWriter("NewSample.txt");

while ((ch = fr.read())!=-1)

{

System.out.print((char)ch);

fw.write((char)ch);

}

fr.close();

fw.close();

}

}

**Output:**

Text

Description automatically generatedText

Description automatically generatedA picture containing shape

Description automatically generated

**Source Code:**

import java.util.Scanner;

import java.util.StringTokenizer;

public class Tokenizer

{

public static void main(String args[])

{

int n,sum=0;

Scanner s = new Scanner(System.in);

System.out.print("String Tokenizer\nEnter The String : ");

String t=s.nextLine();

StringTokenizer st = new StringTokenizer(t," ");

while (st.hasMoreTokens())

{

String temp = st.nextToken();

n = Integer.parseInt(temp);

sum = sum+n;

System.out.println(n);

}

System.out.print("Sum ="+sum);

s.close();

}

}

**Output:**

Text

Description automatically generated

**Source Code:**

//Usage Of Try, Catch,throws and finally

import java.util.Scanner;

class Test

{

public static void main(String args[])

{

Scanner sc = new Scanner(System.in);

try

{

System.out.println("Program To Perform Devision:");

System.out.print("Enter Number 1: ");

int a = sc.nextInt();

System.out.print("Enter Number 2: ");

int b = sc.nextInt();

int c = a/b;

System.out.println("Result : "+c);

}

catch ( ArithmeticException e )

{

System.out.println(e.getMessage());

}

finally

{

System.out.println("End Of Operation");

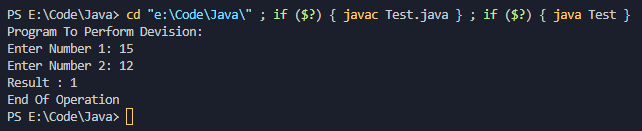
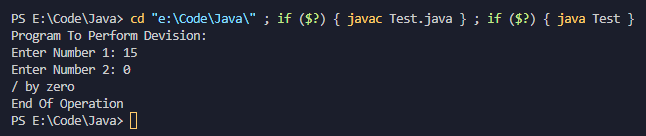
}

sc.close();

}

}

**Output:**



**Source Code:**

import java.util.Random;

class EvenThread extends Thread

{

private int num;

public EvenThread(int num)

{

this.num=num;

}

public void run()

{

System.out.println("Square Of "+num+" = "+num\*num);

}

}

class OddThread extends Thread

{

private int num;

public OddThread(int num)

{

this.num=num;

}

public void run()

{

System.out.println("Cube Of "+num+" = "+num\*num\*num);

}

}

class RandomThread extends Thread

{

public void run()

{

Random r = new Random();

for (int i =0;i<10;i++)

{

int num = r.nextInt(100);

if (num%2 == 0)

{

new EvenThread(num).start();

}

else

{

new OddThread(num).start();

}

}

}

}

class MultiThreading

{

public static void main(String args[])

{

RandomThread r = new RandomThread();

r.start();

}

}

**Output:**

Text

Description automatically generated

**Source Code:**

class Display

{

public synchronized void print(String msg)

{

System.out.print("["+msg);

try

{

Thread.sleep(1000);

}

catch (Exception e)

{

System.out.print(e.getMessage());

}

System.out.print("]");

}

}

class SyncThread extends Thread

{

private Display d;

private String msg;

public SyncThread(Display d,String msg)

{

this.d=d;

this.msg=msg;

}

public void run()

{

d.print(msg);

}

}

class Test\_Synchronization

{

public static void main(String args[])

{

Display d = new Display();

SyncThread t1 = new SyncThread(d,"Hello");

SyncThread t2 = new SyncThread(d,"World");

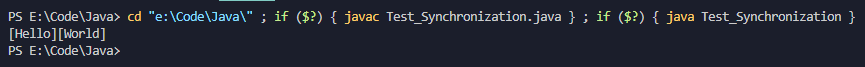
t1.start();

t2.start();

}

}

**Output:**



**Source Code:**

import javax.swing.\*;

import java.awt.event.\*;

class Calculator extends JFrame implements ActionListener

{

private JTextField t1;private JButton b1;private JButton b2;private JButton b3;private JButton b4;

private JButton b5;private JButton b6;private JButton b7;private JButton b8;private JButton b9;

private JButton b10;private JButton b11;private JButton b12;private JButton b13;private JButton b14;

private JButton b15;private JButton b16;private JButton b17;private Integer res;private String operation;

public Calculator()

{

setLayout(null);setSize(680,480);t1 = new JTextField();t1.setBounds(100,100,200,30);

b1 = new JButton("1");b1.setBounds(100,140,50,30);

b2 = new JButton("2");b2.setBounds(150,140,50,30);

b3 = new JButton("3");b3.setBounds(200,140,50,30);

b4 = new JButton("+");b4.setBounds(250,140,50,30);

// Third Row

b5 = new JButton("4");b5.setBounds(100,170,50,30);

b6 = new JButton("5");b6.setBounds(150,170,50,30);

b7 = new JButton("6");b7.setBounds(200,170,50,30);

b8 = new JButton("-");b8.setBounds(250,170,50,30);

// Fourth Row

b9 = new JButton("7");b9.setBounds(100,200,50,30);

b10 = new JButton("8");b10.setBounds(150,200,50,30);

b11 = new JButton("9");b11.setBounds(200,200,50,30);

b12 = new JButton("\*");b12.setBounds(250,200,50,30);

// Fourth Row

b13 = new JButton("/");b13.setBounds(100,230,50,30);

b14 = new JButton("%");b14.setBounds(150,230,50,30);

b15 = new JButton("=");b15.setBounds(200,230,50,30);

b16 = new JButton("C");b16.setBounds(250,230,50,30);

b17 = new JButton("0");b17.setBounds(100,260,50,30);

add(t1);add(b1);add(b2);add(b3);add(b4);add(b5);add(b6);add(b7);add(b8);

add(b9);add(b10);add(b11);add(b12);add(b13);add(b14);add(b15);add(b16);add(b17);

b1.addActionListener(this);b2.addActionListener(this);.addActionListener(this);

b4.addActionListener(this);b5.addActionListener(this);b6.addActionListener(this);

b7.addActionListener(this);b8.addActionListener(this); b9.addActionListener(this);

b10.addActionListener(this);b11.addActionListener(this);b12.addActionListener(this);

b13.addActionListener(this);b14.addActionListener(this);b15.addActionListener(this);

b16.addActionListener(this);b17.addActionListener(this);

}

public void doAction(String op)

{

String out ="";

if(operation==null)

{

operation = op;

res=Integer.parseInt(t1.getText());

t1.setText("");

}

else

{

switch(operation)

{

case "+":

out += res.toString() + "+" + t1.getText();

res=res+Integer.parseInt(t1.getText());

break;

case "-":

out += res.toString() + "-" + t1.getText();

res=res-Integer.parseInt(t1.getText());

break;

case "/":

try

{

if(t1.getText().equals("0"))

{

throw new ArithmeticException("Divide By zero");

}

out += res.toString() + "/" + t1.getText();

res = res/Integer.parseInt(t1.getText());

}

catch(ArithmeticException e)

{

t1.setText(e.getMessage());

operation = null;

res = 0;

}

break;

case "\*":

out += res.toString() + "\*" + t1.getText();

res=res\*Integer.parseInt(t1.getText());

break;

case "%":

out += res.toString() + "%" + t1.getText();

res=res%Integer.parseInt(t1.getText());

break;

}

if(op.equals("="))

{

out += "=" + res.toString();

t1.setText(out);

res = 0;

operation = null;

}

}

else

{

operation = op;

t1.setText("");

}

}

}

public void actionPerformed(ActionEvent e)

{

if(e.getSource() == b1)

t1.setText(t1.getText()+"1");

else if(e.getSource() == b2)

t1.setText(t1.getText()+"2");

else if(e.getSource() == b3)

t1.setText(t1.getText()+"3");

else if(e.getSource() == b5)

t1.setText(t1.getText()+"4");

else if(e.getSource() == b6)

t1.setText(t1.getText()+"5");

else if(e.getSource() == b7)

t1.setText(t1.getText()+"6");

else if(e.getSource() == b9)

t1.setText(t1.getText()+"7");

else if(e.getSource() == b10)

t1.setText(t1.getText()+"8");

else if(e.getSource() == b11)

t1.setText(t1.getText()+"9");

else if(e.getSource() == b17)

t1.setText(t1.getText()+"0");

else if(e.getSource() == b16)

{

t1.setText("");

res=0;

operation=null;

}

else if(e.getSource() == b4)

doAction("+");

else if(e.getSource() == b8)

doAction("-");

else if(e.getSource() == b12)

doAction("\*");

else if(e.getSource() == b13)

doAction("/");

else if(e.getSource() == b14)

doAction("%");

else if(e.getSource() == b15)

doAction("=");

}

public static void main(String [] args){

new Calculator().setVisible(true);

}

}

**Output:**

Graphical user interface

Description automatically generated with medium confidenceGraphical user interface, table

Description automatically generated

**Source Code:**

import java.applet.\*;

import java.awt.\*;

import java.awt.event.\*;

/\*<applet code="Signals" width=400 heigjt=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);

}

}

}

public class QuickSortOnStrings

{

String names[];

int length;

public static void main(String[]args)

{

QuickSortOnStrings obj=new QuickSortOnStrings();

String stringsList[]={"Raja","Gouthu","Rani","Gouthami","Honey","Heyaansh","Hello"};

obj.sort(stringsList);

for(String i:stringsList)

{

System.out.println(i);

}

}

void sort(String array[])

{

if(array==null||array.length==0)

{

return ;

}

this.names=array;

this.length=array.length;

quickSort(0,length-1);

}

void quickSort(int lowerIndex,int higherIndex)

{

int i=lowerIndex;

int j=higherIndex;

String pivot=this.names[lowerIndex+(higherIndex-lowerIndex)/2];

while(i<=j)

{

while(this.names[i].compareToIgnoreCase(pivot)<0)

{

i++;

}

while(this.names[j].compareToIgnoreCase(pivot)>0)

{

j--;

}

if(i<=j)

{

exchangeNames(i,j);

i++;

j--;

}

if (lowerIndex<j)

{

quickSort(lowerIndex,j);

}

if (i<higherIndex)

{

quickSort(i,higherIndex);

}

}

}

void exchangeNames(int i,int j)

{

String temp=this.names[i];

this.names[i]=this.names[j];

this.names[j]=temp;

}

}

**Output:**

Graphical user interface, text, application

Description automatically generatedA picture containing bubble chart

Description automatically generatedA picture containing bubble chart

Description automatically generatedA picture containing diagram

Description automatically generated

**Source Code:**

import java.util.Scanner;

class DoublyLinkedList

{

class node

{

int item;

node previous;

node next;

node(int item)

{

this.item=item;

}

}

node head,tail=null;

public void addNode(int item)

{

node newNode=new node(item);

if(head==null)

{

head=tail=newNode;

head.previous=null;

tail.next=null;

}

else

{

tail.next=newNode;

newNode.previous=tail;

tail=newNode;

tail.next=null;

}

}

public void deleteNode(int data)

{

node temp=head;

node prev=null;

if(head==null)

{

System.out.println("list is empty");

}

else

{

while(temp.item!=data)

{

prev=temp;

temp=temp.next;

}

prev.next=temp.next;

}

}

public void Display()

{

node Current=head;

if(head==null)

{

System.out.println("list is empty");

}

else

{

System.out.println("Elements of list are");

while(Current!=null)

{

System.out.println(Current.item+"");

Current=Current.next;

}

}

}

}

public class DLL

{

public static void main(String[] args)

{

Scanner sc = new Scanner(System.in);

DoublyLinkedList dL=new DoublyLinkedList();

dL.addNode(10);dL.addNode(20);dL.addNode(30);dL.addNode(40);dL.addNode(50);

dL.Display();

System.out.print("Element To Delete : ");

int data = sc.nextInt();

dL.deleteNode(data);dL.Display();

}

}

**Output:**

Graphical user interface, text

Description automatically generated

**Source Code:**

public class QuickSortOnStrings

{

String names[];

int length;

public static void main(String[]args)

{

QuickSortOnStrings obj=new QuickSortOnStrings();

String stringsList[]={"Raja","Gouthu","Rani","Gouthami","Honey","Heyaansh","Hello"};

obj.sort(stringsList);

for(String i:stringsList)

{

System.out.println(i);

}

}

void sort(String array[])

{

if(array==null||array.length==0)

{

return ;

}

this.names=array;

this.length=array.length;

quickSort(0,length-1);

}

void quickSort(int lowerIndex,int higherIndex)

{

int i=lowerIndex;

int j=higherIndex;

String pivot=this.names[lowerIndex+(higherIndex-lowerIndex)/2];

while(i<=j)

{

while(this.names[i].compareToIgnoreCase(pivot)<0)

{

i++;

}

while(this.names[j].compareToIgnoreCase(pivot)>0)

{

j--;

}

if(i<=j)

{

exchangeNames(i,j);

i++;

j--;

}

if (lowerIndex<j)

{

quickSort(lowerIndex,j);

}

if (i<higherIndex)

{

quickSort(i,higherIndex);

}

}

}

void exchangeNames(int i,int j)

{

String temp=this.names[i];

this.names[i]=this.names[j];

this.names[j]=temp;

}

}

**Output:**

Text

Description automatically generated