**Suppose that a table named Table.txt is stored in a text file. The first line in the file is theheader, and the remaining lines correspond to rows in the table. The elements are separatedbycommas. Writea javaprogram todisplay thetableusingLabelsin GridLayout.**

import java.awt.\*;

importjava.awt.event.\*;

import javax.swing.\*;

import java.util.\*;

import java.io.\*;

publicclass Table1 extends JFrame

{

int i=0;

int j=0,k=0;

Object data[][]=new Object[5][4];

Object list[][]=new Object[5][4];

JButton save;

JTable table1; FileInputStream fis;

DataInputStream dis;

public Table1()

{

//String d= "pavan kumar";

Container con=getContentPane(); con.setLayout(new BorderLayout());

final String[] colHeads={"Name","Roll Number","Department","Percentage"};

try

{

String s=JOptionPane.*showInputDialog*("Enter the File name present in the current directory");

FileInputStream fis=new FileInputStream(s);

//DataInputStream dis = new DataInputStream(fis);

BufferedReader dis=new BufferedReader(new InputStreamReader(fis));

while ((d=dis.readLine())!=null)

{

StringTokenizer st1=new StringTokenizer("");

while (st1.hasMoreTokens())

{

for (j=0;j<4;j++)

{

data[i][j]=st1.nextToken();

System.*out*.println(data[i][j]);

}

i++;

}

System.*out*.println (" ");

dis.close();

}}

catch (Exception e)

{

System.*out*.println ("Exception raised" +e.toString());

}

table1=new JTable(data,colHeads);

int v=ScrollPaneConstants.*VERTICAL\_SCROLLBAR\_AS\_NEEDED*;

int h=ScrollPaneConstants.*HORIZONTAL\_SCROLLBAR\_AS\_NEEDED*;

JScrollPane scroll=new JScrollPane(table1,v,h); con.add(scroll,BorderLayout.*CENTER*);

}

publicstaticvoid main(String args[])

{

Table1 t=new Table1();

t.setBackground(Color.*green*);

t.setTitle("Display Data");

t.setSize(500,300);

t.setVisible(true);

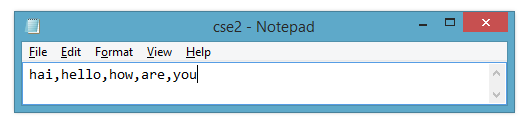
t.setDefaultCloseOperation(JFrame.*EXIT\_ON\_CLOSE*);

}

}

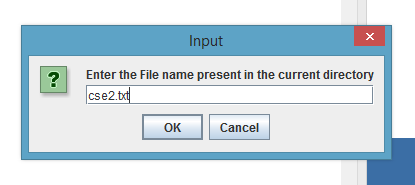
**Output:-**

**NOTE: Before going to compilation we have to create the one text file like below**

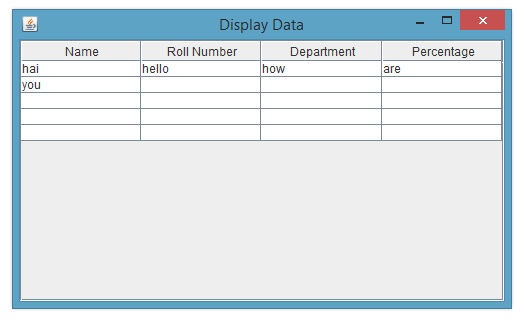


**Compile:javac Table1.java**

**Run:java Table1**



**Here when we submit the ok button it will read the data from cse2.txt file as shown above figure**

****

**Write a Java programthat loads names and phone numbers froma text file where thedataisorganizedasonelineperrecordandeachfieldinarecordareseparatedbyatab(\t).It takes a name or phone number as input and prints the corresponding other value from thehashtable(hint: usehashtables).**

import java.io.BufferedReader;importjava.io.File;

import java.io.FileNotFoundException;importjava.io.FileReader;

import java.io.IOException;import java.util.Hashtable;importjava.util.Iterator;importjava.util.Set;

publicclassHashTab

{

publicstaticvoidmain(String[]args)

{

HashTabprog11=newHashTab();

Hashtable<String, String>hashData = prog11.readFromFile("HashTab.txt");System.out.println("File data into Hashtable:\n"+hashData);prog11.printTheData(hashData, "vbit");

prog11.printTheData(hashData,"123");

prog11.printTheData(hashData, " ");

}

privatevoidprintTheData(Hashtable<String,String>hashData,Stringinput)

{

String output = null;if(hashData!=null)

{

Set<String> keys = hashData.keySet();if(keys.contains(input))

{

}

else

{

output=hashData.get(input);

Iterator<String> iterator = keys.iterator();while(iterator.hasNext()){

String key= iterator.next();

String value = hashData.get(key);if(value.equals(input))

{

output = key;break;

}

}

}

}

System.out.println("Input given:"+input);if(output!=null)

{

System.out.println("DatafoundinHashTable:"+output);

}

else

{

System.out.println("DatanotfoundinHashTable");

}

}

private Hashtable<String, String>readFromFile(String fileName)

{

Hashtable<String,String>hashData=newHashtable<String,String>();

try

{

Filef=newFile("D:\\java\\"+fileName);

BufferedReader br = new BufferedReader(new FileReader(f));Stringline=null;

while((line = br.readLine()) != null)

{

String[] details = line.split("\t");hashData.put(details[0],details[1]);

}

}

catch(FileNotFoundExceptione)

{

e.printStackTrace();

}

catch (IOException e)

{

e.printStackTrace();

}

returnhashData;

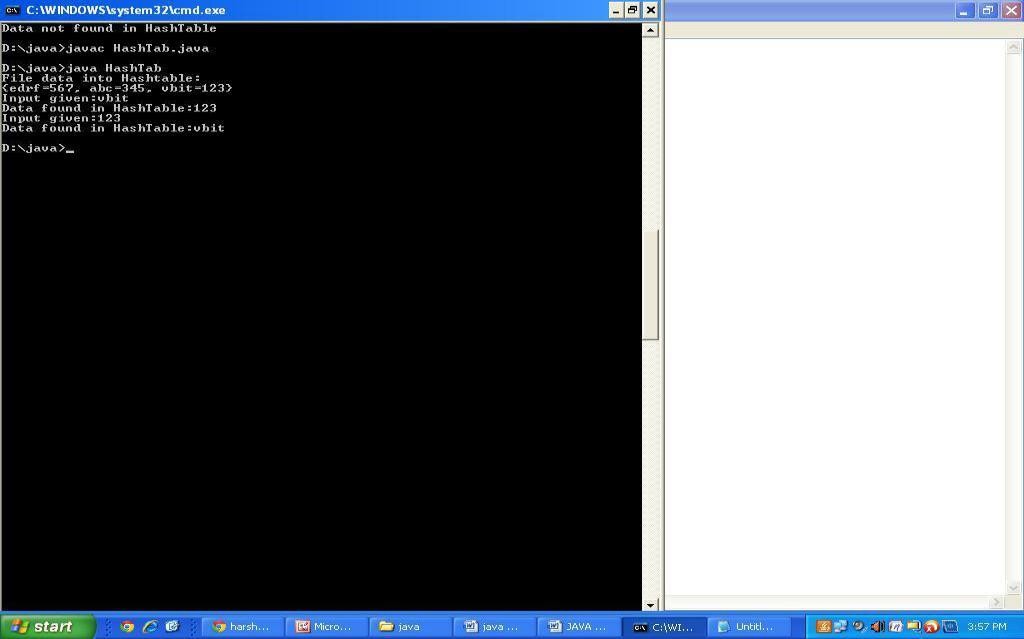
}

}

**HashTab.txt**

|  |  |
| --- | --- |
| abd | 123 |
| abc | 345 |
| edrf | 567 |

**Output:-**

****

**METHOD OVERLOADING**

class Adder

{

static int add(int a, int b){return a+b;}

static double add(double a, double b){return a+b;}

}

class TestOverloading2{

public static void main(String[] args){

System.out.println(Adder.add(11,11));

System.out.println(Adder.add(12.3,12.6));

}}

OUTPUT

22

24.9

**Why Method Overloading is not possible by changing the return type of method only?**

In java, method overloading is not possible by changing the return type of the method only because of ambiguity. Let's see how ambiguity may occur.

class Adder{

static int add(int a,int b){return a+b;}

static double add(int a,int b){return a+b;}

}

class TestOverloading3{

public static void main(String[] args){

System.out.Println(Adder.add(11,11));//ambiguity

}}

**OUTPUT:**

Compile Time Error: method add(int,int) is already defined in class Adder

**CONSTRUCTORS**

class Main {

private String name;

// constructor

Main() {

System.out.println("Constructor Called:");

name = "Program";

}

public static void main(String[] args) {

// constructor is invoked while

// creating an object of the Main class

Main obj = new Main();

System.out.println("The name is " + obj.name);

}

}

**OUTPUT**

Constructor called the name is program

**ARRAYS**

class Main {

public static void main(String[] args) {

// create an array

int[] age = {12, 4, 5, 2, 5};

// access each array elements

System.out.println("Accessing Elements of Array:");

System.out.println("First Element: " + age[0]);

System.out.println("Second Element: " + age[1]);

System.out.println("Third Element: " + age[2]);

System.out.println("Fourth Element: " + age[3]);

System.out.println("Fifth Element: " + age[4]);

}

}

**OUTPUT**

Accessing Elements of Array:

First Element: 12

Second Element: 4

Third Element: 5

**Java program to find the sum of positive numbers using while loop**

import java.util.Scanner;

class Main {

public static void main(String[] args) {

int sum = 0;

Scanner input = new Scanner(System.in);

System.out.println("Enter a number");

int number = input.nextInt();

while (number >= 0) {

// add only positive numbers

sum += number;

System.out.println("Enter a number");

number = input.nextInt();

}

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

input.close();

}

}

**Output**

Enter a number

25

Enter a number

9

Enter a number

5

Enter a number

-3

Sum = 39

**ARITHMTIC OPERATORS**

class Main {

public static void main(String[] args) {

int a = 12, b = 5;

System.out.println("a + b = " + (a + b));

System.out.println("a - b = " + (a - b));

System.out.println("a \* b = " + (a \* b));

System.out.println("a / b = " + (a / b));

System.out.println("a % b = " + (a % b));

}

}

**OUTPUT**

a + b = 17

a - b = 7

a \* b = 60

a / b = 2

a % b = 2

**FileOutputStream to write data to a File**

import java.io.FileOutputStream;

public class Main {

public static void main(String[] args) {

String data = "This is a line of text inside the file.";

try {

FileOutputStream output = new FileOutputStream("output.txt");

byte[] array = data.getBytes();

output.write(array);

output.close();

}

catch(Exception e) {

e.getStackTrace();

}}}