Write a Java Program which has an interface named arraymodify that has a static method named transpose which will convert and print the transpose of a matrix, a default method named arrayremove which will remove an element from an array, and a method named arrayinsert which will insert an element into an array. A class named arraymodifyclass will implement the arraymodify interface. Create objects of the class arraymodify and call the methods.

Transpose of a matrix is where the row is converted to column and column is converted to row

Suppose a matrix has row=2 and column=3

1 2 3

4 5 6

In a transpose of this matrix, the row will be equal to 3 and column will be equal to 2

1 4

2 5

3 6

Differences between built-in array and arraylist

In a built-in array you cannot simply modify the array that means it will be hard for to insert, remove or modify an element in any position inside a built-in array. However java has a class named ArrayList which you can use to create dynamical sized arrays. That means you can insert, remove or modify elements in any position inside an array by calling the methods from the ArrayList class

Syntax:

ArrayList <Data type> <name of the ArrayList>= new ArrayList<Data type>();

```
package ice107onlineclass;
import java.util.*;
interface arraymodify{
    static void transpose() {// A static method can
have a body inside an interface
        int row, column;
        Scanner input = new Scanner(System.in);
        System.out.println("Enter the number of
rows");
        row=input.nextInt();
        System.out.println("Enter the number of
columns");
        column=input.nextInt();
```

```
        int a[][]=new int[row][column];
        for(int i=0; i<row; i++) {
            for(int j=0; j<column; j++) {
                a[i][j]=input.nextInt();
                 // user is taking a matrix as input
            }
        }
        System.out.println("Before the transpose the
matrix is");
        for(int i=0; i<row; i++) {
            for(int j=0; j<column;j++) {
                System.out.print(a[i][j]);
            }
            System.out.println();
        }
        System.out.println("After the transpose the
matrix is");
        for(int i=0; i<column;i++ ) {
            for(int j=0; j<row;j++ ) {
                System.out.print(a[j][i]+" ");
                // here I am creating a transpose of
a matrix by converting row into columns and columns
into rows.
            }
            System.out.println();
        }
    }
    default void arrayremove() {// a default method
can have a body inside an interface
    ArrayList<Integer>list=new ArrayList<Integer>();

        list.add(10);// I am inserting an element in
the arraylist by using add() method
        list.add(35);
```

```
        list.add(42);
        list.add(50);
        list.add(60);
        list.add(70);
        System.out.println("Before removing an index,
the arraylist is " +list);
        list.remove(2); // I am removing an element
which is in the 2nd index of the arraylist “list” by
using remove() method
        System.out.println("After removing an index,
the arraylist is " +list);
    }
    void arrayinsert();
}
class arraymodifyclass implements arraymodify{
    public void arrayinsert() {
        ArrayList <Integer> node= new
ArrayList<Integer>();
        node.add(23);
        node.add(37);
        node.add(51);
        node.add(73);
        System.out.println("Before inserting an
element in an index the arraylist is " +node);
        node.add(2, 50); // Here I am adding element
“50” in the 2nd index of the ArrayList “node”
        System.out.println("After inserting an
element in an index the arraylist is " +node);
    }
}


public class inheritancestaticdefaultexample {
```

```
    public static void main(String[] args) {
        arraymodifyclass a1= new
arraymodifyclass();// TODO Auto-generated method stub
        arraymodify.transpose(); // I am calling the
static method “transpose” by using
interfacename.staticmethodname
        a1.arrayremove();
// I am calling the default method “arrayremove” by
using the a1 object.defaultmethodname
        a1.arrayinsert();
    }

}
```

```
package ice107onlineclass;

import java.awt.*;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;


import javax.swing.*;

// I have imported these packages to create a graphical application

class TextFieldFrame extends JFrame{

        private JTextField textField1; // I am creating a textfield

        private JTextField textField2;

        private JTextField textField3;

        private JPasswordField passwordField; // I am creating a password field

        TextFieldFrame(){

                super("Testing JTextField and JpassowordField");

               // I am setting the title of the JFrame

                setLayout(new FlowLayout());

               // I am setting the Layout of the JFrame

                textField1= new JTextField(10);// Here the size of the textfield1 will be 10.
```

```
                add(textField1); // I am adding textfield1 in the JFrame
                textField2= new JTextField("Enter text here");
                add(textField2);
                textField3= new JTextField("Uneditable Text Field",21);
                textField3.setEditable(false); // textfield3 cannot be edited
                add(textField3);
                passwordField= new JPasswordField("Hidden text");
                add(passwordField); // I am adding the passwordfield
                TextFieldHandler handler= new TextFieldHandler();
               // I am creating an handler so that it can listen to the action of the user
                textField1.addActionListener(handler);
                // textfield1 will now listen to the action of the user
                textField2.addActionListener(handler);
                textField3.addActionListener(handler);
                passwordField.addActionListener(handler);
        }
        class TextFieldHandler implements ActionListener{
                public void actionPerformed(ActionEvent event) {
                    // GUI application will perform action based on user input by using the
actionlistener interface and the actionperformed method
                        String string="";
                        if(event.getSource()==textField1) {// it is getting the input from the textfield1
                                string= String.format("textField1: %s", event.getActionCommand()); // it
is going to add the string based on the user action command in the text field1
                        }
                        else if(event.getSource()==textField2) {
                                string= String.format("textField2: %s", event.getActionCommand());
                        }
                        else if(event.getSource()==textField3) {
```

```
                    string= String.format("textField3: %s", event.getActionCommand());
                }
                else if(event.getSource()==passwordField) {
                    string= String.format("passwordField: %s", event.getActionCommand());
                }
                JOptionPane.showMessageDialog(null, string);
                // showMessageDialog method will show a dialog
            }
        }

}

public class Guiexample {

    public static void main(String[] args) {
        TextFieldFrame textfieldframe=new TextFieldFrame();
        textfieldframe.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);// TODO Auto-generated method stub
    textfieldframe.setSize(350,100);// I am setting the height and width of the JFrame
    textfieldframe.setVisible(true); // I am setting the visibility true for the JFrame
    }

}
```